Market Making: Comparison between Reinforcement Learning and Analytical Benchmarks

José E. Figueroa-López

Washington University in St. Louis

figueroa-lopez@wustl.edu

Stochastic Control in Finance, NUS, Singapore
July 22, 2019
Joint work with Agostino Capponi and Chuyi Yu
Market Makers (MM) **continually quote bid and ask prices** at which they are willing to buy and sell on both sides of a Limit Order Book;

If both orders get executed or lifted, the MM gains their quoted spread, which will be at least as big as the market spread;

There is a natural tradeoff when placing orders deeper in the book to maximize the profit, if executed, but with reduced chances of execution.

In addition, there is also considerable **inventory risk** in which the MM may accumulate large amount of inventory that must be cleared aggressively at the end of day at suboptimal prices or even worst at market prices.
Avellaneda and Stoikov (2008):
- Maximization of the exponential utility from terminal trading profit/loss $W_T$ and residual inventory $I_T$ liquidation: $E[-e^{-\gamma(W_T+I_T S_T)}]$;
- Optimize bid/ask LO placements $S_t \pm \delta^\pm_t$ of one share unit under a Brownian midprice dynamics $S_t = \sigma B_t$ and Poisson MOs arrival times with lifting rates $\pi_S(\delta^-)$ and $\pi_B(\delta^+)$. 

Cartea and Jaimungal (2015):
- Optimize terminal wealth $W_T$ and residual inventory liquidation while penalizing terminal and running inventory:
\[ \sup_{(\delta^\pm_s)_{t \leq s \leq T}} E\left[ W_T + I_T S_T - \lambda I_T^2 - \phi \int_t^T I_s^2 ds \bigg| W_{t^-} = w, I_{t^-} = i, S_{t^-} = s \right] \]
- Still optimize bid/ask LO placements $S_t \pm \delta^\pm_t$ of one unit, which is lifted with probability $e^{-\kappa \pm \delta^\pm_t}$ when a Poisson MO arrives;
- Brownian midprice dynamics between MOs but with i.i.d. jumps at MOs arrival times (price impact of MOs);
Very Selective Literature on Market Making


- Optimize $\sup_{(\delta^\pm)_t \leq s \leq T} E_t[W_T + l_T S_T - \lambda I_T^2]$, but allows placement of orders of arbitrary (but fixed) volume $\vartheta$;

- Use of exogenously specified linear demand and supply functions to determine number of lifted shares at the arrival of MOs;

- Brownian midprice dynamics and Poissonian arrival times of MOs;

Chan and Shelton (2001):

- Apply reinforcement learning techniques in a simulated market environment; Mid price is the difference of Poisson processes: $S_t = (N_t^+ - N_t^-) \times$ spread.

- Focus on an information-based model. Poissonian arrivals of informed and uniformed traders.


- Train the reinforcement learning market making agent in a realistic, data-driven limit order book. **Actions taken at every LOB event.**
Our Objectives

- Assess the performance of the reinforcement learning techniques under modeled market dynamics with analytical optimal policies.

- Specifically, we devise a discrete-time version of the model in Adrian, Capponi, Vogt, & Zhang (2018) with actions taking place at either the arrival times of MOs or at prespecified discrete times $t_j$.

- Apply reinforcement learning techniques to deal with nonlinear demand functions and compare the RL policies with approximate analytical benchmarks.
A Simple Market Making Model I

- Finite time horizon $[0, T]$
- The fundamental price is denoted by $\{S_t\}_{t \geq 0}$
- Market orders (MOs) (either buy or sell) arrive in the market at times $0 < \tau_1 < \tau_2 < \ldots$. Let

$$\alpha_j = \begin{cases} 
1, & \text{if the MO at } \tau_j \text{ is to sell}, \\
-1, & \text{if the MO at } \tau_j \text{ is to buy}.
\end{cases}$$

- The counting processes of sell and buy MOs are denoted by $N^S_t$ and $N^B_t$, respectively:

$$N^S_t = \#\{j : \tau_j \leq t, \alpha_j = 1\}, \quad N^B_t = \#\{j : \tau_j \leq t, \alpha_j = -1\}$$

$$N_t = N^S_t + N^B_t.$$

- We consider two cases for the times $t_j$’s of action or control:

  - Case 1: Actions at the arrivals of MOs: $t_0 = 0$, $t_j = \tau_j \land T$,
  - Case 2: Actions at $n$ fixed regular times: $t_j = \frac{T}{n}j$. 
The following give the details in 1st setting (2nd case is similar).

- At times $t_j < T$ ($j = 0, 1, \ldots, N_T$), the market maker places simultaneous bid and ask limit orders (LOs) of constant volume $\vartheta$ at levels

$$b_{t_j} = S_{t_j} - \delta_j^-, \quad a_{t_j} = S_{t_j} + \delta_j^+,$$

where $0 \leq \delta_j^-, \delta_j^+ \in \mathcal{F}_{t_j}$, information available at $t_j$.

- The LOs placed at time $t_j$ will stay in the market until the next arrival of MO or until the terminal time $T$. We set

$$a_t = a_{t_j}, \quad b_t = b_{t_j}, \quad t \in [t_j, t_{j+1}).$$

(Remark: Note that $a_{t_j}^- = a_{t_{j-1}}$ and $b_{t_j}^- = b_{t_{j-1}}$, for $j = 1, 2, \ldots$).
After placing LOs at time $t_j$, there are three possibilities:

(i) If $t_{j+1} = T$ (i.e. $t_j$ is the arrival time of the last MO before termination), the market maker liquidates her inventory $I_T$ at midprice for a cashflow of $S_T I_T$;

(ii) If $t_{j+1} < T$ and $\alpha_{j+1} = 1$ (i.e., $t_{j+1}$ is the time of a sell MO), then $Q^S(S_{t_{j+1}}, b_{t_j})$ shares are filled at bid price $b_{t_j}$;

(iii) If $t_{j+1} < T$ and $\alpha_{j+1} = -1$ (i.e., $t_{j+1}$ is the time of a buy MO), then $Q^B(S_{t_{j+1}}, a_{t_j})$ shares are filled at ask price $a_{t_j}$;

$Q^S(s', b)$ and $Q^B(s', a)$ are given deterministic supply/demand functions;

Remark: In the cases (ii)-(iii), the unexecuted LOs are “immediately” canceled and new LO’s placement positions are taken right after at levels $a_{t_{j+1}}$ and $b_{t_{j+1}}$. 
The agent aims to maximize her final wealth at time $T$:

$$V^*_0 = \sup_{\delta^+_j, \delta^-_j} \mathbb{E} \left[ W_T + S_T I_T \mid S_0 = s, I_0 = i, W_0 = w \right],$$

where, for $t \in (0, T]$, 

$$W_t = W_0 + \int_0^t a_u^- Q^B(S_u, a_u^-) dN_u^B - \int_0^t b_u^- Q^S(S_u, b_u^-) dN_u^S,$$

$$I_t = I_0 - \int_0^t Q^B(S_u, a_u^-) dN_u^B + \int_0^t Q^S(S_u, b_u^-) dN_u^S.$$
Linear Supply/Demand Functions

In terms of a **given fixed constant** $c$, the linear supply/demand functions ([Adrian et. al., 2018],[Hendershott and Menkveld, 2014]) are given by:

$$Q^S(s', b) = \vartheta + c(b - s'), \quad Q^B(s', a) = \vartheta + c(s' - a)$$

- At the arrival of a Sell MO at time $t_{j+1}$, all of the agent’s volume $\vartheta$ is filled if $S_{t_{j+1}} = b_{t_j}$; otherwise, this amount decays linearly as $S_{t_{j+1}}$ is farther away (above) $b_{t_j}$.

- Similarly, at the arrival of a Buy MO at time $t_{j+1}$, the agent sell all its volume $\vartheta$ if $S_{t_{j+1}} = a_{t_j}$; otherwise, this amount decays linearly as $S_{t_{j+1}}$ is farther away (below) $a_{t_j}$.

In what follows, it would be useful to write $Q^S$ and $Q^B$ in terms of $p := \vartheta/c$ and $c$ (instead of $\vartheta$ and $c$):

$$Q^S(s', b) = c(p + b - s'), \quad Q^B(s', a) = c(p + s' - a).$$
Theorem (J. F-L, A. Capponi, & C. Yu ’19)

Assume the following conditions:

Linear Demand Functions: \[
\begin{align*}
Q^S(s', b) &= c(p + b - s') \\
Q^B(s', a) &= c(p + s' - a)
\end{align*}
\]

Martingale Condition: \[
\mathbb{E}[\mathbf{1}_{t+1 < T, \alpha_{t+1} = \pm 1} (S_{t+1} - S_t) | \mathcal{F}_t] = 0
\]

Then, the optimal placement policy and achieved expected maximal wealth are given by

\[
a^*_j = S_t + \frac{1}{2}p, \quad b^*_j = S_t - \frac{1}{2}p, \quad j = 0, 1, \ldots, N_T,
\]

\[
V^*_0 = W_0 + \frac{cp^2}{4} \mathbb{E}[N_T] - c \mathbb{E} \left[ \sum_{i=0}^{N_T-1} (S_{t_{i+1}} - S_{t_i})^2 \right].
\]
Nonlinear Demand Functions

- Even though linear supply/demand functions
  \[ Q^S(S_{t+1}, b_t) = c(p + b_t - S_{t+1}), \quad Q^B(S_{t+1}, a_t) = c(p + S_{t+1} - a_t) \]
  allow us to find explicit optimal policies, they are not realistic in that
  \( Q^S \) and \( Q^B \) give negative values when \( S_{t+1} - p > b_t \) and
  \( S_{t+1} + p < a_t \)

- It is then natural to constrain the demand functions to be positive:
  \[
  \tilde{Q}^S(S_{t+1}, b_t) = \max\{0, c(p + b_t - S_{t+1})\}
  \]
  \[
  \tilde{Q}^B(S_{t+1}, a_t) = \max\{0, c(p + S_{t+1} - a_t)\}
  \]

- Similarly, if we think of \( \vartheta = cp \) as the initial volume of placed orders,
  it is not natural to have demand/supply larger than \( cp \). Hence, we
  also consider
  \[
  \tilde{\tilde{Q}}^S(S_{t+1}, b_t) = \min\{\max\{0, c(p + b_t - S_{t+1})\}, cp\}
  \]
  \[
  \tilde{\tilde{Q}}^B(S_{t+1}, a_t) = \min\{\max\{0, c(p + S_{t+1} - a_t)\}, cp\}
  \]
Alternative Optimal Placement Policies

- There is no closed form solution for the optimal placement policy under the previous modified demand functions.
- Two possibilities to deal with this is to impose some simplifying constraints on the policies \( a_{tj} = S_{tj} + \delta_j^- p \) and \( b_{tj} = S_{tj} - \delta_j^+ p \) such as
  - **Time-invariance:**
    \[ \delta_j^+ \equiv \delta^+ \text{ and } \delta_j^- \equiv \delta^- \], for all \( j \).
  - **Myopic or one-step ahead optimality:**
    Maximize the immediate expected reward:
    \[
    \max_{\delta_j^\pm \in F_{tj}} \mathbb{E}[R_{tj+1} | F_{tj}],
    \]
    \[ R_{tj+1} = (W_{tj+1} - W_{tj}) + (S_{tj+1} l_{tj+1} - S_{tj} l_{tj}) \]

- While we expect the 2nd strategy to yield more optimal strategies, they are also more computationally expensive. So, in practice, the 1st strategy may suffice.
When imposing the restriction $\hat{Q}^{S,B} \geq 0$, the immediate reward at time $t_{j+1} < T$ (ignoring $I_t (S_{t_{j+1}} - S_t)$) takes the form:

$$R_{t_{j+1}} = 1_{\{\alpha_{j+1}(s_{t_{j+1}} - s_t) \leq (1 - \delta \alpha_{j+1})p\}} \left[ c p^2 \delta \alpha_{j+1} (1 - \delta \alpha_{j+1}) - c (S_{t_{j+1}} - S_t)^2 + \alpha_{j+1} c p (1 - 2 \delta \alpha_{j+1}) (S_{t_{j+1}} - S_t) \right]$$

depending on whether $\alpha_{j+1} = 1$ (sell) or $\alpha_{j+1} = -1$ (buy).

It can be shown that the optimal placement is symmetric:

$$\delta := \delta^+ = \delta^-$$

Assuming that $\{N_t\}$ and $\{S_t\}$ are independent and conditioning on $\{N_t\}$, we get the expected reward

$$\hat{V}_0^* = \mathbb{E} \left[ \sum_{j=0}^{N_T - 1} \left\{ c p^2 \delta (1 - \delta) \Phi \left( \frac{(1 - \delta)p}{\sigma \sqrt{t_{j+1} - t_j}} \right) - c \sigma^2 (t_{j+1} - t_j) \Phi \left( \frac{(1 - \delta)p}{\sigma \sqrt{t_{j+1} - t_j}} \right) + c p \sigma \delta \sqrt{t_{j+1} - t_j} \phi \left( \frac{(1 - \delta)p}{\sigma \sqrt{t_{j+1} - t_j}} \right) \right\} \right]$$
Recall that $\tilde{Q}^{S,B}$ imposes both conditions $0 \leq \tilde{Q}^{S,B} \leq cp$.

The immediate reward at time $t_{j+1} < T$ (ignoring $I_t (S_{t_{j+1}} - S_t)$), takes the form:

$$R_{t_{j+1}} = 1\{\delta \pm p \leq \pm (S_{t_{j+1}} - S_t) \leq (1 - \delta \pm) p\} \left[ cp^2 \delta \pm (1 - \delta \pm) - c(S_{t_{j+1}} - S_t)^2 \right. \\
+ \left. \pm cp(1 - 2\delta \pm)(S_{t_{j+1}} - S_t) \right]$$

$$+ 1\{\pm (S_{t_{j+1}} - S_t) \leq -\delta \pm p\} cp \left[ \pm (S_{t_{j+1}} - S_t) + \delta \pm p \right],$$

depending on whether $\alpha_{j+1} = 1$ (sell) or $\alpha_{j+1} = -1$ (buy).

Again, the optimal policy is symmetric (i.e., $\delta := \delta^+ = \delta^-$). So, we can focus on such policies.
By conditioning on \( \{N_t\}_{t \geq 0} \) (assuming independence between \( N \) and \( S \)), the expected reward now takes the form

\[
\tilde{V}_0^* = \hat{V}_0^* + \mathbb{E} \left[ \sum_{j=0}^{N_T-1} cp^2 \delta^2 \Phi \left( \frac{-\delta p}{\sigma \sqrt{t_{j+1} - t_j}} \right) \right]
\]

\[
+ c \sigma^2 (t_{j+1} - t_j) \Phi \left( \frac{-\delta p}{\sigma \sqrt{t_{j+1} - t_j}} \right)
\]

\[
- cp \sigma \delta \sqrt{t_{j+1} - t_j} \phi \left( \frac{\delta p}{\sigma \sqrt{t_{j+1} - t_j}} \right) \}
\]

where \( \hat{V}_0^* \) is the expected total reward for the previous case.
Under Poisson arrival rates, we compute $\hat{V}_0^*, \tilde{V}_0^*$ by Monte Carlo$^1$:

<table>
<thead>
<tr>
<th>Demand Functions</th>
<th>Expected Total Reward ($V^*$)</th>
<th>Optimal Placement ($\delta^*$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q^S(S_{t_j+1}, b_{t_j}) = c(p + b_{t_j} - S_{t_j+1})$</td>
<td>0.016</td>
<td>0.50</td>
</tr>
<tr>
<td>$Q^B(S_{t_j+1}, a_{t_j}) = c(p + S_{t_j+1} - a_{t_j})$</td>
<td>0.163</td>
<td>0.65</td>
</tr>
<tr>
<td>$\tilde{Q}^S(S_{t_j+1}, b_{t_j}) = \max{0, c(p + b_{t_j} - S_{t_j+1})}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tilde{Q}^B(S_{t_j+1}, a_{t_j}) = \max{0, c(p + S_{t_j+1} - a_{t_j})}$</td>
<td>0.196</td>
<td>0.60</td>
</tr>
</tbody>
</table>

$^1$Parameters as in [Adrian et. al., 2018]: $T = 10$ (seconds), $\lambda^S = \lambda^B = 10$, $c = 30$, $p = 0.017$, $\sigma = 0.0375$, $S_0 = $100, the initial inventory and cash holding are both 0.
Goal: Maximize the immediate expected reward:

\[
\max_{\delta_j^+ \in \mathcal{F}_t} \mathbb{E}[R_{tj+1} | \mathcal{F}_t]
\]

This is equivalent to maximizing

\[
\mathbb{E} \left[ \mathbf{1}_{\{t_{j+1} < T\}} \left\{ cp^2 \delta(1 - \delta) \Phi \left( \frac{(1 - \delta)p}{\sigma \sqrt{t_{j+1} - t_j}} \right) - c\sigma^2(t_{j+1} - t_j) \Phi \left( \frac{(1 - \delta)p}{\sigma \sqrt{t_{j+1} - t_j}} \right) \right. 
\left. + cp\sigma\delta \sqrt{t_{j+1} - t_j} \phi \left( \frac{(1 - \delta)p}{\sigma \sqrt{t_{j+1} - t_j}} \right) \right\} \mathcal{F}_t \right]
\]

For most values of \(t\) (\(t < 0.9 T\)), \(\delta^* \approx 0.65\). As \(t\) gets very close to \(T\), \(\delta^*\) moves quickly to 0.5. A similar phenomenon is observe when using the demand functions \(\widehat{Q}^{S,B}\). For \(t < 0.9 T\), \(\delta^* \approx 0.6\). As \(t\) gets very close to \(T\), \(\delta^*\) moves quickly to 0.5.
The agent in reinforcement learning (RL) learns to make the optimal decisions by interacting with the environment.

No knowledge of the market environment, such as the order flow or price process, is assumed.

Trajectory: $S_0^e, A_0, R_1, S_1^e, A_1, R_2, S_2^e, A_2, R_3, \ldots$

In most cases, a **Markov Decision Process** is assumed:

$$p(s', r | s, a) = \mathbb{P}[S_{t+1}^e = s', R_{t+1} = r | S_t^e = s, A_t = a].$$
Policy and Action-Value Functions

- Policy $\pi(a|s)$: the probability of selecting action $a$ under state $s$.
- The goal of the RL agent is to choose the policy $\pi$ to maximize the expected discounted cumulative reward:

$$v^*(s) := \max_{\pi} v_\pi(s) := \max_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t^e = s \right]$$

- Equivalently, we can first find the $q$-function:

$$q^*(s, a) := \max_{\pi} \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S^e_t = s, A_t = a \right],$$

and, then, the optimal policy as:

$$v^*(s) = \sup_a q^*(s, a), \quad \pi^*(s) = \text{argmax}_a q^*(s, a)$$
Bellman Optimality Equation

- For the value function,

\[ v_*(s) = \max_a \mathbb{E} \left[ R_{t+1} + \gamma v_*(S^e_{t+1}) \middle| S^e_t = s, A_t = a \right] \]

- For the q-function,

\[ q_*(s, a) = \mathbb{E} \left[ R_{t+1} + \gamma \max_{a'} q_*(S^e_{t+1}, a') \middle| S^e_t = s, A_t = a \right] \]

- Reinforced Learning Algorithms Build on The Bellman’s Equations Above and a Combination of Monte Carlo and Fixed Point Steps.
Suppose we want to evaluate \( \mu := \mathbb{E}(X) \) based on independent replicas \( X_1, X_2, \ldots \). A useful iterative method to evaluate \( \mu \) starts with \( \hat{\mu}_0 = 0 \) and set:

\[
\hat{\mu}_1 = \hat{\mu}_0 + \alpha[X_1 - \hat{\mu}_0] = \alpha X_1 \\
\hat{\mu}_2 = \hat{\mu}_1 + \alpha[X_2 - \hat{\mu}_1] = \alpha X_2 + \alpha(1 - \alpha)X_1 \\
\hat{\mu}_k = \hat{\mu}_{k-1} + \alpha[X_k - \hat{\mu}_{k-1}] = \alpha \sum_{i=0}^{k-1} (1 - \alpha)^i X_{k-i}
\]

1st and 2nd order Properties of \( \hat{\mu}_k \):

\[
\mathbb{E}[\hat{\mu}_k] = \alpha \sum_{i=0}^{k-1} (1 - \alpha)^i \mu = (1 - (1 - \alpha)^k)\mu \xrightarrow{k \to \infty} \mu \\
\text{Var}(\hat{\mu}_k) = \alpha^2 \sum_{i=0}^{k-1} (1 - \alpha)^{2i} \sigma_X^2 = \frac{\alpha(1 - (1 - \alpha)^{2k})}{2 - \alpha} \sigma_X^2 \xrightarrow{k \to \infty} \frac{\alpha}{2 - \alpha} \sigma_X^2.
\]
Monte Carlo Step (Step-Dependent Rate)

- To ensure consistence, we need $\alpha = \alpha_i \downarrow 0$:

$$\hat{\mu}_1 = \hat{\mu}_0 + \alpha_1[X_1 - \hat{\mu}_0] = \alpha_1 X_1$$
$$\hat{\mu}_2 = \hat{\mu}_1 + \alpha_2[X_2 - \hat{\mu}_1] = \alpha_2 X_2 + \alpha_1(1 - \alpha_2)X_1$$
$$\hat{\mu}_3 = \hat{\mu}_2 + \alpha_3[X_3 - \hat{\mu}_2] = \alpha_3 X_3 + \alpha_2(1 - \alpha_3)X_2 + \alpha_1(1 - \alpha_2)(1 - \alpha_3)X_1$$

$$\hat{\mu}_k = \hat{\mu}_{k-1} + \alpha_k[X_k - \hat{\mu}_{k-1}] = \sum_{i=1}^{k} \alpha_i \prod_{j=i+1}^{k} (1 - \alpha_j)X_i$$

- We need that:

$$\mathbb{E}[\hat{\mu}_k] = \sum_{i=1}^{k} \alpha_i \prod_{j=i+1}^{k} (1 - \alpha_j) \mu \xrightarrow{k \to \infty} \mu$$

$$\text{Var}(\hat{\mu}_k) = \sum_{i=1}^{k} \alpha_i^2 \prod_{j=i+1}^{k} (1 - \alpha_j)^2 \sigma_X^2 \xrightarrow{k \to \infty} 0.$$  

- Fact: $\alpha_i = \frac{1}{i}$ suffices.
Fixed Point Step

In this step, we evaluate $q_*$ using a type of fixed point algorithm:

$$q_{*,0}(s, a) = \text{Initialize a q-function}$$

$$q_{*,1}(s, a) = \mathbb{E} \left[ R_{t+1} + \gamma \max_{a'} q_{*,0}(S_{t+1}^e, a') \middle| S_t^e = s, A_t = a \right]$$

$$q_{*,2}(s, a) = \mathbb{E} \left[ R_{t+1} + \gamma \max_{a'} q_{*,1}(S_{t+1}^e, a') \middle| S_t^e = s, A_t = a \right]$$

$$\vdots$$
Q-learning Algorithm

Initialize $Q(s, a)$ for all states $s$ and actions $a$;

Repeat (for each episode):
  Set initial state $S^e_0$
  Repeat for each step $t = 0, 1, \ldots$ of episode
    Choose action $A_t$ from $S^e_t$ based on $Q$ by $\varepsilon$-greedy method:
    $$A_t = \begin{cases} \arg\max_a Q(S^e_t, a), & \text{w.p. } 1 - \varepsilon, \\ \text{any other action } a, & \text{with equal prob. } \frac{\varepsilon}{k-1} \end{cases}$$
    Take action $A_t$, observe $R_{t+1}, S^e_{t+1}$
    $$Q(S^e_t, A_t) \leftarrow Q(S^e_t, A_t) + \alpha [R_{t+1} + \gamma \max_{a'} Q(S^e_{t+1}, a') - Q(S^e_t, A_t)]$$
    $$S^e_t \leftarrow S^e_{t+1}$$
  until $S^e_t$ is terminal
When the state space is continuous, we apply tile coding method to approximate the value function $Q(s, a)$.

A partition of the continuous space, but with more than one layer (tiling).

Low-dimensional continuous state $s \rightarrow$ High dimensional binary feature vector $x(s) = [x_1(s), \ldots, x_\ell(s)]$.

Example: Consider tiling a unit square into 4 tilings of $4 \times 4$. We have $[0, 0] \rightarrow [1, 1, 1, 1, 0, 0, \ldots, 0]_{64}$  
$[0, 0.1] \rightarrow [1, 0, 1, 1, 1, 0, 0, \ldots, 0]_{64}$
The action-value function can then be approximated by assigning a weight to each tile:

\[
\hat{q}(s, a, w) = \sum_i w_i \cdot x_i(s, a)
\]

Close points have close values in \(\hat{q}\)

Instead of updating the action-value function for each state and action, tile coding method only needs to update the weights of tiles

Apply Stochastic Gradient Descent (SGD) to update the weight parameter \(w\) during the training; i.e., change

\[
w_{t+1} \leftarrow w_t + \alpha [R_{t+1} + \gamma \max_{a'} \hat{q}(S_{t+1}^e, a', w_t) - \hat{q}(S_t^e, A_t, w_t)] \nabla_w \hat{q}(S_t^e, A_t, w_t)
\]

instead of

\[
Q(S_t^e, A_t) \leftarrow Q(S_t^e, A_t) + \alpha [R_{t+1} + \gamma \max_{a'} Q(S_{t+1}^e, a') - Q(S_t^e, A_t)]
\]
Reinforced Learning Implementation Of Market-Making

- **State Space:** $S^e_t = (S_t, I_t, t)$.
- **Action Space:** 10 available actions for RL agent: Placing bid and ask LOs simultaneously at prices
  
  \[
  \begin{align*}
  a_t &= S_t + \delta p & \text{for Ask LO} \\
  b_t &= S_t - \delta p & \text{for Bid LO}
  \end{align*}
  \]

  where $\delta$ is chosen among $[0.1, 0.2, ..., 1]$.
- **Design of the Immediate Reward** ($\lambda = 0$):
  
  \[
  R_j = (W_{t_j+1} - W_{t_j}) + (S_{t_j+1} I_{t_j+1} - S_{t_j} I_{t_j}), \quad j = 0, 1, \ldots
  \]

  and discount rate $\gamma = 1$, so that

  \[
  R = \sum_{j=0}^{\infty} \left\{ (W_{t_j+1} - W_{t_j}) + (S_{t_j+1} I_{t_j+1} - S_{t_j} I_{t_j}) \right\} = W_T + S_T I_T
  \]

  The objective to maximize in RL is the same as that in the previous control problem.
RL Policy for $Q^{S,B}(\cdot)$

- $Q^S(S_{t+1}, b_t) = c(p + b_t - S_{t+1})$, $Q^B(S_{t+1}, a_t) = c(p + S_{t+1} - a_t)$
- $\delta^* = 0.5$, $V^* = 0.016$

<table>
<thead>
<tr>
<th>Optimal Action (a=S+0.5p, b=S-0.5p)</th>
<th>RL</th>
<th>Random policy</th>
<th>RL-Random Policy</th>
<th>Optimal Action - RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.018</td>
<td>0.005</td>
<td>-0.127</td>
<td>0.132</td>
<td>0.013</td>
</tr>
</tbody>
</table>

Optimal policy given by RL at different time $t$:

![Images showing the optimal policy at different times](image1.png)
RL Policy for $\hat{Q}^{S,B}(\cdot)$

- $\hat{Q}^S(S_{t+1}, b_t) = \max\{0, c(p + b_t - S_{t+1})\}$, $\hat{Q}^B(S_{t+1}, a_t) = \max\{0, c(p + S_{t+1} - a_t)\}$
- Time-invariant case: $\delta^* \approx 0.65$, $\hat{V}^* \approx 0.163$ for ($t < 0.9T$)
- Myopic case: $\delta^* \approx 0.65$ at most of the time and moves to 0.5 quickly when $t \rightarrow T$

### Average Final Reward in Testing Simulations for Multiple Strategies

<table>
<thead>
<tr>
<th>Benchmark (a=S+0.65p, b=S-0.65p)</th>
<th>RL</th>
<th>Random policy</th>
<th>RL-Random Policy</th>
<th>Benchmark - RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.161</td>
<td>0.147</td>
<td>0.059</td>
<td>0.088</td>
<td>0.014</td>
</tr>
</tbody>
</table>

Optimal policy given by RL at different time $t$:
RL Policy for $\tilde{Q}^{S,B}(\cdot)$

- $\tilde{Q}^{S}(S_{t+1}, b_{t}) = \min \{ \max \{ 0, c(p + b_{t} - S_{t+1}) \}, c p \}$,
  
- $\tilde{Q}^{B}(S_{t+1}, a_{t}) = \min \{ \max \{ 0, c(p + S_{t+1} - a_{t}) \}, c p \}$

- Time-invariant case: $\delta^{*} \approx 0.6$, $\tilde{V}^{*} \approx 0.196$
  
  Myopic case: $\delta^{*} \approx 0.6$ at most of the time and moves to 0.5 quickly when $t \to T$

<table>
<thead>
<tr>
<th>Benchmark (a=S+0.6p, b=S-0.6p)</th>
<th>RL</th>
<th>Random policy</th>
<th>RL-Random Policy</th>
<th>Benchmark - RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.195</td>
<td>0.181</td>
<td>0.120</td>
<td>0.061</td>
<td>0.014</td>
</tr>
</tbody>
</table>

Optimal policy given by RL at different time $t$:
Conclusion

Market making models with linear/non-linear demand functions:

- For linear demand functions, we get an analytical optimal placement policy with only a martingale condition on the fundamental price.

- For non-linear demand functions with lower bound zero and those with both upper and lower bounds, we present approximate optimal placement policies with time-invariance assumption and myopic assumption, respectively.

We train a RL agent in a simulated environment from the market making model above:

- Overall the optimal RL policies is consistent with the benchmarks. But the states near the boundary are seldomly visited by RL agent and thus the optimal policies on those states haven’t been reached yet.

- Flat total rewards for different actions leads to 'islands' of similar RL policies under some states.
Future Work I

Improvement on Market Making Model:

- Penalty on large inventory
  - $\max \mathbb{E}(W_T + I_T S_T - \lambda I_T^2)$: penalty on end-of-day inventory
  - $\max \mathbb{E}(W_T + I_T S_T - \lambda \int_0^T I_t^2)$: penalty on intraday inventory

- Model bid and ask prices in LOB instead of the fundamental price

- Consider the impact of our agent’s placements on the market:
  - the arrival rate and volume of the incoming MOs
  - price impact
Improvement on Reinforcement Learning Technique:

- When the underlying model gets more complex,
  - Incorporate action-critic method to allow continuous action space
  - Apply model with higher flexibility (e.g. neural network) to approximate action value function $q(s, a)$ in high-dimensional problem.

Reinforcement learning helps improve the model?

- Apply reinforcement learning result as a benchmark to check the validity of the proposed model on real world problem.
References


Appendix: Outline Of Theorem Proof

Set $W_0 = I_0 = 0$ and write $b_{t_j} = S_{t_j} - \delta_j^- p$ and $a_{t_j} = S_{t_j} + \delta_j^+ p$ with $\delta_j^\pm \in \mathcal{F}_{t_j}$. The final reward can be decomposed in a sequence of immediate rewards:

$$
W_T + S_T I_T = \sum_{j=0}^\infty \left\{ (W_{t_{j+1}} - W_{t_j}) + (S_{t_{j+1}} I_{t_{j+1}} - S_{t_j} I_{t_j}) \right\} =: \sum_{j=0}^\infty R_{t_{j+1}}.
$$

If $t_j = T$, $R_{t_{j+1}} = 0$. If $t_{j+1} = T$ and $t_j < T$, $R_{t_{j+1}} = I_{t_j} (S_{t_{j+1}} - S_{t_j})$, which has mean 0. If $t_{j+1} < T$, then

$$
R_{t_{j+1}} = cp^2 (1 - \delta_j^{\alpha j+1}) \delta_j^{\alpha j+1} - c (S_{t_{j+1}} - S_{t_j})^2 \\
\quad \quad \quad + cp\alpha_j (1 - 2\delta_j^{\alpha j+1})(S_{t_{j+1}} - S_{t_j}) + I_{t_j} (S_{t_{j+1}} - S_{t_j}).
$$

The “martingale condition” implies that the contribution of the last two terms is 0. The first term is maximal when $\delta_j^\pm = \frac{1}{2}$, while the second term does not depend on the $\delta_j$'s.