Predictive Market Making via Machine Learning

Abbas Haider¹ · Hui Wang¹ · Bryan Scotney¹ · Glenn Hawe¹

Received: 12 October 2020 / Accepted: 6 January 2022 / Published online: 26 January 2022

© The Author(s) 2022

Abstract

Market making (MM) is an important means of providing liquidity to the stock markets. Recent research suggests that reinforcement learning (RL) can improve MM significantly in terms of returns. In the latest work on RL-based MM, the reward is a function of equity returns, calculated based on its current price, and the inventory of MM agent. As a result, the agent's return is maximised and liquidity is provided. If the price movement is known and this information is optimally utilised, there is potential that the MM agent's return can be further improved. Important questions are, how to predict stock price movement, and how to utilise such prediction? In this paper, we introduce the concept of predictive market marking (PMM) and present our method for PMM, which comprises a RL-based MM agent and a deep neural network (DNN)-based price predictor. A key component of PMM is the consolidated price equation (CPE), which amalgamates an equity's present and predicted market prices into a consolidated price, which is used to generate ask and bid quotes that reflect both current price and future movement. Our PMM method is evaluated against the state-of-the-art (RL-based MM) and a traditional MM method, using ten stocks and three exchange traded funds (ETFs). Out-of-sample backtesting showed that our PMM method outperformed the two benchmark methods.

Keywords Market making · Reinforcement learning · Deep neural network · Asset price prediction

✉ Abbas Haider
haider-a@ulster.ac.uk

Hui Wang
h.wang@ulster.ac.uk

Bryan Scotney
bw.scotney@ulster.ac.uk

Glenn Hawe
gi.hawe@ulster.ac.uk

¹ School of Computing, Ulster University, Northern Ireland, UK

1 Introduction

Market making (MM) is a well-known high frequency trading (HFT) strategy widely used in large stock exchanges around the world including NYSE and NASDAQ [1]. Unlike MM however, HFTs are not obligated to always trade, and therefore two are dissimilar. A market maker or a MM agent is solely responsible for providing market liquidity through high-speed execution of a large number of orders within a fraction of a second. The market becomes illiquid in the absence of MM agents as orders will take time to execute, so the market participants lose interest in trading. MM therefore is a crucial component of the market and is a means of attracting investors. The profit in MM comes from the difference between the quoted ask (sell) and the quoted bid (buy) price of a stock. A MM firm has an obligation to continuously place buying and selling limit orders to add liquidity to the market. Each stock listed on the market for trading has a highest-bid price (best-bid) and a lowest-ask price (best-ask), and the difference between best-bid and best-ask at an instant, is known as market spread. The MM agent stands ready to continuously buy and sell stocks from other market participants during market operational hours in order to add liquidity to the market. Market liquidity can be measured by the quoted spread of a MM agent and the number of successful trades; hence, the higher the number of trades and lower the quoted spread, the more liquid the market becomes.

With the advent of algorithmic trading, humans vanish from the market in MM roles, except over-the-counter markets, e.g. corporate bonds market, as it would no longer be possible to make profit manually. MM is nowadays completely based on high-speed trading systems in order to convert the speed advantage into profit. However, the high-speed trading alone is not enough for MM agents to compete with other traders. There is a need for automated MM which can incorporate human level expertise into high-speed trading. There exists a machine learning technique known as reinforcement learning (RL) which is capable of being an automated MM agent. In the latest work on RL-based MM, [2] developed an automated MM agent using RL where the goal of the agent is to maximise the profit and minimize the inventory. This work is the state-of-the-art and is used as a baseline model in this paper to evaluate our new method. Moreover, this work is the first attempt to design a practical model of MM considering all necessary market phenomenon's including price to ticks conversion, limit order book handling operations such as insertion, deletion and amendment of orders and latency in order executions. For simplicity, we refer to this work by *RMM-Spooner*. *RMM-Spooner* used a well-known state discretization function approximation method of RL called *tile-codings* to design a RL-based MM agent.

We also use a traditional MM model by [3] as the second baseline model for comparison with our proposed method. This traditional model used a reservation price to calculate the optimal bid and ask quotes of MM agent. For simplicity, we refer to this traditional MM model by *AS model*. Existing MM methods

consider only the current price. If MM additionally takes into consideration the predicted future price, we call it predictive market making (PMM). If the prediction is accurate enough, higher profits are expected with PMM. There is ample evidence in the literature that HFT performance can be improved significantly by accurate prediction of future prices, see, for example, [4–6]. However, as far as we know, there is no existing study on PMM in the literature. In this paper we study PMM by answering the following questions: 1) How to predict the future price of a stock?; 2) How to incorporate the predicted price in MM?; 3) Will such a PMM method generate higher profit and higher market liquidity?; and 4) How does this PMM method compare with the baseline methods namely *RMM-Spooner* and *AS model*? To answer these questions, different PMM strategies (see Section 3.2 for detail), based on the type of deep neural network (DNN) architecture for price prediction and the value of consolidated price equation (CPE) weight ($w \in [0.5, 1]$), are designed. CPE uses the weighted mean of two asset prices, where the weight of each component is denoted by w and $1-w$. DNN architectures consist of multiple layers of artificial neurons connected fully with each other. Three different DNN architectures are used, namely multi-layer perceptron (MLP), long-short term memory (LSTM) and convolutional neural network (CNN) to design the price prediction model of PMM. The DNN prediction models use w to combine the current and the predicted price of a stock to generate a consolidated price. Moreover, the value of w controls the contribution of current and predicted prices in the consolidated price. The empirical observations say that if we increase the contribution of predicted price in CPE as the prediction accuracy increases then higher returns are observed.

The limit order book (LOB) simulated in this research uses intraday trading, where stocks are bought and sold the same day during market operational hours. LOB gets updated regularly at a fixed interval, typically five seconds, and a MM agent makes a decision based on the up-to-date information of the market. ETFs are nowadays popular with traders for a number of reasons, including appropriate diversification in portfolio, and the fact that they can be traded like regular stocks [7]. These PMM strategies are evaluated on a basket of ten stocks to find an optimal strategy for exchange traded funds (ETFs). These ten stocks and three ETFs are picked randomly from Yahoo finance.

The main contributions of this paper are: 1) the introduction of a novel concept of PMM, which enhances the returns of a RL-based MM agent and hence the market liquidity through LOB midprice prediction; 2) the design of a consolidated price equation (CPE), which is responsible for the amalgamation of present and future market prices of stocks; 3) a RL-based MM agent which can anticipate the future market trend in real-time.

The remainder of this paper is organised as follows. Section 2 reviews the related work. Section 3 provides necessary background details. Section 4 describes the proposed method. Then, Section 5 explains the empirical study and the statistical analysis of out-of-sample backtesting. Finally, Section 6 concludes the paper.

2 Related Work

MM has been studied in multiple fields including AI. [8] used RL to develop the first practical model of MM. They investigated the influence of uninformed market participants on quote placement behaviour of market makers and argued that RL policies converge successfully while keeping the balance between profit and spread. [3] approached the MM problem as an optimal bid and ask quotes placement in LOB using probabilistic knowledge where order arrivals and executions follow their model (AS model). [9] studied the impact of MM on price formation and treated exploration versus exploitation problem as price discovery versus profit earning. However, this work concentrated on price prediction and stability rather than enhancing the quality of market measured by market spread. [10] designed an automated MM framework based on convex hull optimisation. [11] used an online learning approach to design a market maker and empirically evaluated it while assuming sufficient liquidity. [12] designed an intelligent market maker which uses a prediction signal to improve profits but fails to suggest any successful MM strategy in general. [13] designed a high frequency MM model that takes advantage of speed in placing quotes and also argues that speed and profit are positively correlated. [14] proposed a trade execution model for high frequency MM that predicts sudden price fluctuation from price ticks (a tick is a smallest unit of change in price) information. However, they argue that recurrent neural network-based price flip prediction is advantageous in MM, but fail to test their model on other MM strategies including RL, which limits the practicality of their model. Then, [2] developed a state-of-the-art model (*RMM-Spooner*) based on RL which is the first attempt to study this problem on realistic grounds taking into consideration all market phenomenons that affect MM. [15] employs asset price momentum signals in decision making strategy of MM agent. This momentum signal-based decision making closely resembles this work. However, we minimise the risk of sudden or unexpected momentum shift (due to the irrational behavior of traders towards new information) using CPE, unlike [15]. Moreover, this work designs a more robust MM strategy which predicts an actual price rather than just the direction (upward or downward trend) like [15]. The similarity between this work and [15] is directional trading based on price changes.

As far as machine learning (ML) in finance and economics is concerned, [16] improved the results of a recurrent RL-based trading system using a genetic algorithm. [17] designed an automated trading system which combines DNN and RL together and predicts the number of shares to enhance the trading decisions. They evaluated their system on different stock indices including S&P500 and reported significant improvement in profits as compared to their baseline RL trading method. Research conducted by [17] improved financial trading decisions using deep q network. Many researchers applied machine learning-based forecasting techniques to enhance the current state-of-the-art methodologies used in economics and finance. [18] used extreme learning machine for bankruptcy prediction whereas [19] also used extreme learning machine but used it to design a financial soundness predictor using bank data. Moreover, [20] evaluated various ML

algorithms for money laundering detection. Another work published by [21] analyzed the role of market sentiment and technical indicators in conjunction with ML techniques to predict stock trends.

In financial time-series prediction, neural networks have been a popular choice among the AI community, particularly LSTMs. [22] studied stock market price prediction using backpropagation and multi-layer feed forward neural network. They stated that mathematical or statistical techniques are not appropriate as market indicators do not represent any significant relationships that make stock markets quite difficult to predict. [23] focused on MLP, CNN and LSTM in proposing a hybrid architecture using wavelets for stock price prediction. [24] used LSTM to predict stock market movement based on historical prices and technical indicators. They reported 55.9% accuracy in their prediction results and described them as promising. [25] used backpropagation neural networks (BPNN) and the improved sine cosine algorithm (ISCA) with Google Trends to predict opening price movement for the S&P 500 and DJI Indices, respectively. [26] used LSTM for price prediction with wavelet transform to remove noise. Recently, [27] found MLP outperforming vector autoregressive model in predicting crude oil prices. Based on these existing studies mentioned above, we chose MLP, CNN and LSTM neural network models to solve the time-series price prediction problem for use in the various PMM strategies.

3 Background

In this research, we integrate the *RMM-Spooner* model with market price prediction feature. This integrated machine learning-based model is known as PMM model, and is shown in Fig. 1. This section comprises three subsections; the first subsection describes about LOB simulation method. Then, the second subsection explains the MDP formulation of the PMM model, and finally the last subsection is about the proposed RL-based MM method.

3.1 LOB Model

A limit order (LO) arrives at a certain price and volume and has two sides, namely the ask and the bid. An LOB is a sorted list of LOs awaiting execution at their quoted or higher price. The LOB simulator designed here displays top five ask and bid orders at multiple levels, as shown in Fig. 2.

An order which executes immediately on arrival against the resting LOs in LOB starting with the best price, is known as a *market order* (MO). Each MO gets matched with the best available LO on opposite side. Simply, a buy MO is filled against a lowest ask LO, and viceversa. The difference between prices at multiple levels is denoted by the *tick size*. On the other side, a LO arrives and rests in a queue at a price value until an opposite MO arrives in the LOB. In case no MO arrives and an opposite LO already exists at the same price level, then both LOs get executed immediately.

Fig. 1 Flowchart of the PMM model, comprised of DNN-based price predictor and RL-based MM agent connected via CPE

LOBs are widely accepted and presumably almost half of the financial markets use them for maintaining arrival and execution of orders [28]. Furthermore, they provide remarkable market insight, hence we use a realistic data driven LOB simulator for the RL environment. The PMM agent places both ask and bid LOs together with some intended spread, known as $quoted_{spread}$ (denoted by qs in Eq. 1).

$$Return = qs \times v \times z \quad (1)$$

Fig. 2 LII order book snapshot containing price and volume information

LIMIT ORDER BOOK		
	Price	Shares
<div>Asks</div>	\$150.10	100
	\$150.05	100
	\$150.00	200
	\$149.96	100
	\$149.94	100
<div>Bids</div>	\$149.73	100
	\$149.57	100
	\$149.50	374
	\$149.49	100
	\$149.17	52

A trade is considered successful when both the ask side and the bid side of a LO are executed. The v denotes the number of units (e.g. number of shares of a stock) to be bought and sold in a trade. The z is the number of trades occurred between the PMM agent and other traders. The PMM agent makes a profit in every successful trade. In the case of only one side executing, inventory accumulation occurs. When only the ask side LO gets filled, the inventory decreases; otherwise it increases. There is a set inventory range (e.g. -100 to 100) beyond which all LOs are automatically cancelled to make the inventory zero. The nonzero inventory is cleared against MOs, otherwise they are cancelled at the end of the day. The *Return* is accumulated incrementally at each z number of units per equity traded.

3.2 MDP Formulation

Markov decision processes (MDPs) are the stochastic control processes, and [29] state that MDPs are the suitable candidates to model the sequential decision making problems including MM. [30] describe RL as the method of mapping situations to actions by assessing the scalar reward signal and can solve MDPs. RL is a type

of machine learning method which is neither supervised nor unsupervised. In other words, RL reinforces a learning agent to directly interact with its surrounding environment without any supervision. The RL agent observes the environment state and picks the best action to execute, then the environment returns back the scalar reward signal. This returned reward signal is used as a feedback by the agent to assess the quality of the picked action. Then, the agent again observes a new environment state and picks the best action and receives a reward. In this manner, the sequence of rewards is obtained in discrete time steps and the goal of RL agent is to maximize this sequence.

3.2.1 State Space

From the concepts of MDPs, a RL problem consists of an agent transiting one situation to another in discrete time steps. Mathematically, these situations are known as states and are either discrete or continuous in nature. The RL agent starts in some initial state, then transits from one state to another, and finally reaches some final state or terminal state. This transition from initial to terminal state, is known as an episode. The agent transits from one particular state to another particular state with certain probability. This probability is governed by the state transition function. In other words, the state transition function provides the connection among states means which state leads to which other state. This connection information of state space is known as the model of the RL environment. There are two types of RL environments, namely model-based and model free. In model-based, the state transition function is known in advance. However, in model free no transition probability information is available. In most practical scenarios the model of the environment is not known or difficult to obtain due to the unavailability of transitional dynamics information, and so the model free RL approach is widely used in solving real problems including MM. Moreover, when the states are only observable to the agent at the time of decision making then the RL algorithm solves a partially observable Markov decision process (POMDP) instead of a MDP.

Some technical indicators provide a useful insight and facilitates the LOB simulation for the PMM agent, namely:

- Volatility: the dispersion of returns of an equity
- Volume imbalance: the ratio of ask to bid volume in LOB
- Relative strength index: measures the change in price recently
- Market spread: the difference between lowest ask and the highest bid of LOB
- Mid-price movement: the change in the average value of lowest ask and highest bid prices in LOB

We use lookback window method to calculate volatility, relative strength index, market spread from historical time-series LOB data. The standard formulae for the calculation of each of above-mentioned indicators are used. The length of lookback windows is 60 days. The historical volatility value is computed via calculating the square root of the variance ($1/n(\sum_{i=1}^n (x_i - \mu_i)^2)$) of historical prices window or sim-

ply the standard deviation of historical prices. The relative strength index is computed using the formula $(\text{midprice price up} - \text{midprice price down}) / (\text{midprice price up} + \text{midprice price down})$. The midprice price up is the mean of historical mid price move (in upward direction) window, whereas the midprice down is the average of historical mid price move (in downward direction) window. The market spread is computed using the formula $(\text{lowest ask price} - \text{highest bid price})$. In our formulation, the state is continuous in nature, similar to *RMM-Spooner* and [31], and denoted by 8 variables, namely volatility, volume imbalance, relative strength index, market spread, mid-price move, inventory, ask distance, bid distance. The inventory denotes the volume of asset(s), either positive (long position) or negative (short position), of the PMM agent. The ask distance calculates the distance between the best open order in the ask book and the best ask price, whereas the bid distance is the difference between the best open order on the bid side and the best bid price.

3.2.2 Action Space

As defined in the section above, the RL agent transits between states within the state space. From the concepts of MDP and RL, these transitions are performed when the RL agent executes some action. Actions can be discrete or continuous in nature, like states. We design our action space similar to *RMM-Spooner*. Each action has two parts, namely ask-book level and bid-book level. The ask-book level specifies the level of LOB for ask quote and bid-book level specifies for bid quote. The action space contains 9 discrete actions, namely $Quote(1, 1)$, $Quote(2, 2)$, $Quote(3, 3)$, $Quote(1, 0)$, $Quote(0, 1)$, $Quote(2, 0)$, $Quote(0, 2)$, $Quote(3, 0)$, $clear_{inventory}$.

3.2.3 Reward

A reward is a scalar feedback signal for a specified action in a state of the environment and distinguishes RL from unsupervised learning where the goal is to extract hidden patterns in unlabeled data [30]. In other words, a reward function is a mathematical representation of the targeted goal of the RL agent. In the realm of MM, *RMM-Spooner* defined the reward as a function of asset returns and the RL agent's inventory. We use the similar reward function represented by Eq. 2.

$$Reward_t = \phi_t - (\lambda \times \max(inv_t \times \xi_t, 0)) \quad (2)$$

The components of the reward function are as follows: net profit/loss received is denoted by ϕ_t , inv_t is the inventory of the PMM agent (see Section 3.2.1 for detail), $\lambda \in [0, 1]$ is a parameter to control the influence of inventory on reward ($\lambda=0.7$ from *RMM-spooner*) and ξ_t is the LOB midprice at time t . The PMM agent trains itself to make optimal decisions through trial-error method, the post learning behaviour solely relies on the components of the reward function. The correct modelling of the reward function, depending on the problem, is crucial to achieve a desired behaviour of the RL agent. The intended goal of a PMM agent is to maximize the value of reward function via maximizing the asset return and minimizing the inventory. Moreover, the inventory risk can be controlled by a regulator λ (Eq. 2), while the

PMM agent learns to pick an optimal action that always yields higher return in long term with minimum risk in every state of the environment.

3.2.4 Value Function

The objective of a PMM agent is to discern the relationship between the state space and the action space through an evaluation of reward signals. This objective is achieved through deriving an optimal policy (a mapping from states to actions which leads to the desired behaviour of agent) which is either an optimal state-value function denoted by $V^*(s)$ or an optimal action-value function denoted by $Q^*(s, a)$. We prefer to use $Q^*(s, a)$ over $V^*(s)$ to find an optimal policy since $Q^*(s, a)$ provides a better estimate of the policy than $V^*(s)$ [32], because action-value function can distinguish the expected cumulative reward among different actions. However, the state-value function considers only state to provide an estimate of expected cumulative reward.

The state space is continuous in nature. This makes traditional RL methods such as tabular method for storing the learning experience impractical due to infinitely many state-action pairs. Therefore, a function approximation method is required to estimate the $Q^*(s, a)$ function. We use a linear parameteric function approximation method, namely tile-codings [33] for this purpose. $Q^*(s, a)$ estimates the quality of an action a , in terms of the expectation of total cumulative reward, when the PMM agent started in a state s and reaches the terminal state. A sequence of interactions from initial to the terminal state between the agent and its environment, is known as an episode. As mentioned earlier, the $Q^*(s, a)$ denotes the optimal policy of PMM agent which means using this optimal policy the agent selects an optimal action in an unseen state.

There are two types of RL policies, namely target policy and action-selection policy. The target policy is the policy which is being learned to estimate the action-value function. However, the action-selection policy is used to select actions while interaction with the environment. RL algorithms can be categorized into : 1) off-policy; and 2) on-policy. An off-policy method estimates a different target policy and follows another fixed action-selection policy, whereas an on-policy-based RL agent uses the target policy as the action-selection policy while learning. Off-policy method requires an action-selection policy in advance and is difficult to obtain in most real-world problems including MM. Therefore, we prefer to employ a well-known on-policy approach, namely SARSA (see Eq. 3), based on the pioneering temporal-difference (TD) learning algorithm [33].

$$Q_{new}(s_t, a_t) \leftarrow Q_{old}(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q_{old}(s_t, a_t)) \quad (3)$$

As mentioned earlier, the state space of the PMM agent is continuous, therefore a function approximator (FA) (e.g. linear function of features such as tile-codings, neural networks, Gaussian distribution-based FA [34], etc.) is required to generalize the learning experience across entire state space. SARSA uses state, action at time t along with the reward, state and action at time $t+1$ to estimate the TD-error term $(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q_{old}(s_t, a_t))$ in Eq. 3. The TD-error is computed by

bootstrapping the future Q value discounted with γ , and then subtracting the current Q estimate from the future estimate and adding the future reward r_{t+1} . The FA can either be a linear function or a neural network, FA represents the continuous state spaces in terms of the product of features and their weights. The PMM agent estimates the weights by minimizing the TD-error. These learned weights of FA represent the unbiased estimate of the true action-value function. Hence, the learned $Q^*(s, a)$ function is treated as an optimal RL policy, denoted by π^* .

4 The Proposed Method

We propose the PMM model-based MM agent which comprises two main components : 1) an RL-based MM agent; 2) DNN-based price predictor. The PMM model-based MM agent merges the two market prices, i.e. the current price and the predicted price together to estimate a final consolidated market price. The PMM model-based agent places LOs in the LOB, with certain price and volume. All LOs are placed in the LOB at each discrete point of time, during the trading hours. At an instant of time t , the PMM agent can access the market price (average of the best ask and best bid prices) from the LOB. This market price of an equity is considered as a reference point to calculate the simultaneous bid and ask quotes [3]. The PMM agent has the flexibility of merging the future market movement with the current market trend as opposed to the two benchmarks (*RMM-Spooner* and *AS model*). However, the PMM agent might not handle large volume orders (a limitation of market replay method) effectively, like *RMM-Spooner*.

The time-series market price prediction problem is solved using a sliding-window supervised learning method. Sliding-window is an array containing s number of historical market prices at time t . This array feeds the historical prices into the DNN-based price predictor (right-hand side of Eq. 4) to predict the equity price at the next time step denoted by ψ_{t+1} (left-hand side of Eq. 4).

$$\psi_{t+1} = \Gamma(\xi_t, \xi_{t-1}, \dots, \xi_{t-s}) \mid s \geq 1 \quad (4)$$

At time t , the historical LOB data provides the state space information, in terms of state variables, to the PMM agent. Then, the agent amalgamates the market prices of two discrete time steps, i.e. (ξ_t) and (ψ_{t+1}) , using CPE (Eq. 5) and computes the consolidated price denoted by Ψ_t (left-hand side of Eq. 5).

$$\Psi_t = (w \times \xi_t) + ((1 - w) \times \psi_{t+1}) \quad (5)$$

A linear function approximator, named tile-codings, is used to retain the learning experience. Then, the $Q(s, a)$ function values, from tile-codings, for all actions in the discrete action space are computed. An *epsilon-greedy* action selection policy is used to choose an action from action space using $Q(s, a)$ function values. The selected action (an ask and a bid quote is calculated using consolidated price Ψ_t) is then executed and a reward value is obtained using the reward function. The returned reward is then used to update the $Q(s, a)$ function using SARSA algorithm. In this manner, the PMM agent minimizes the sudden price fluctuation risk which

continuously persists in the markets. CPE makes PMM flexible enough by providing automatic quotes adjustment according to the future movement of market price. Moreover, there is a sudden market momentum reversal risk associated with the predicted price ψ_{t+1} due to the irrational behaviour of traders in the market. CPE weight acts as a risk controller for PMM agent, which is simply not present in *RMM-Spooner* and *AS model*.

As mentioned earlier, the feed-forward DNN models, responsible for market price prediction, uses sliding-window method for training on the time-series LOB historical data. These DNNs are built using TensorFlow framework and are trained on 80% of the data. In the MLP neural network architecture, each hidden layer contains 100 neurons with *rectified linear unit* activation function and *mean-squared-error* loss function. LSTM architecture also contains 100 neurons in each hidden layer with *adam* optimizer and *mean-squared-error* loss function. CNN has *convolutional layer* as input layer and *max-pooling* and *flattening* layers as hidden layers. Each *convolutional* layer contains filters each of size 2 with *rectified linear unit* activation. CNN uses *adam* optimizer function for optimizing the network. The hidden layers in MLP, LSTM and CNN are 2, 3 and 3, respectively.

The hyperparameters of the first component (a RL-based MM agent) of the PMM model are used from *RMM-Spooner* (1-11 in Table 1) and therefore are optimal. However, the second component (DNN model for price prediction) of PMM employs a random selection of the values of hyperparameters 12-14 in Table 1, and uses *hit and trial* to find the best combination for each DNN architecture.

The DNN models are trained using a well-known backpropagation method for propagating the absolute error (positive difference between the predicted and the actual value) and then adjusting the weight coefficients of input neurons. This training process includes trying different random values of the hyperparameters 12-14 shown in Table 1 in order to minimize the *root-mean-square-error(RMSE)* and to attain the best hyperparameters values for each of the DNN architectures. Multiple

Table 1 Optimal hyperparameters of the RL-based MM agents, including PMM

Serial No.	Configuration	Value
1	No. of state variables	8
2	No. of actions	9
3	RL algorithm	Sarsa
4	RL policy	ϵ -greedy
5	Training data (split %)	80
6	Testing data (split %)	20
7	RL training episodes	2000
8	RL testing episodes	20
9	Inventory control (λ)	0.7
10	Data frequency	5 secs
11	Quote size	10
12	Sliding window size	100
13	Prediction horizon	10
14	DNN training epochs	500

PMM strategies are developed based on the hyperparameters shown in Table 1 and two further variables : 1) CPE weight w ; 2) the type of DNN model (MLP, LSTM and CNN). We test numerous PMM strategies based on the combination of these further two hyperparameters (CPE weight w and type of a DNN architecture), such as $PMM_{(LSTM, w=0.90)}$, $PMM_{(CNN, w=0.95)}$ etc.

5 Experiments

Section 5.1 describes the LOB simulation datasets which are collected from an open source data service owned by CBOE. The dataset contains quotes and trades data, where quotes represent the unfilled limit orders resting in LOB and trades are the filled quotes. Section 5.2 analyses the empirical performance of various PMM strategies in terms of MM returns on individual stocks and ETFs.

5.1 Data

An LII (or L2) LOB keeps track of quotes placed by different market participants including MM agents. CBOE provides the LII book trades and quotes intraday high frequency (typically 5 seconds) tick-by-tick data which contains the top five asks and bids along with the recent trades during market operational hours. Each quote contains price and volume information at which it is awaiting to get filled. Moreover, price and volume information gets updated at the rate of every five seconds throughout the day. A computerized method randomly chooses ten options¹ from the top 100 listed on CBOE namely: Vodafone Group Plc (VOD), American Airlines Group Inc (AAL), GlaxoSmithKline Plc (GSK), Altria Group Inc (MO), Amazon Inc (AMZN), Walmart Inc (WMT), Nvidia Corporation (NVDA), Chevron Corporation (CVX), United Parcel Service (UPS) and Texas Instruments Incorporation (TXN), belonging to different sectors providing diversity like ETFs. The quotes and trades data of ten stocks and three ETFs (SPY, DIA and XLF) for ten months (1 Jan 2019 - 30 Sep 2019) from market opening to the closing time, i.e. 8.30 to 16.30 Monday to Friday, is gathered and preprocessed for experimentation.

5.2 Results and Discussion

As mentioned earlier, the hyperparameters used to develop each MM strategy, including both benchmarks, are same and shown in Table 1. The PMM strategies are the MM strategies developed through optimizing the CPE weight w (Eq. 5) for each of the DNN model (MLP, LSTM and CNN). We design an “ideal” PMM strategy, denoted by $PMM_{(perfect, w=0.5)}$, so as to compare against the best performing PMM methods. DNN-based price prediction models are trained on a historical data range (1 Jan 2019 - 30 Sep 2019), and the “ideal” method was aware of future prices while

¹ For simplicity we refer them stocks throughout.

Table 2 Out-of-sample backtesting averaged returns (average value of PnL returns of multiple RL agent's testing episodes) of the best PMM strategies and the benchmarks

Equity	$PMM_{(MLP,w=0.80)}$ (\$ strategy 1)	$PMM_{(LSTM,w=0.90)}$ (\$ strategy 2)	$PMM_{(CNN,w=0.95)}$ (\$ strategy 3)	$PMM_{(perfect,w=0.50)}$ (\$)	RMM-Spooner (\$)	AS model (\$)
VOD	6.6356	4.3670	10.3290	14.4481	9.7375	-4.3579
AAL	-2.0023	-1.9823	-2.4106	-0.3446	-1.9563	-4.3562
GSK	-6.4512	-10.7045	-5.9494	-8.8139	-14.2216	-4.3849
MO	4.2870	-3.8337	8.5449	6.7650	7.3401	-4.4449
AMZN	3992.554	3885.496	3951.784	4176.199	3828.689	0.0132
WMT	-7.8498	-0.9850	-7.8599	0.5000	-4.5429	-4.6585
NVDA	-4.3581	-1.9429	1.2900	1.9002	-1.6871	0.2147
CVX	-18.7016	-13.5129	-20.179	-2.1177	-14.2416	-4.3447
UPS	-2.6849	-3.6351	-1.1051	-0.955	-1.775	-4.4126
TXN	-16.4483	-12.575	-14.934	-5.5586	-15.7651	-3.1253

training, hence there is no error in prediction. The “ideal” PMM method knows the true future market price value in advance, hence we consider this method as a theoretical PMM benchmark for our empirical study. The “ideal” approach superseded every other MM strategy, including PMM, in terms of total number of positive returns (USD) in ten stocks (Tables 2 and 3). The contributions of both the current and the predicted market prices are kept equal ($w=0.5$ in Eq. 5) towards the consolidated price. From the empirical analysis, the CPE weight w and the RMSE of DNN model are linearly related with each other (Fig. 3). In simple words, the more accurate the prediction of the model the lesser the w – this suggests w depends on accuracy, rather than accuracy on w . The curve recommends using lower w with lower RMSE and higher w with higher RMSE.

The contribution of the predicted market price should decrease (by increasing the w) with the increase in RMSE of DNN models. When the RMSE > 0 , then $w \in (0.5, 1]$, and when the RMSE is 0, then $w=0.5$. Moreover, the existing practical MM benchmark, namely *RMM-Spooner*, uses $w=1$. Therefore, the range of w , used in developing and evaluating PMM strategies, is $(0.5, 1]$.

However, in practice no such DNN time-series prediction model exists which has RMSE exactly 0. In fact, in practice the price prediction component of PMM needs to predict the future price in real time, which would definitely be not accurate ($RMSE \neq 0$). For all three DNN models (MLP, LSTM and CNN) w values lie in the range $(0.5, 1]$, as they are real prediction models and do not have 0 RMSE. Therefore, we tried multiple arbitrary w values for each architecture, as shown in Table 3. Then, we select the best performing w values, as shown in Table 2, for all three architectures. Strategies 1, 2 and 3 are the best performing practical PMM strategies, with their respective w values depending on the RMSE values (Table 4) of their corresponding DNN architectures.

We optimize the hyperparameter w of PMM model developed using hyperparameters (Table 1) and design multiple PMM strategies, as shown in Tables 2 and 3. The

Table 3 Out-of-sample backtesting averaged returns (average value of PnL returns of multiple RL agent's testing episodes) of the PMM strategies, based on arbitrary CPE weight (w) values for each DNN architecture

Equity	$PMM_{(MLP,w=0.95)}$	$PMM_{(LSTM,w=0.95)}$	$PMM_{(CNN,w=0.80)}$	$PMM_{(MLP,w=0.60)}$	$PMM_{(LSTM,w=0.85)}$	$PMM_{(CNN,w=0.60)}$	Regression Tree
VOD	8.4615	7.6570	12.7810	-10.867	8.9669	-8.3338	-395.75
AAL	-0.5641	-0.9418	-0.9874	-2.3305	-0.9411	-7.1046	-6.911
GSK	-8.5705	-5.3603	-3.6554	-9.5500	-2.3104	-8.7727	-18.099
MO	6.4121	4.1685	5.8150	-5.1916	3.8941	2.5209	-20.2199
AMZN	3688.178	3760.843	3486.591	4229.078	3602.86	3509.765	-29708.3
WMT	-1.7851	-2.9199	-5.0586	-9.61	-2.8599	-3.8955	-1119.35
NVDA	-0.7336	-1.7701	-60.1738	-38.689	-5.5307	-92.8977	-3025.4044
CVX	-6.3604	-4.8210	-54.7627	-80.4055	-15.8406	-105.418	-19785.5
UPS	-1.8050	-1.9799	-0.8199	-21.5576	-2.4201	-36.0512	-51.0280
TXN	-14.886	-15.9551	-10.8401	-38.0975	-20.6905	-78.5033	-2293.17

Fig. 3 Linear relationship between RMSE and CPE weight (w)

curves shown in Fig. 4 look noisy, and therefore a direct conclusion is difficult to be drawn.

Hence, we use averaged return value of each equity of all strategies. Table 2 shows the average return over number of testing episodes, means in twenty episodes twenty return values are obtained and the average of these twenty values is the average return value of an equity. Using average return values helps in reducing noise and a significant statistical conclusion can be easily drawn. The experiments behind these results are repeated five times for each of the MM strategy with optimal hyperparameter settings, and observed that the same curves are obtained in every repetition. No uncertainty is observed during experiments repetition; hence the results have not originated from any statistical anomaly. Moreover, the confidence intervals can be computed only when the results vary in each repetition.

The out-of-sample backtesting returns (USD) of stocks are averaged over the number of backtesting episodes and represented in Tables 2 and 3. Table 2 shows the optimized

Table 4 RMSE of DNN models and market spread of stocks

Equity	RMSE (MLP)	RMSE (LSTM)	RMSE (CNN)	Market Spread (\$)
VOD	0.0113	0.0221	0.0134	0.017
AAL	0.0121	0.0124	0.0163	0.011
GSK	0.0082	0.0081	0.0092	0.060
MO	0.1638	0.4812	0.4873	0.009
AMZN	0.3118	0.3504	0.4941	1.447
WMT	0.0182	0.0270	0.0255	0.031
NVDA	1.2522	1.2446	1.2465	0.156
CVX	0.0266	0.0462	0.0433	0.029
UPS	0.0272	0.0437	0.0396	0.052
TXN	0.0301	0.0358	0.0437	0.044
Mean :	0.1861	0.2271	0.2418

Fig. 4 The out-of-sample backtesting investment returns of different RL-based MM strategies for twenty episodes in ten equities. Strategy 3 beats the *RMM-Spooner* in 3(a) despite the unclear pattern. *RMM-Spooner* model yields lesser loss than all PMM strategies in 3(b). Strategy 3 is more profitable than the *RMM-Spooner* in 3(c), 3(d), 3(e), 3(f), 3(g), 3(i) and 3(j). Strategy2 is better than considering the *RMM-Spooner* in 3(h)

PMM strategies (Fig. 4), and Table 3 shows how returns (USD) differ with w and the type of DNN model architecture. Out of these optimized PMM strategies (strategy 1, 2 and 3), we select one best out of these three based on the number of stocks yielding positive returns in the basket of ten stocks. If we carefully analyse the Table 2, then we observe that after $PMM_{(perfect, w=0.5)}$ (an “ideal” PMM method or a theoretical PMM benchmark) strategy 3 outperforms all others including benchmarks (*RMM-Spooner* and *AS model*). Strategy 3 obtains positive returns (USD) in 4/10 stocks, whereas

Table 5 Out-of-sample backtesting of Strategy 3, *RMM-Spooner* and *AS model* on ETFs

Symbol	Strategy 3 (\$)	RMM-Spooner (\$)	AS model (\$)	Market spread (\$)	RMSE (CNN)
SPY	25.8445	20.7483	-4.5587	0.0193	0.2786
DIA	62.1945	40.4736	-4.4901	0.0765	0.0513
XLF	4.1764	-0.4012	-3.5596	0.0137	0.0202
Mean (\$ per 30 units):	30.7384	20.2735	-4.2028		

RMM-Spooner gets 3/10 stocks with positive return (USD) values. We observe that in case of AMZN stock, the returns (USD) are highest in every MM strategy as the AMZN has highest market spread (Table 4). Moreover, another interesting observation includes the mean of absolute differences between stock returns (USD) among strategy 3 and *RMM-Spooner* is **\$19.66** in cases where strategy 3 outperforms *RMM-Spooner* and **\$3.23** in cases where *RMM-Spooner* wins. We use a simple regression tree-based model for price prediction and conducted out-of-sample backtesting (see Table 3). Empirical results conclude that the best PMM strategies outperforms “Regression Tree”-based MM strategy as well.

Until now, strategy 3 or $PMM_{(CNN, w=0.95)}$ is a best practical PMM strategy amongst all. The MM agents used to be large firms or banks which trades in large collection of stocks rather than individual stocks. Moreover, a clear observation of Fig. 4 says that no single PMM strategy can outperform the benchmarks in all individual stocks. Hence, we evaluate our identified PMM strategy in a diverse and large collection of stocks, so that it can be applied in a real market. ETFs, as mentioned earlier, are popular among traders as they provide a large and diversified collection of stocks and they can be traded as regular stocks. Based on these advantages of ETFs, strategy 3 is tested for 3 different ETFs, namely SPY (SPDR S&P 500 ETF, one of the largest ETF in the world), DIA (SPDR Dow Jones Industrial Average ETF) and XLF (Financial Select Sector SPDR ETF). Table 5 contains average values of returns (USD) obtained from out-of-sample backtesting of strategy 3, *RMM-Spooner* and *AS model*.

The returns (USD) obtained from using strategy 3 in all 3 ETFs are significantly higher than the *RMM-Spooner* (Fig 5).

Fig. 5 Out-of-sample backtesting of strategy 3, *RMM-Spooner* and *AS model* over ETFs

Table 6 Q-Spread (quoted spread of MM agent) and Trades (number of successfully executed orders) collectively represent the market liquidity provided

Equity	Strategy 3		RMM-Spooner		AS model	
	qs (\$)	z	qs (\$)	z	qs (\$)	z
Stocks	0.04	≈ 98	0.08	≈ 48	0.2	≈ 1
ETFs	0.002	≈ 512	0.006	≈ 112	0.008	≈ 18

The overall aim of any MM agent is to place the large number of orders with tight qs in order to enhance the market liquidity. The general rule says that the market liquidity is inversely proportional to the qs and directly proportional to the z . Clearly, the average value of qs and z of strategy 3, as shown in Table 6, is significantly higher than the two benchmarks, both in ETFs and individual stocks category. The final outcome of this study suggests that PMM with CNN and $w=0.95$ is a practical PMM approach and can be applied in a large stock exchange for MM in a large collection of stocks including ETFs.

6 Conclusion

Market making plays a vital role in preserving the market liquidity and the interest of participants in compensation of minute profit at every successful trade. Almost all major stock exchanges employ MM agents to boost up the market liquidity and induce smoothness in trades execution. Therefore, it becomes necessary to develop MM methods that can enhance their (MM agents) profits and bolster the market liquidity further. This paper proposes a novel concept, known as PMM, which aims to improve the investment returns of a RL-based MM agent by integrating the DNN-based market price prediction feature. The proposed PMM method was evaluated in individual stocks and a large collection of stocks, known as ETF. Empirical analysis of out-of-sample backtesting suggests that $PMM_{(CNN, w=0.95)}$ or strategy 3 is a practical MM approach and can trade in large collection of stocks, profitably. The research concludes that the PMM agent is intelligent due to the market anticipation feature and can provide more liquidity to the market as compared to the benchmarks (*RMM-Spooner* and *AS model*).

Funding Information This study is a part of a PhD project funded by VCRS (E4G3EN3G39F0A4). This work is a part of a PhD project funded by VCRS.

Availability of Data and Material Data source [CBOE](#).

Code Availability The project is ongoing; hence, code will be available after completion.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. O'Hara M (2014) High frequency market microstructure. *J Fin Econ*
2. Spooner T, Fearnley J, Savani R, Koukorinis A (2018) Market making via reinforcement learning. In: *Proceeding AAMAS '18 Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, pp 434–442
3. Avellaneda M, Stoikov S (2008) High-frequency trading in a limit order book. *Quant Fin* 8(3):217–224
4. Granados N, Gupta A, Kauffman RJ (2010) Research commentary: Information transparency in business-to-consumer markets: Concepts, framework, and research agenda. *Inf Syst Res* 21(2):207–226
5. Feng F, Chen H, He X, Ding J, Sun M, Chua TS (2019) Enhancing stock movement prediction with adversarial training. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, pp 5843–5849
6. Ceffer A, Levendovszky J, Fogarasi N (2019) Applying independent component analysis and predictive systems for algorithmic trading. *Comput Econ* 54:281–303
7. Deville L (2008) *Handbook of Financial Engineering*, vol 18, Springer, Boston, MA, chap Exchange Traded Funds: History, Trading, and Research
8. Chan NT, Shelton C (2001) An electronic market-maker. Tech. rep, MIT
9. Das S (2008) The effects of market-making on price dynamics. In: *Proceeding AAMAS '08 Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, International Foundation for Autonomous Agents and Multiagent Systems, vol 2, pp 887–894
10. Abernethy J, Chen Y, Vaughan JW (2011) An optimization-based framework for automated market-making. In: *EC '11: Proceedings of the 12th ACM conference on Electronic commerce*, p 297–306
11. Abernethy J, Kale S (2013) Adaptive market making via online learning. In: *Proceeding NIPS' 13 Proceedings of the 26th International Conference on Neural Information Processing Systems*, Curran Associates Inc., vol 2, pp 2058–2066
12. Li X, Deng X, Zhu S, Wang F, Xie H (2014) An intelligent market making strategy in algorithmic trading. *Front Comp Sci* 8:596–608
13. Ait-Sahalia Y, Saglam M (2017) High frequency market making: Optimal quoting. Available at SSRN: <http://dx.doi.org/10.2139/ssrn.2331613>
14. Dixon M (2017) High frequency market making with machine learning. <http://www.smallake.kr/wp-content/uploads/2017/10/1710.03870.pdf>
15. Álvaro Cartea, Wang Y (2019) Market making with alpha signals. <http://dx.doi.org/10.2139/ssrn.3439440>
16. Zhang J, Maringer D (2016) Using a genetic algorithm to improve recurrent reinforcement learning for equity trading. *Comput Econ* 47:551–567
17. Jeong G, Kim HY (2019) Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Syst Appl* 117:125–138
18. Zhao D, Huang C, Wei Y, Yu F, Wang M, Chen H (2017) An effective computational model for bankruptcy prediction using kernel extreme learning machine approach. *Comput Econ* 49:325–341
19. Fernández-Arias D, López-Martín M, Montero-Romero T, Martínez-Estudillo F, Fernández-Navarro F (2018) Financial soundness prediction using a multi-classification model: Evidence from current financial crisis in oecd banks. *Comput Econ* 52:275–297

20. Zhang Y, Trubey P (2019) Machine learning and sampling scheme: An empirical study of money laundering detection. *Comput Econ* 54:1043–1063
21. Ahmed M, Sriram A, Singh S (2019) Short term firm-specific stock forecasting with bdi framework. *Comput Econ*
22. Khan ZH, Alin TS, Hussain MA (2011) Price prediction of share market using artificial neural network (ann). *Int J Comp Appl* 22(2):975–8887
23. Persio LD, Honchar O (2016) Artificial neural networks architectures for stock price prediction: comparisons and applications. *Int J Circ Syst Signal Proc* 10
24. Nelson DMQ, Pereira ACM, de Oliveira RA (2017) Stock market's price movement prediction with lstm neural networks. In: *International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp 1419–1426
25. Hu H, Tang L, Zhang S, Wang H (2018) Predicting the direction of stock markets using optimized neural networks with google trends. *Neurocomputing* 285:188–195
26. Liang X, Ge Z, Sun L, He M, Chen H (2019) Lstm with wavelet transform based data preprocessing for stock price prediction. *Math Probl Eng*
27. Ramyar S, Kianfar F (2019) Forecasting crude oil prices: A comparison between artificial neural networks and vector autoregressive models. *Comput Econ* 53:743–761
28. Gould MD, Porter MA, Williams S, McDonald M, Fenn DJ, Howison SD (2013) Limit order books. *Quantitative Finance* 13(11):1709–1742
29. Otterlo M, Wiering M (2012) *Adaptation, Learning, and Optimization*, vol 12, Springer, Berlin, Heidelberg, chap Reinforcement Learning and Markov Decision Processes
30. Sutton R, Barto A (2018) *Reinforcement Learning: An Introduction*, 2nd edn. MIT Press
31. Chakraborty T, Kearns M (2011) Market making and mean reversion. In: *EC '11 Proceedings of the 12th ACM conference on Electronic commerce*, ACM New York, pp 307–314
32. Busoniu L, Babuska R, Schutter BD, Ernst D (2010) *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press Inc
33. Sutton RS, Barto A (1998) *Reinforcement Learning: An Introduction*, 1st edn. MIT Press, Cambridge, MA
34. Haider A, Hawe G, Wang H, Scotney B (2021) Gaussian based non-linear function approximation for reinforcement learning. *SN Comp Sci* 2(223)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.