Reinforcement Learning for Algorithmic Trading: A Review of Algorithms and Techniques

Shijie Shao
School of Data Science
The Chinese University of Hong Kong, Shenzhen
shijieshao@link.cuhk.edu.cn

Abstract

In recent years, considerable efforts have been devoted to developing algorithmic trading strategies. One popular method is to train intelligent trading agent through reinforcement learning (RL). In this review, we mainly talked about the applications of deep reinforcement learning methods and corresponding techniques in algorithmic trading. We first formalize the financial trading problem into the Markov Decision Process setting with constraints. Then we presented RL methods and their applications including value-based, policy-based, and actor-critic methods. Finally, we summarize the techniques for RL in trading including overfitting detection, auto adaption for financial markets, and ensemble methods.

1 Introduction

Traditional approaches for modelling financial decision problems are techniques like stochastic control. With the rapid development of reinforcement learning (RL), recent studies focus on the combination of RL methods and financial trading. Reinforcement learning, based on its property of learning from rewards, provides a new paradigm to financial trading and outperforms traditional trading strategies in certain areas. In this review, we summarized different RL-based methods and techniques that are used in financial trading and investigated the effectiveness.

1.1 Algorithmic Trading

Algorithmic trading is a process for executing orders on financial assets to maximize the net value at the end of a trading period. Nowadays it is considered as quite profitable and develops quickly around the world. In 2020, quantitative trading accounts for over 70% and 40% trading volume in developed market (e.g., U.S.) and emerging market (e.g., China), respectively [17].

1.2 Motivation of Applying RL on Algorithmic Trading

The overall objective of algorithmic trading is to maximize long-term profit , which shares similarity with the goal of reinforcement learning (i.e., maximizing cumulative rewards over time). Therefore, reinforcement learning is applied to design profitable trading strategies by training a trading agent [7]. After the training process, the trading agent makes profit through buying and selling financial assets based on provided market information.

1.3 Advantages of RL in Algorithmic Trading

- RL allows training an end-to-end agent, which combines the mixture problem of price prediction and time selection into a single policy searching problem [17].
- RL-based methods can easily convert task-specific constraints like transaction cost into the environment setting [7, 20].

1.4 Challenges of RL in Algorithmic Trading

Traditional reinforcement learning tasks are from fields like robotics and games. For example, Atari is a popular benchmark environment for reinforcement learning algorithms, which involves training a learning agent to play classic Atari games. Different from such well-defined tasks, algorithmic trading is much more difficult due to the following challenges:

- **Data instability:** The financial time series is usually unstable and non-stationary, containing a lot of noise, jumping and moving, which makes it difficult to apply gradient-based learning algorithms [7, 12]. The assumption that the context variables are Markovian fails [4] .
- **Data complexity:** It is difficult to summarize and represent the highly complex financial environment, especially the high-frequency financial data [12].
- **Exploration and exploitation dilemma:** The existence of transaction costs hinder the trading agent from exploration, and thus a low-level of exploration in feasible in financial trading setting [7].

2 Background

2.1 Markov Decision Process Formalization

Usually, an algorithmic trading process can be modeled into a discrete MDP with finite time horizon $(t = 0, 1, \cdots, T)$. The ultimate goal is to obtain the trading strategy (i.e., policy $\pi(s)$) that maximizes the cumulative return $\sum_{t=0}^{T-1} r(s_t, a_t, s_{t+1})$.

The vanilla formalization of MDP is based on the Zero Market Impact hypothesis, which assumes market participators have no impact on the current market condition [7]. This MDP takes price as the only financial information and combines it with the holding positions as the state space [9, 11].

- State $s = [p, h, b]$: a set that includes the information of the prices of assets $p \in \mathbb{R}_+^D$, the amount of holdings of assets $h \in \mathbb{Z}_+^D$, and the remaining balance $b \in \mathbb{R}_+$, where D is the number of assets that we consider in the market.
- Action a: a set of actions on all D assets. The available actions of each asset include selling, holding, and buying:
  - Selling: $a_t[d] = -k \in [-h_t(d), -1]$, and $h_{t+1}(d) = h_t(d) - k$.
  - Holding: $a_t[d] = 0$, and $h_{t+1}(d) = h_t(d)$.
  - Buying: $a_t[d] = k > 0$, and $h_{t+1}(d) = h_t(d) + k$.
- Reward $r(s, a, s')$: the change of the portfolio value v when action a is taken at state s and arriving at the new state s'. The portfolio value $v_t = \boldsymbol{p_t^T h_t} + b_t$ is the sum of the equities in all held assets and remaining balance.

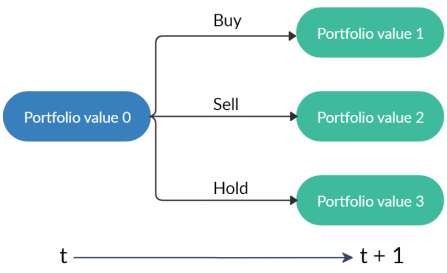

Figure 1: Illustration of actions [21]

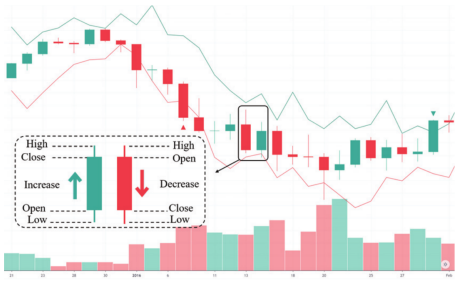

Figure 2: Illustration of price movement [12]

However, in reality, the financial market is much more complicated and risk management is usually considered in financial trading. Therefore, other enhanced MDP formalizations adds market features into the state space [7, 21] and adopts more well-performed reward functions [7, 18, 20], but usually leaving action space unchanged.

- State $s = [p, h, b, M]$: When M represents the market features. Usually it consists of some technical factors reflecting price trend and volume change, like Moving Average Convergence Divergence (MACD) and Relative Strength Index (RSI).
- Reward $r(s, a, s')$: Advanced reward functions may consider other measures. In the work **DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding [19]**, the authors adopt reward functions including (i) profit criterion like return (ii) risk criterion like annualized volatility and maximum drawdown (iii) risk-profit criterion like Sharpe ratio and Calmar ratio. And their experiments show that the maximum drawdown reaches a significantly better control of risk without losing much return, which is considered as the best.

2.2 Importing Financial Trading Constraints

- **Transaction cost:** Each trade contains a part of transaction cost, and different brokers acquire varying commission fees. For example, in cryptocurrency trading, the transaction cost is often assumed as 0.3% of the value of each trade [5]. Therefore, the reward function based on net profit becomes

$$r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) = \left(b_{t+1} + \boldsymbol{p}_{\boldsymbol{t+1}}^\top \boldsymbol{h}_{\boldsymbol{t+1}}\right) - \left(b_t + \boldsymbol{p}_{\boldsymbol{t}}^\top \boldsymbol{h}_{\boldsymbol{t}}\right) - c_t,$$

  and the transactions cost c_t is

$$c_t = \boldsymbol{p}_t^\top |\boldsymbol{a}_t| \times 0.3\%$$

  where $|\boldsymbol{a}_t|$ means taking entry-wise absolute value of $\boldsymbol{a}_t$ [5].
- **Restriction on short:** In many financial trading setting, short selling is restricted due to legal regulations and its high cost. And this restriction is easy to be embedded in RL environments. Suppose We do not allow short, then we make sure that $b_{t+1} \in \mathbb{R}_+$ is non-negative,

$$b_{t+1} = b_t + \boldsymbol{p}_t^\top \boldsymbol{a}_t^S + \boldsymbol{p}_t^\top \boldsymbol{a}_t^B \geq 0, \text{ for } t = 0, \ldots, T-1$$

  where $\boldsymbol{a}_t^S \in \mathbb{R}_-^D$ and $\boldsymbol{a}_t^B \in \mathbb{R}_+^D$ denote the selling orders and buying orders, respectively, such that $\boldsymbol{a}_t = \boldsymbol{a}_t^S + \boldsymbol{a}_t^B$. Therefore, action $\boldsymbol{a}_t$ is executed as follows: first execute the selling orders $\boldsymbol{a}_t^S \in \mathbb{R}_-^D$ and then the buying orders $\boldsymbol{a}_t^B \in \mathbb{R}_+^D$; and if there is not enough cash, a buying order will not be executed [5].

3 Algorithms

Model-free deep reinforcement learning algorithms are divided into 3 categories: Value-Based, Policy-Based, and Actor-Critic [15]. Value-Based methods aim to learn the optimal action-value function for a given state; Policy-Based methods directly learn the optimal policy that maps a state to an action; Actor-Critic methods combine value-based and policy-based methods by learning both an action-value function and a policy. In this section, we briefly introduce a DRL algorithm for each category and illustrate how those methods are applied in financial trading domain.

3.1 Value-Based RL

Deep Q-learning Networks (DQN) [14] A DQN approximates the action value function (Q function) to estimates how good it is for the agent to perform a given action in a given state by adopting a neural network [23]. Suppose our Q function is parameterised by some θ, which can be updated through backpropagation. The goal is to minimize the MSE between the current and target Q functions to derive the optimal state-action value function:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')}\left[\left(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)\right)^2\right]$$

One related work is **Financial Trading as a Game: A Deep Reinforcement Learning Approach** [7]. This work constructs the state space with time features, market features, and position features and uses the log return of portfolio value as the reward function. The author adopts Deep Recurrent Q-Network (DRQN), which means using LSTM as the Q-network to better process sequence data.

Most importantly, to mitigate the need for random exploration, the authors propose a novel vector-formed loss function called action augmentation loss [7]. Action augmentation loss is implemented

by offering additional feedback signals to the agent for all actions taken. And this modification gains an additional 6.4% annual return in average compared with ϵ-greedy method.

$$\tilde{L}(\theta) = \mathbb{E}_{(s,\boldsymbol{a},\boldsymbol{r},\boldsymbol{s}')}\left[\left\|\boldsymbol{r} + \gamma Q_{\theta^-}\left(\boldsymbol{s}', \arg\max_{a'} Q_\theta\left(\boldsymbol{s}', a'\right)\right) - Q_\theta(s, \boldsymbol{a})\right\|^2\right]$$

There are several other need-to-mention modifications of training process in this work, which are helpful for value-based RL methods in trading:

- **Smaller replay buffer:** Different from the large replay buffers (N > 10^6) commonly required for value-based deep RL, they find adopting a much smaller (N = 480) replay buffer is more helpful. They attribute it to the "short-termism" property of financial time series, which means recent data carries more weight than past data in financial time series.
- **Longer sampling sequence length:** Motivated by the fact that a profitable trading strategy often requires opening a position at an optimal time and maintaining it for a sufficiently long period before closing it, the authors draw longer sequences (T = 96) from the replay buffer compared to the original DRQN paper [6].

3.2 Policy-Based RL

Trust Region Policy Optimization (TRPO) [16] TRPO aims to improve the stability and convergence speed of policy optimization. It adopts a trust region approach, which constrains the maximum KL-divergence between the updated policy and the previous policy. This constraint ensures that the updated policy only makes small changes to the policy distribution, which in turn helps to improve stability. Its objective optimization problem is shown as below:

$$\max_\pi L(\pi) = \mathbb{E}_{\pi_{\text{old}}}\left[\frac{\pi(a \mid s)}{\pi_{\text{old}}(a \mid s)} A^{\pi_{\text{old}}}(s, a)\right]$$
$$\text{s.t. } \mathbb{E}_{\pi_{\text{old}}}\left[d_{\text{KL}}\left(\pi \| \pi_{\text{old}}\right)\right] \leq \epsilon$$

One related work is **Risk-Averse Trust Region Optimization for Reward-Volatility Reduction** [2]. The authors highlighted the issue of controlling the risk in the intermediate steps of trading. Therefore, they developed a novel measure of risk to address this issue, which is called "reward volatility". This measure is based on the variance of rewards under the state-occupancy measure and is shown to bound the return variance. It combines normalized expected return J_π and reward volatility ν_π^2 to get the objective function $\eta_\pi = J_\pi - \lambda\nu_\pi^2$, where J_π and ν_π^2 are defined as:

$$J_\pi = (1-\gamma) \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}\left(s_t, a_t\right)\right] = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[\mathcal{R}(s, a)]$$

$$\nu_\pi^2 = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[\left(\mathcal{R}(s, a) - J_\pi\right)^2\right] = (1-\gamma) \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}\left[\sum_{t=0}^{\infty} \gamma^t \left(\mathcal{R}\left(s_t, a_t\right) - J_\pi\right)^2\right]$$

The authors prove that the performance of attained η for the policy $\tilde{\pi}$ is lower bounded. They adapt TRPO with the risk averse measure and proposed their Trust Region Volatility Optimization (TRVO) method, which exploits a monotonic improvement bound of the objective. Finally, the author applied TRVO on equity index S&P 500 and Foreign Exchange (FX). In Figure 3, the authors find TRVO outperforms the baseline models by achieving better Pareto frontiers in short time.

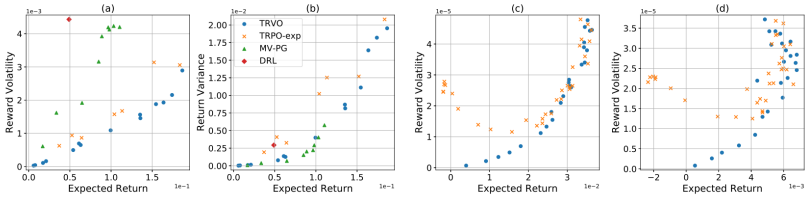

Figure 3: training (a) and testing (b) on S&P 500; training (c) and testing (d) on FX [2]

3.3 Actor-Critic RL

Deep Deterministic Policy Gradient (DDPG) [10] DDPG is improved from Deterministic Policy Gradient (DPG). DDPG directly learns from observations through policy gradient. It is designed to deterministically map states to actions to better adapt to the continuous action space environment, so it is appropriate for financial trading. DDPG is updated by minimizing the following MSBE loss $L(\phi)$ and maximizing $Q_\phi(s,a)$:

$$L(\phi) = \underset{(s,a,r,s',d)}{\mathrm{E}} \left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d) Q_{\phi_{\text{targ}}} \left(s', \mu_{\theta_{\text{targ}}}(s') \right) \right) \right)^2 \right]$$

$$\max_\theta \underset{s}{\mathrm{E}} \left[Q_\phi \left(s, \mu_\theta(s) \right) \right]$$

One related work is **Optimistic Bull or Pessimistic Bear: Adaptive Deep Reinforcement Learning for Stock Portfolio Allocation [9]**. In this paper, the authors proposed Adaptive DDPG for algorithmic trading, which is shown in Figure 4. The temporal difference (TD) error, given by $\delta(t) = r(s_t, a_t, s_{t+1}) - Q_\pi(s_t, a_t)$ is considered as environment emotional news in this work. They make intuitive adjustments to the actor network part by adjusting the learning rate α^+ and α^- corresponding to positive and negative environment emotional news.

$$Q_\pi(s_{t+1}, a_{t+1}) = Q_\pi(s_t, a_t) + \begin{cases} \alpha^+ \delta(t), \text{if } \delta(t) > 0 \\ \alpha^- \delta(t), \text{if } \delta(t) < 0 \end{cases}$$

Adaptive DDPG adopts both optimistic and pessimistic DRL by modifying the Q-learning algorithm by RW$\pm$ model [8], Therefore, when TD error is positive, which implies that the actual reward $r(s_t, a_t, s_{t+1})$ is better than the expected reward $Q_\pi(s_t, a_t)$, the amplitude of α^+ can determine the step from one trial to the next; reversely, the amplitude of α^+ will be adjusted based on the signs of sequential TD errors.

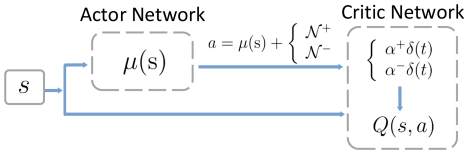

Figure 4: Adaptive DDPG architecture [9]

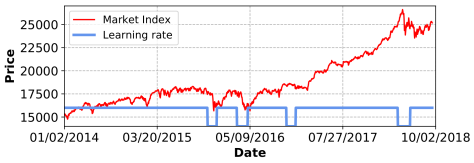

Figure 5: The market index during the testing data and the corresponding learning rate [9]

As shown in Figure 5, this dynamic adjustment of learning rate based on sequential TD errors is helpful in different market conditions. When the trend effect is strong, the actor network will be updated with a high learning rate; when the fluctuation effect is strong, the actor network turns reduces its learning rate to reduce its exposure to the high-noise period. To illustrate the effect of Adaptive DDPG, the authors select Dow Jones 30 component stocks as trading stocks and trading them with different DRL methods. As shown in Table 1, the Adaptive DDPG has a much better performance than other strategies. Adaptive DDPG creates a significantly higher return with lower risk. The auto adaption of learning rate is a simple but useful design.

Method	**Adaptive DDPG**	DDPG	DJIA	Min-variance	Mean-variance
Initial value	**10,000**	10,000	10,000	10,000	10,000
Final value	**21,880**	18,156	16,089	16,333	19,632
Annualized return	**18.84%**	14.71%	11.36%	11.48%	15.86%
Annualized Std. error	**11.59%**	14.68%	12.43%	11.64%	12.70%
Sharpe ratio	**1.63**	1.01	0.91	0.99	1.25

Table 1: Trading performance of Adaptive DDPG [9]

4 Techniques

4.1 Overfitting Detection

Overfitting is a common issue for machine learning in the training periods of financial trading. [1]. This problem hinders the trained RL agents from performing continuously well with time.

One related work for addressing overfitting is **Deep Reinforcement Learning for Cryptocurrency Trading: Practical Approach to Address Backtest Overfitting** [5]. To overcome this issue, they raise a detection technique for over-fitted agents. Their detection procedure of overfitting contains three steps: (i) implement random split of training-validation data and get multiple groups (ii) set a set of hyperparameters for tuning (iii) evaluate the agent's performance on all validation set and get average performance for length T' (smaller than T because of split). They repeat the detection H times and get a matrix $M \in \mathbb{R}^{T' \times H}$. They randomly split the M across rows into four sub-matrices $\boldsymbol{M^1}, \boldsymbol{M}^2, \boldsymbol{M}^3, \boldsymbol{M}^4 \in \mathbb{R}^{T'/4 \times H}$, where each sub-matrices represent the average performance over length $T'/4$. Setting two of them as the in-sample (IS) set and two of them as the out-of-sample (OOS) set,

$$c = \begin{bmatrix} M^1 \\ M^2 \end{bmatrix}, \text{IS set} ; \quad \bar{c} = \begin{bmatrix} M^3 \\ M^4 \end{bmatrix}, \text{OOS set}$$

the initial paper investigate backtest overfitting [1] by comparing whether the best performance in IS set is better than the median performance in OOS set. The authors in this work develops a more quantitative method. Define ϵ as the index of the bestperforming IS strategy. Then, they check the corresponding OOS rank $\overline{\boldsymbol{r}}^c[\epsilon]$ and define a relative rank ω^c:

$$\omega^c = \frac{\overline{\boldsymbol{r}}^c[\epsilon]}{H+1}, \text{where } c \text{ is a possible split}$$

Then the corresponding logit function is defined as:

$$\lambda^c = \ln \frac{\omega^c}{1-\omega^c}, \text{where } c \text{ is a possible split}$$

The authors integrate over the region where the best strategy IS has an expected ranking lower than the OOS set,

$$p = \int_{-\infty}^{0} f(\lambda) d\lambda$$

and get the probability function to characterize the probability p of overfitting. The trained agent is assumed to be overfitted if $p < \alpha$.

Finally, the authors adopt 3 DRL algorithms (PPO, TD3, and SAC) and test their performance on trading over 10 cryptocurrencies. Setting a threshold of $\alpha = 10\%$, they tested the probability of overfitting over the three agents, where the probabilities of overfitting are: $p(PPO) = 8.0\% < \alpha$, $p(TD3) = 9.6\% < \alpha$, and $p(SAC) = 21.3\% > \alpha$ (rejected). In Figure 6, we can find PPO (cumulative return $> 24\%$) is significantly better than other trading agents. The lower volatility of PPO implies its better robustness toward risk under different market conditions.

Figure 6: The average trading cumulative return curves for DRL algorithms [5]

4.2 Auto Adaption for Financial Markets

In financial trading area, modern investment strategies rely on investigating historical patterns that can be quantified in a systematic way, which is called backtesing [1]. Due to the structure breaks in financial markets, the RL algorithms suffer from non-stationarity, which significantly increases the difficulty for optimizing the algorithms [4]. Researchers designed many adaptive methods to fit the non-stationary time series like segmentation based on different patterns across segments of series [4].

One related work **Optimistic Bull or Pessimistic Bear: Adaptive Deep Reinforcement Learning for Stock Portfolio Allocation [9]** is an article we have already mentioned in the Section 3.2 . This work adopts dynamic learning rate based on optimistic and pessimistic market condition and improves the performance of DDPG in stock trading [9].

Another related work is **Adaptive Quantitative Trading: An Imitative Deep Reinforcement Learning Approach [12]** The authors first formulate the algorithmic trading process into a Partially Observable Markov Decision Process (POMDP), so that the non-observable underlying states can also be modelled. Based on the POMDP framework and imitation learning techniques, the authors proposed their model imitative Recurrent Deterministic Policy Gradient (iRDPG).

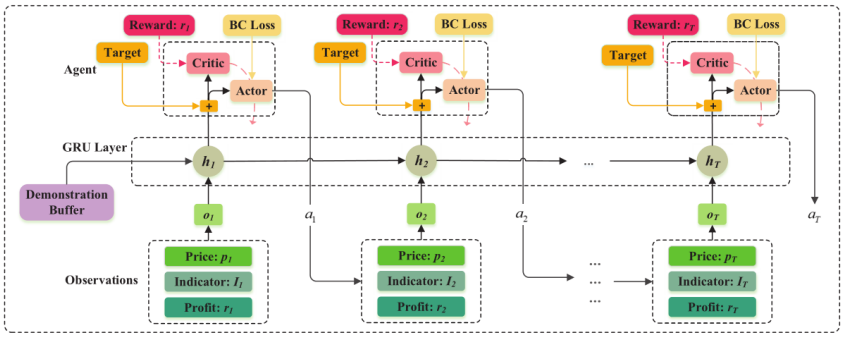

Figure 7: Architecture of iRDPG [12]

In their POMDP framework for algorithmic trading, the agent receives an observation o_t from the market and personal account a_t each time period. The observation contains the market price $\boldsymbol{p}$, technical indicators $\boldsymbol{q}$ and the account profit $\boldsymbol{r}$. Though the underlying market state is non-observable, the trained trading agent may benefit from the history $\boldsymbol{H}$. As shown in Figure 7, the observation-action history H can be described as $h_t = (o_1, a_1, \cdots, o_{t-1}, a_{t-1}, o_t)$. The authors adopt the GRU network to get the next hidden state after each action and next observation:

$$h_t = GRU(h_{t-1}, a_{t-1}, o_t)$$

For the Imitative learning part, the authors adopt a demonstration buffer and behavior cloning, the trading agent can learn a fundamental strategy at the early stage of the trading process. Finally, the authors present the generalization ability of iRDPG in different markets are good. They conducted the iRDPG algorithm on both IF and IC futures market. As shown in Table 2, the base line model Dual Thrust Strategy performs totally different on two futures markets. In contrast, the performance of iRDPG is much more stable and consistent on the both markets.

Methods	Data	Total Return (%)	Sharpe ratio	Volatility	Maxium Drawdown (%)
Dual Thrust	IF	43.40	1.110	0.369	17.24
	IC	-35.59	-0.577	0.700	65.51
iRDPG	IF	38.26	0.842	0.422	26.57
	IC	24.73	0.413	0.521	29.35

Table 2: Performance of comparison methods on IF and IC [12]

4.3 Ensemble Methods

Ensemble methods are widely applied in financial trading environment [3, 21, 22]. There are several reasons for using ensemble models of in algorithmic trading:

- Agents trained with RL algorithms are sensitive to different market conditions [15], which makes them adapted to several certain types of market.
- The Different RL models can to some extent diversify our selection of assets [22] and therefore reduce the investment risk [13].

One related work is **Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy [21]**. Actually, the approach that the authors pick the best model to trade is very easy. First, they train 3 RL trading agents (PPO, A2C, and DDPG) parallelly. Then they use a growing window as the training set and a 3-month rolling window as the validation set. Then for each three months, they select the model with the highest Sharpe ratio in the validation set to trade in the next 3 months, where Sharpe ratio is defined as

$$\text{Sharpe ratio } = \frac{\bar{r}_p - r_f}{\sigma_p}.$$

As we can see from the cumulative return in Figure 8 and Table 3, the ensemble method attains best balance between return and risk. The idea of ensemble method is quite intuitive and the performance is good based on the simple modification.

Figure 8: Plot of cumulative return for different strategies [21]

	Ensemble	PPO	A2C	DDPG	Min-Variance	DJIA
Cumulative Return	**70.4%**	83.0%	60.0%	54.8%	31.7%	38.6%
Annual Return	**13.0%**	15.0%	11.4%	10.5%	6.5%	7.8%
Annual Volatility	**9.7%**	13.6%	10.4%	12.3%	17.8%	20.1%
Sharpe Ratio	**1.30**	1.10	1.12	0.87	0.45	0.47
Max Drawdown	**−9.7%**	−23.7%	−10.2%	−14.8%	−34.3%	−37.1%

Table 3: Performance comparison among different strategies [21]

Another related work is **Dynamic stock-decision ensemble strategy based on deep reinforcement learning [22]**, which is on the basis of the previous work [21]. The authors proposed Weighted Random Strategy with Confidence (WRSC) for ensembling. And the idea is again very simple. First they choose the agent with the highest annual return as the best agent and check its confidence over the chosen action of best agent. If the confidence is higher than a threshold, then the action of the best agent is executed. If the confidence is lower than the threshold, then with the confidence of the rest 2 agents' actions as the weights, WRSC randomly selects an agent and execute its action. The work flow can be found in Figure 9.

Therefore, after using the ensemble methods, the strategy can adaptively choose the best agent among the trading agents and the random selection to some extent added the robustness of the whole model. The performance of WRSC on U.S. stock market shown in Figure 10 beats the others.

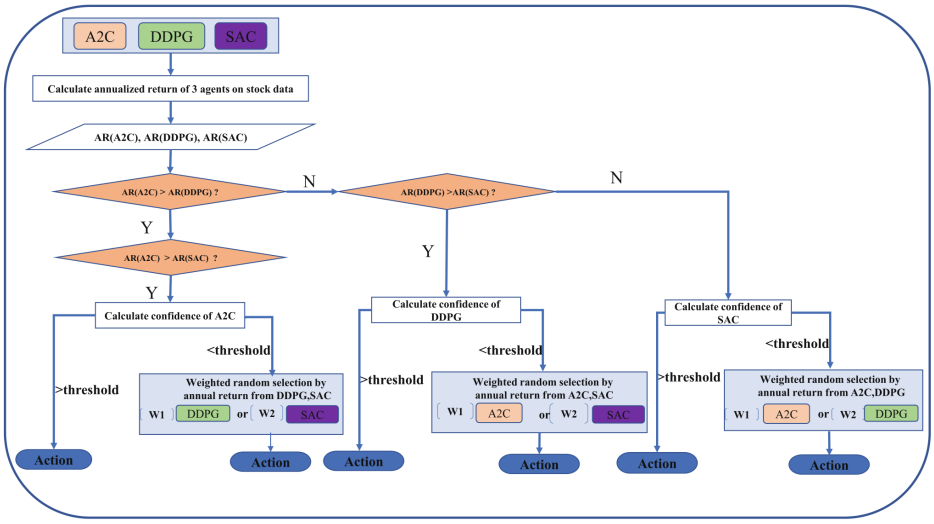

Figure 9: Workflow of the Weighted Random Strategy with Confidence (WRSC) strategy [22]

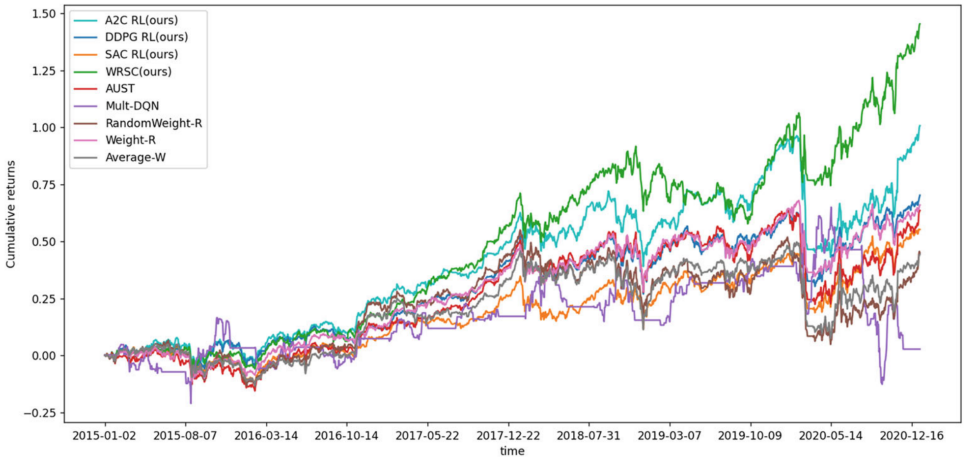

Figure 10: Cumulative return of WRSC in U.S. stock markets [22]

Therefore, the underlying idea for current ensemble methods are similar: choosing the best trading agent among the trained ones. But the relevant works have not taken an internal-ensemble pattern of different RL agents into account, which may be combined with meta learning and further explored in the future.

5 Conclusion

In this review, we mainly talked about the applications of deep reinforcement learning methods and corresponding techniques in algorithmic trading. We first formalize the financial trading problem into the Markov Decision Process setting with constraints. Then we presented RL methods and their applications including value-based, policy-based, and actor-critic methods. Finally, we summarize the techniques for RL in trading including (i) overfitting detection (ii) auto adaption for financial markets (iii) ensemble methods.

The objectives for future work may include (i) explore more advanced RL models (ii) increase adaption of RL methods for financial trading (iii) incorporate with more schemes like price prediction, risk aversion, and anomaly detection (iv) improve the robustness of the algorithms (v) make the models more interpretable.

References

[1] David H Bailey, Jonathan Borwein, Marcos Lopez de Prado, and Qiji Jim Zhu. The probability of backtest overfitting. *Journal of Computational Finance, forthcoming*, 2016.

[2] Lorenzo Bisi, Luca Sabbioni, Edoardo Vittori, Matteo Papini, and Marcello Restelli. Risk-averse trust region optimization for reward-volatility reduction. *arXiv preprint arXiv:1912.03193*, 2019.

[3] Salvatore Carta, Andrea Corriga, Anselmo Ferreira, Alessandro Sebastian Podda, and Diego Reforgiato Recupero. A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. *Applied Intelligence*, 51:889–905, 2021.

[4] Xiaoyu Chen, Xiangming Zhu, Yufeng Zheng, Pushi Zhang, Li Zhao, Wenxue Cheng, Peng Cheng, Yongqiang Xiong, Tao Qin, Jianyu Chen, et al. An adaptive deep rl method for non-stationary environments with piecewise stable context. *arXiv preprint arXiv:2212.12735*, 2022.

[5] Berend Jelmer Dirk Gort, Xiao-Yang Liu, Xinghang Sun, Jiechao Gao, Shuaiyu Chen, and Christina Dan Wang. Deep reinforcement learning for cryptocurrency trading: Practical approach to address backtest overfitting. *arXiv preprint arXiv:2209.05559*, 2022.

[6] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

[7] Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787*, 2018.

[8] Germain Lefebvre, Maël Lebreton, Florent Meyniel, Sacha Bourgeois-Gironde, and Stefano Palminteri. Behavioural and neural characterization of optimistic reinforcement learning. *Nature Human Behaviour*, 1(4):0067, 2017.

[9] Xinyi Li, Yinchuan Li, Yuancheng Zhan, and Xiao-Yang Liu. Optimistic bull or pessimistic bear: Adaptive deep reinforcement learning for stock portfolio allocation. *arXiv preprint arXiv:1907.01503*, 2019.

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[11] Xiao-Yang Liu, Zhuoran Xiong, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading. *arXiv preprint arXiv:1811.07522*, 2018.

[12] Yang Liu, Qi Liu, Hongke Zhao, Zhen Pan, and Chuanren Liu. Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2128–2135, 2020.

[13] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[15] Tidor-Vlad Pricope. Deep reinforcement learning in quantitative algorithmic trading: A review. *arXiv preprint arXiv:2106.00123*, 2021.

[16] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[17] Shuo Sun, Rundong Wang, and Bo An. Reinforcement learning for quantitative trading. *ACM Transactions on Intelligent Systems and Technology*, 14(3):1–29, 2023.

[18] Yang Wang, Dong Wang, Shiyue Zhang, Yang Feng, Shiyao Li, and Qiang Zhou. Deep q-trading. *cslt. riit. tsinghua. edu. cn*, pages 1–9, 2017.

[19] Zhicheng Wang, Biwei Huang, Shikui Tu, Kun Zhang, and Lei Xu. Deeptrader: a deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 643–650, 2021.

[20] Xing Wu, Haolei Chen, Jianjia Wang, Luigi Troiano, Vincenzo Loia, and Hamido Fujita. Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, 538:142–158, 2020.

[21] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance*, pages 1–8, 2020.

[22] Xiaoming Yu, Wenjun Wu, Xingchuang Liao, and Yong Han. Dynamic stock-decision ensemble strategy based on deep reinforcement learning. *Applied Intelligence*, 53(2):2452–2470, 2023.

[23] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep reinforcement learning for trading. *arXiv preprint arXiv:1911.10107*, 2019.