POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Market Timing Strategies with Fitted Q-Iteration and Action Persistence

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Nicola Paghera

Advisor: Professor Marcello Restelli

Co-advisor: Lorenzo Bisi, Pierre Liotet, Antonio Riva

Academic year: 2022-2023

---

1. Introduction

Artificial Intelligence (AI) has revolutionized the financial sector, particularly through the application of Reinforcement Learning (RL) algorithms [16] to develop automated systems capable of solving sequential decision problems such as trading, market timing and optimal execution. Their ability to continuously learn and adapt to changing market conditions is crucial in the fast-paced financial landscape [12].
The goal of this project is to develop a market timing model for a set of thirteen financial indexes through RL algorithms. Market data from 2000 to 2022 were analyzed, and each index was treated independently to generate tailored buy, sell, or neutral signals. The Fitted-Q Iteration [5] algorithm was implemented and trained to learn profitable market timing strategies and different action persistence was tested to find the optimal control frequency. The model's performance was evaluated on out-of-sample data, achieving positive results for most of the indexes considered.

2. Reinforcement Learning

Reinforcement Learning [16] focuses on the training of an *agent* to engage with a specific system, referred to as the *environment*, with the ultimate goal of optimizing a *reward* signal. This process involves a continuous sequence of decision-making or *action*-taking by the agent, based on its observations of the current *state* of the environment.

2.1. Markov Decision Process

The theory of Reinforcement Learning is founded upon the framework of *Markov Decision Processes* (MDPs), which are stochastic mathematical systems capable of modeling the agent-environment interaction. An MDP is defined as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu >$, where: $\mathcal{S}$ represents the set of all possible states; $\mathcal{A}$ the set of all possible actions; $\mathcal{P}$ denotes the stationary transition probability matrix, defining the probability of transitioning from state s to state s' when taking action a; $\mathcal{R}$ represents the reward function; γ is the discount factor, a value ranging from 0 to 1; and μ represents the distribution of the initial state of the environment.
In RL, the agent selects its actions based on a policy, denoted as $\pi(|s)$, which assigns a prob-

ability distribution over the action space $\mathcal{A}$ for each state s. Most RL algorithms revolve around estimating the *action-value function*, denoted as Q_π, which describes the expected future reward for taking a specific action a in a given state s while following a certain policy π. This function can be expressed as:

$$Q_\pi(s,a) := \mathbb{E}_\pi \left[\sum_{k=t+1}^{T} \gamma^{k-t+1} R_k | S_t = s, A_t = a \right].$$

The objective of the agent is to find the optimal action-value function, which is defined as:

$$Q_*(s,a) := \max_\pi Q_\pi(s,a),$$

which allows us to determine the optimal policy.

2.2. Fitted Q-Iteration

Fitted Q-Iteration [5] is a RL algorithm that approximates the optimal action-value by using a chosen regression method, in our case XGBoost [6]. It is a batch algorithm that iteratively increases the optimization horizon to learn the optimal policy. The training set used for regression is updated in each iteration, starting from the agent's past experience, and it is defined as:

$$\mathcal{D} = \{(s_t^k, a_t^k, s_{t+1}^k, r_{t+1}^k) \,|\, k = 1, 2, ..., |\mathcal{D}|\}.$$

FQI tries to generalize over the whole space $\mathcal{S} \times \mathcal{A}$ applying regression techniques over $\mathcal{D}$. Once the regressor is trained, it can estimate the value function. Specifically, at each iteration of the algorithm, the horizon considered increases by one step; given $\mathcal{Q}^*_{N-1}(s,a) \forall (s,a)$, the training set $TS = \{(i_k, o_k) | k = 1, 2, ..., |\mathcal{D}|\}$ is built, where each input is equivalent to the state-action pair (i.e. $i_k = (s_t^k, a_t^k)$) and the target is defined as $o_k = r_{t+1}^k + \gamma \max_a Q_{N-1}(s_{t+1}^k, a)$. In this way, the regression algorithm adopted is trained on TS to learn $\mathcal{Q}^*_N(s,a)$. Note that, iterating the Q-function fitting procedure leads to the propagation of estimation errors. Therefore, we have to deal with the trade-off between extending the optimization horizon considered and accumulating estimation errors through training iterations.

2.3. Persistent Action

In RL for continuous-time control problems, a common approach is to discretize time-based on a chosen control frequency. The selection of this control frequency is crucial as it offers advantages but also presents drawbacks. A higher control frequency provides more opportunities for control, which is particularly important in trading where optimal execution timing is crucial. In the context of training an agent to trade, considering a persistence parameter $\rho > 1$ forces the agent to open trades that will only be evaluated after ρ time steps and, importantly, cannot be closed before that time.

3. Related Works

In recent years, the financial trading world has been highly interested in AI applications, with a particular focus on RL. This section provides a concise overview of policy-based and value-based RL approaches applied to financial trading.

Policy Based Approaches

In [4], the execution problem in trading is analyzed, focusing on market impact and price risk. The authors propose a sub-class of execution strategies called "deep execution," which utilizes deep RL to achieve optimal execution. They compare value-based RL (using DDQN, [17]) and policy-based RL (using PPO, [14]) agents trained against the TWAP benchmark. PPO was observed to converge to the TWAP strategy in optimal environments, while DDQN struggled or required longer training. [10] utilizes advanced deep learning algorithms, including LSTM [18], Temporal Convolutional Network [9], and Transformer [7], to predict Bitcoin prices. LSTM is selected for building a deep RL agent using PPO. Experimental results demonstrate that the constructed policy generates positive returns.

Value Based Approaches

[2] apply the double Q-learning algorithm to Forex trading, using ReLU activation functions and a gradient descent algorithm. They integrate experience replay and auxiliary Q-network from previous works to improve training the neural network. The study focuses on the EUR/USD pair and demonstrates the effectiveness of the adapted algorithm for short-term speculation in the Forex market. [3] investigates the application of DQN and Deep Recurrent Q-network (DRQN [8]) in automated stock trading. They replace the fully connected layer with an LSTM layer to handle variable-size input data. Both DQN and DRQN outperform

benchmarks, with DRQN performing better due to its ability to discover profitable patterns in time-related sequences.
[11] apply RL techniques to foreign exchange trading: they employ FQI to train an artificial agent for learning profitable trading strategies. The concept of action persistence is introduced to optimize trading frequency, and the results emphasize the importance of determining the optimal one. [1] employs the FQI algorithm in Forex trading and introduces the concept of risk aversion as a risk measure. The authors adopt Multi-Objective Fitted Q-Iteration to handle the multi-objective optimization problem, balancing profit maximization and risk minimization.

4. Problem Formulation

To effectively utilize RL algorithms for identifying the optimal investment strategy, the initial step involves modeling the trading dynamics as an MDP and defining the concepts of state, action, and reward. In order to effectively do that, we first introduce some fundamental financial concepts.

4.1. Financial Preliminaries

Most of financial assets are traded on the market during the so call trading session, which is an interval of time that is typically aligned with the operating hours of the market itself. The *closing price* of an asset denotes the final price at which the instrument is traded before the trading session ends. The performance of an asset over a period of ρ trading sessions is usually evaluated by calculating its *returns* over this time interval as follows:

$$r_{t,\rho} = \frac{\text{ClosePrice}_{t+\rho}}{\text{ClosePrice}_t} - 1$$

Based on its market expectations, an investor has the option of buying, selling, or holding financial assets. Therefore, three are the possible portfolio allocation: *Long*, *Flat*, and *Short*. A *Long* position is taken when the investor expects that the value of a financial asset will increase by buying a certain number of shares of the asset. On the other hand, a *Short* position involves selling a financial asset that the investor does not own, with the expectation that its price will decrease in the future. Finally, the investor has a *Flat* position when it has not bought nor sold any share of the asset.

4.2. Environment Formulation

State
The state in an MDP contains all information needed to select the best action to maximize future rewards. In our project, we have chosen to consider a broad set of financial variables as state features to effectively capture the dynamics of the market. Then, for each combination of traded index and persistence, we have applied a *Recursive Feature Addition* algorithm to select the optimal subset of features for that specific combination. Given that, we build the state space considering both the original features and the transposition in deciles of the features themselves.
Action
Given an action persistence equal to ρ, the action consists of the allocation the agent wants to keep for the next ρ days. Therefore, the set of possible actions is defined as:

$$\mathcal{A}(s) := \begin{cases} \{0\} & \text{if } s \in \mathcal{S}^{\mathcal{T}} \\ \{-1, 0, +1\}, & \text{otherwise,} \end{cases}$$

where $\mathcal{S}^T$ is defined as the set of terminal states (i.e., the states corresponding to the last available day to take an action).
For each of the traded indexes, the action persistence ρ is set to three values: 10, 20, or 60 days. This choice aligns with the desired time horizon for signal generation.
Reward
In our project, we assess the performance of individual assets using a reward function, rather than treating the portfolio as a whole. We have chosen to invest a unitary amount of money instead of trading a fixed number of shares. This choice not only impacts the reward function but also establishes a standardized framework for evaluating and comparing the performance of different assets. We define the reward as follows:

$$R_{t+1} := \frac{(p_{t+1} - p_t)}{p_t} a_t - f \left| \frac{a_t}{p_t} - \frac{a_{t-1}}{p_{t-1}} \right|$$

where: p_t is the closing price of the asset at time t, $a_t \in \mathcal{A}$ is unitary fixed investment, $q_t = \frac{a_t}{p_t}$ is number of shares at time t, ϵ is equal

to $sign(q_{t-1} - q_t)$, and f represents transaction costs, fixed at 0.0005. The first term of the reward reflects the price change relative to the action taken, the second term represents transaction costs associated with market entry and exit.

4.3. Algorithm Selection

To choose the most suitable RL algorithm, we prefer a value-based approach over a policy-based approach. Although the policy-based approach is more efficient in terms of sample usage, it tends to be less stable and more sensitive to the quality of the function approximator and the choice of hyperparameters, which can result in sub-optimal solutions. We opt for the FQI algorithm because it is more robust compared to other value-based methods like DQN which is more sensitive to noise or inaccuracies in the data. Furthermore, FQI is more interpretable as it utilizes XGBoost or Extra Trees instead of neural networks, which are considered a black-box approach.

5. Data Analysis

After the selection of indexes to trade, a meticulous process of feature selection was performed. The starting point of this phase was the target variable we wanted to analyze: *returns*. For each index, returns were calculated for 10, 20, and 60 days, resulting in a total of 39 possible combinations.

In our project, we have chosen algorithms such as ExtraTrees and XGBoost because, in addition to effectively performing regression or classification tasks, they also offer the capability of feature selection through feature importance analysis. We tackled a classification task where the labels are based on the intensity level of returns in terms of the standard deviation (std) of returns:

- Class 0 $= \{r_t | r_t < -1 * std\}$
- Class 1 $= \{r_t | -1 * std \leq r_t < -0.4 * std\}$
- Class 2 $= \{r_t | -0.4 * std \leq r_t \leq 0.4 * std\}$
- Class 3 $= \{r_t | 0.4 * std < r_t \leq 1 * std\}$
- Class 4 $= \{r_t | r_t > 1 * std\}$

For each of the 39 possible combinations considered, we use the **Recursive Feature Addition** algorithm to select the optimal subset of features: it starts with an empty set of features and evaluates the model's performance. In each iteration, it adds the second most important feature (according to its *Feature Importance*) to the subset and re-evaluates the model's performance. If the added feature improves the performance above a certain threshold is retained in the subset. Since this algorithm uses a classification model to select the subset of optimal features, we need to tune its parameters. Therefore, a **Grid Search** algorithm was implemented for this purpose. Using XGBoost as the classification method, we have tuned *min_child_weight* and *learning_rate*. The first parameter is a regularization parameter that prevents overfitting by penalizing the creation of child nodes with low weights. The second one determines the step size at each boosting iteration in XGBoost, a smaller value makes the boosting process more conservative, improving generalization but requiring more iterations for accuracy. To run the hyperparameter tuning and the feature selection procedure in parallel, an evaluation algorithm called **grid-RFA** was created. Essentially, for each combination of XGBoost parameters selected from the grid search, we assess the performance of the classification after executing the RFA. In this way, we can choose the optimal combination of parameters based on the performance obtained by the classifier on the optimal set of features. The performance evaluation is not based on a classic accuracy score: we defined a new metric called **Custom Weighted Accuracy** in order to tackle the problem of imbalanced classes and the misclassifications of extreme classes to be penalized more heavily. To address this issue, the new metric is based on a 5x5 *Weight Matrix* (WM) where the element $w_{i,j}$ represents the penalty assigned to the model if the predicted class is j while the true class is i. The *Weight Matrix* that contains the penalization is defined as follows:

$$\text{WM} = \begin{bmatrix} 0 & 0.25 & 1.25 & 1.5 & 1.5 \\ 0.05 & 0 & 0.5 & 0.75 & 0.75 \\ 0.25 & 0.25 & 0 & 0.25 & 0.25 \\ 0.75 & 0.75 & 0.5 & 0 & 0.05 \\ 1.5 & 1.5 & 1.25 & 0.25 & 0 \end{bmatrix}$$

The elements on the main diagonal of the weight matrix are set to zero to avoid penalizing correct classifications. Moreover, misclassifications of classes that are further apart are penalized

more than misclassifications of classes that are closer. Once this matrix is established, it is multiplied element-wise with the confusion matrix generated by the classifier and, to obtain the metric value, we sum up all the elements of this new matrix and normalize it through a min-max normalization.
For the same unbalanced class problem, we developed a new objective function **Guess-Averse Loss Function**, replacing the traditional *Softmax* loss function. This new function incorporates the matrix WM for differential penalties for different types of classification errors. Following the paper [13], we have defined the *Guess-Averse Loss Function* as follows:

$$L(M, z, S(x)) = \log(1 + \sum_{j=1}^{C} WM_{z,j} e^{S_j(x) - S_z(x)})$$

where WM is the matrix presented in Section 5, C is the number of classes, z is the true class, $S(x)$ is the score given by the classifier to each class for the data point x. All these phases were carried out using a **Walk-Forward Cross-Validation** methodology, suitable for handling time series data.

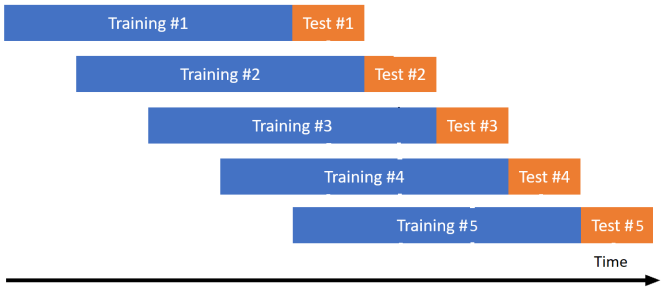


Figure 1: Walk-Forward Cross-Validation

As it can be seen in Figure 1, it is constructed with a shifting window, moving forward in time, instead of the typical time-series cross-validation where an expanding window is used. This shifting window keeps the training and test sets of fixed size and shifts over time which can be particularly useful if the underlying series are non-stationary, as in our case. The test folds are not overlapping in order to have a robust performance evaluation.
Basing our considerations on all the 39 assets, three persistence for each of the thirteen indexes, the most significant result is the fact that the RFA algorithm consistently tends to select the features transformed into decile values, supporting the hypothesis that such a transformation provides greater robustness for the classifier.

6. Experimental Results

During the training process of FQI, the dataset was divided into three periods: a training set (2003-2016), a validation set (2017-2019), and a test set (2020-2022). The training set was utilized to train the FQI algorithm, enabling the agent to learn sequential decision-making and optimize actions based on observed rewards and feedback. The validation set was crucial for fine-tuning parameters and assessing the performance of FQI's regressors, where XGBoost outperformed ExtraTree in terms of results and computational efficiency. Through systematic experimentation, we explored different combinations of regressor parameters and FQI iterations to identify the optimal model. After selecting the best-performing combination of regressor parameters and FQI iteration, the agent was retrained using the entire training period from 2003 to 2019. The model's performance was then tested on the remaining years, 2020-2022.
This methodology aimed to ensure a comprehensive and unbiased evaluation of the FQI algorithm's performance. By using separate training, validation, and test sets, we assessed the agent's effectiveness in different time periods and validated its ability to perform well on unseen data. However, the non-stationarity of financial time series poses a significant challenge in model selection since they are characterized by fluctuating patterns, trends, and regime shifts, making it difficult to identify models that consistently capture the complexities of the market. Therefore, dividing the dataset into train, validation, and test sets can lead to the exclusion of assets that may perform well in unseen data. Hence, relying on an offline validation procedure for model selection has limitations that need to be considered.

6.1. Model Selection

After the hyperparameters tuning, we test different FQI datasets, each one characterized by a different set of features. More specifically, six alternatives were considered: using the features selected by RFA, using the five probabilities ob-

tained from the classifier, or using both of them. Moreover, for each of these possibilities, we also include the last ten disjoint returns, computed with the same horizon as the persistence considered. So, in conclusion, we test six different sets of features which are reported in Table 1.

	Features
D1	RFA-Feat
D2	Probs
D3	RFA-Feat, Probs
D4	RFA-Feat, Returns
D5	Probs, Returns
D6	RFA-Feat, Probs, Returns

Table 1: FQI Datasets

To select the best set of features that will define the state space, we adopted the following methodology. For each training set, we evaluated the performance of FQI for each combination of traded asset and action persistence. Then, for each index, we selected the best persistence value in each training set based on the cumulative reward results during the validation period. Finally, we summed up these performances, which are reported in Table 2, and chose as the optimal training set the one associated with the highest results.

D1	D2	D3	D4	D5	D6
2.885	3.494	4.557	4.35	4.874	4.932

Table 2: Cumulative Rewards Sum; Validation Set

From the results collected in Table 2, the following observations can be made: the use of classifier probabilities in defining the state space improves the agent's performance, including disjoint returns significantly enhance the agent's decision-making process, and combining RFA-selected features with classifier probabilities yields improved performance, leveraging the strengths of both components.

6.2. FQI Results

After selecting the FQI training set, and determining the optimal combination of FQI hyperparameters and action persistence based on validation results, we assess then the performance on the test set. Table 1 presents both the validation and test results. The results obtained on the test set reveal that more than half of the assets demonstrate positive outcomes.

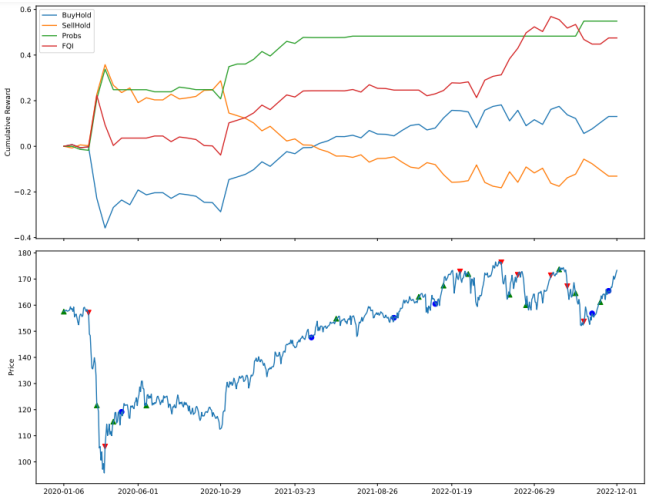
Index	Persist.	Valid.	Test
S&P	20	0.449	**0.359**
EX50	20	0.429	**0.078**
FTSE	10	0.443	**0.475**
TOPIX	20	0.378	**0.195**
Ger7-10	20	0.307	**-0.063**
UK7-10	10	0.141	**0.037**
Treas10	20	0.185	**-0.306**
HY	10	0.113	**0.251**
Ener	10	0.731	**-0.403**
Ind	10	0.365	**0.233**
Met	10	0.468	**0.225**
USD	10	0.295	**-0.161**
JPY	10	0.628	**0.016**

Table 3: Cumulative Rewards; Validation and Test Sets

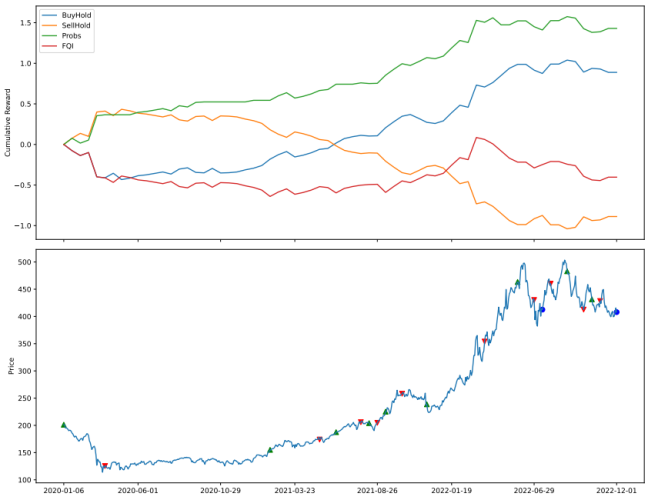
From Table 3, we can assess the impact of persistence: lower values (10 and 20 days) result in better performance, capturing shorter-term market trends and maximizing profit opportunities. In contrast, a higher persistence value of 60 days leads to poorer performance, potentially due to delayed responses to market fluctuations. Choosing the appropriate persistence value is crucial for aligning the agent's decisions with market dynamics and optimizing trading outcomes.

6.3. Baseline Strategies

In this section, we compare the performance of our FQI agent with three baseline strategies: the first two are the classical "Buy & Hold" and "Sell & Hold", the third one is a deterministic trading strategy based on the classifier probabilities defined as follows: a buy signal is generated if the predicted class is 4, a sell signal is generated if the predicted class is 0, a flat signal is generated if the predicted class is 2. If the predicted class is 1 or 3: if there are no open positions, we do not enter the market; if we are already in a position, we do not close it. By comparing these strategies, we aim to assess the effectiveness of our agent in generating higher returns and making informed investment decisions. We present the results for two assets, *FTSE* and *Ener*, both with a persistence of 10.



(a) *FSTE_10.*



(b) *Ener_10.*

Figure 2: Cumulative Reward and Trades, Test Set.

The first asset, *FTSE_10*, shows positive cumulative returns on the test set, indicating successful learning and improvement by the FQI agent. It demonstrates adaptability and active participation in the market, outperforming static strategies but falling short of the probability-based strategy. The second asset, *Ener_10*, exhibits negative cumulative rewards on the test set, likely due to high volatility in its historical series. Consequently, the agent faced challenges during the training phase. The FQI agent performs worse compared to buy-and-hold and probability-based strategies, suggesting the need for further refinement. Additionally, the FQI agent demonstrates agility by frequently adjusting its positions, which is a positive aspect of its behavior. Despite the presence of transaction costs, the agent actively seeks position changes to maximize returns. Overall, the FQI agent shows promise in generating favorable returns and reacting to market conditions, but improvements are necessary to align its performance with the probability-based strategy.

7. Conclusions

The main objective of the thesis was to develop a highly effective RL algorithm for generating market timing signals. After the feature selection phase, which allows us to determine the set of financial features with the highest predictive power on the return, we trained the FQI algorithm considering different values of action persistence. We implemented XGBoost as FQI's regressor because it offers great interpretability and accurate performance.

The results of the FQI agent indicate positive performance for the majority of the assets. This can be attributed to accurate feature engineering and the consideration of varying action persistence tailored to each asset. These findings highlight the importance of these phases in achieving successful outcomes in Reinforcement Learning-based financial modeling.

However, there are several future developments that could enhance this work, including:

- Currently, the signals are generated every persistence-days. An interesting alternative is the generation of daily signal, each one the contributes on the overall allocation with a weight of 1 / persistence. This aggregated signal could then serve as a more robust and representative input for decision-making processes.
- Considering the use of a regressor to estimate the Q-function, an intriguing avenue for further improvement lies in incorporating a higher-level algorithm, such as Conformal Prediction, to enhance the validation of the agent's choices [15].
- Adopting a more dynamic approach in selecting optimal hyperparameters and persistence values would yield better performance. In this regard, multi-expert algorithms offer a promising solution due to their online model selection procedure [12].

References

[1] Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Gianmarco Reho, Nico Montali, Marcello Restelli, and Cristiana Corno. Foreign exchange trading: A risk-averse batch reinforcement learning approach. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

[2] João Carapuço, Rui Neves, and Nuno Horta. Reinforcement learning applied to forex trading. *Applied Soft Computing*, 73:783–794, 2018.

[3] Lin Chen and Qiang Gao. Application of deep reinforcement learning on automated stock trading. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, pages 29–33. IEEE, 2019.

[4] Kevin Dabérius, Elvin Granat, and Patrik Karlsson. Deep execution-value and policy based reinforcement learning for trading and beating market benchmarks. *Available at SSRN 3374766*, 2019.

[5] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

[6] M Ester, HP Kriegel, and X Xu. Xgboost: A scalable tree boosting system. in proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (vol, pg 785, 2016). *GEOGRAPHICAL ANALYSIS*, 2022.

[7] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021.

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

[9] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017.

[10] Fengrui Liu, Yang Li, Baitong Li, Jiaxin Li, and Huiyang Xie. Bitcoin transaction strategy construction based on deep reinforcement learning. *Applied Soft Computing*, 113:107952, 2021.

[11] Antonio Riva, Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Edoardo Vittori, Marco Pinciroli, Michele Trapletti, and Marcello Restelli. Learning fx trading strategies with fqi and persistent actions. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

[12] Antonio Riva, Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Edoardo Vittori, Marco Pinciroli, Michele Trapletti, and Marcello Restelli. Addressing non-stationarity in fx trading with online model selection of offline rl experts. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 394–402, 2022.

[13] Francesco Sammarco. Non-stationary rl for forex trading with automatic market regime clustering. 2022.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[15] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[16] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[17] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[18] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.