Addressing Non-Stationarity in FX Trading with Online Model Selection of Offline RL Experts

Antonio Riva*
antonio.riva@polimi.it
Politecnico di Milano

Lorenzo Bisi*
lorenzo.bisi@mlcube.com
ML cube

Pierre Liotet*
Politecnico di Milano

Luca Sabbioni*
Politecnico di Milano

Edoardo Vittori
Politecnico di Milano
Intesa Sanpaolo

Marco Pinciroli
Intesa Sanpaolo

Michele Trapletti
Intesa Sanpaolo

Marcello Restelli
Politecnico di Milano

ABSTRACT

Reinforcement learning has proven to be successful in obtaining profitable trading policies; however, the effectiveness of such strategies is strongly conditioned to market stationarity. This hypothesis is challenged by the regime switches frequently experienced by practitioners; thus, when many models are available, validation may become a difficult task. We propose to overcome the issue by explicitly modeling the trading task as a non-stationary reinforcement learning problem. Nevertheless, state-of-the-art RL algorithms for this setting usually require task distribution or dynamics to be predictable, an assumption that can hardly be true in the financial framework. In this work, we propose, instead, a method for the dynamic selection of the best RL agent which is only driven by profit performance. Our modular two-layer approach allows choosing the best strategy among a set of RL models through an online-learning algorithm. While we could select any combination of algorithms in principle, our solution employs two state-of-the-art algorithms: Fitted Q-Iteration (FQI) for the RL layer and Optimistic Adapt ML-Prod (OAMP) for the online learning one. The proposed approach is tested on two simulated FX trading tasks, using actual historical data for the AUS/USD and GBP/USD currency pairs.

CCS CONCEPTS

• **Theory of computation** → **Sequential decision making**; • **Computing methodologies** → *Artificial intelligence*.

KEYWORDS

FX Trading, FQI, Optimistic Adapt ML Prod, Online Learning, RL

ACM Reference Format:
Antonio Riva, Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Edoardo Vittori, Marco Pinciroli, Michele Trapletti, and Marcello Restelli. 2022. Addressing Non-Stationarity in FX Trading with Online Model Selection of Offline RL Experts. In *3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3533271.3561780

*Equal contribution

ICAIF '22, November 2–4, 2022, New York, NY, USA

ACM ISBN 978-1-4503-9376-8/22/10…$15.00
https://doi.org/10.1145/3533271.3561780

1 INTRODUCTION

Automated Trading is one of the most interesting applications for artificial intelligence techniques in the financial field, where the use of AI is receiving growing attention in recent years both from the point of view of investments [41] and research [24, 43, 53]. Financial markets are indeed a formidable playground for AI algorithms as the economic consequences of each action, i.e., the profits or losses (P&L) that it generates, can be measured precisely.

This natural definition of reward especially favors the application of the Reinforcement Learning (RL) paradigm, which builds on the idea of improving the ability of an agent in interacting with an external environment by providing a reward as feedback. This has been highlighted in [43], where the authors demonstrated the RL capability of exploiting statistical patterns and market inefficiencies to optimize profits, and in [47], where RL has been applied to the option hedging problem to minimize the hedging costs.

Nevertheless, the effectiveness of these techniques when applied to real market settings strongly depends on the accuracy of the model validation and selection procedure, which often consists of back-testing the candidate strategies on the most recent available historical data [43]. This approach is inherently flawed since it builds on the assumption that the trading patterns that the selected model will face in the test phase are the same as those encountered in the validation set. However, it is well-known by practitioners that trading opportunities are ephemeral and strongly dependent on market conditions. Markets are indeed subject to regime shifts [48], which violate the stationarity assumption of the standard RL setting [40]. Non-stationary extensions of the RL setting have been recently developed [26], but they typically rely on modeling either the task distribution [33] or on the intra-task dynamics [52]. In practice, such kind of models can hardly be estimated in the trading setting, where, even in a stable regime, reliable market models are difficult to learn [27].

In this paper, we propose a novel technique to tackle the non-stationarity of financial markets which overcomes the aforementioned modeling issues. We develop a two-step approach, where an algorithmic layer is added on top of an RL part to allow a profit-driven online validation of the strategies produced by the latter. In this way, the model selection procedure becomes dynamic and it is not necessary to choose a strategy based on its performance in past regimes. Practically, the first step consists in exploiting the

favorable structure of the financial Markov decision process [5]: we employ an offline RL algorithm to obtain a set of strategies, trained under different conditions and possibly with different levels of complexity. In the second step, an online learning algorithm is in charge of updating, at the beginning of each trading day, the weights associated with these strategies and it receives an expert feedback [9] based on the P&L obtained by each strategy. These weights directly determine the portion of the total budget that is assigned to each expert to be invested.

We apply this approach to the foreign exchange (FX) market, focusing on the AUD/USD and GBD/USD pairs. We chose to study the FX market since data is readily available even for recent years, so to have at hand different market regimes, and because the trading costs tend to be very low when compared with the market volatility, which increases the opportunities of finding profitable trades.

In Section 2, we illustrate the theoretical background, introducing both the RL offline algorithm (FQI) and the expert learning approach (OAMP). In Section 3, we present the most relevant literature before discussing the problem setting in Section 4. In Section 5, we describe the two-step approach in detail, focusing on how the two algorithms interact. Finally, we analyze the experimental results in Section 6.

2 BACKGROUND

This section provides the necessary theoretical background in order to understand the proposed approach.

2.1 Reinforcement Learning

Reinforcement Learning (RL) is the branch of machine learning dedicated to sequential decision problems. In such setting, a decision-maker, called the *agent*, interacts with some system, called the *environment*, by means of a sequence of actions and it receives as feedback both some observation of the environment current *state* and a *reward* signal. If the process is Markovian and fully observable, it falls within the mathematical framework of Markov Decision Processes (MDP), which can be represented as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$. $\mathcal{S}$ constitutes the space of the possible states of the environment and $\mathcal{A}$ the one of the actions available for the agent. The transition kernel $\mathcal{P}(\cdot|s,a)$ assigns the probability of reaching state s' by taking action a while in state s. Each state-action pair is mapped to a reward by the reward function $R : \mathcal{S} \times \mathcal{A} \to [-R_{max}, R_{max}]$. Finally the discount factor $\gamma \in [0,1]$ drives the agent to balance instant rewards for future ones. The agent selects its action based on a policy, $\pi(\cdot|s)$, which assigns a distribution over the action space $\mathcal{A}$ to each state s.

The quality of some policy π is measured by its associated action-value function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. It denotes the expected return starting from state s, taking action a, and then following policy π. In some cases trajectories are modeled to have a fixed horizon T:

$$Q_\pi(s,a) \coloneqq \mathop{\mathbb{E}}_{\substack{s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)\\ a_{t+1}\sim\pi(\cdot|s_{t+1})}} \left[\sum_{t=0}^{T} \gamma^t R(s_t,a_t) | s_0 = s, a_0 = a \right]. \quad (1)$$

The goal of RL methods is to find the optimal policy $\pi^\star$ such that:

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A} : Q_{\pi^\star}(s,a) = \sup_\pi Q_\pi(s,a) =: Q^\star(s,a), \quad (2)$$

where $Q^\star$ is called the optimal Q-function. When both the transition model P and the reward function R are known, the MDP can be solved by means of dynamic programming approaches, exploiting the Bellman equation [40]. On the other hand, if the model is not available, RL techniques can be employed to approximately recover the optimal policy and/or value function.

2.2 Persistent Fitted Q-Iteration

Fitted Q-Iteration (FQI) [18] is a model-free, off-policy, and offline algorithm that aims to learn the optimal policy without directly interacting with the environment. As an offline algorithm, it estimates $Q^\star$ starting from the information gathered from the previous agent-environment interactions, and collected in the dataset:

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

where s_{t+1} is the state that the agent reaches after applying action a_t in state s_t while collecting a reward r_{t+1} for this transition. As a model-free and off-policy algorithm, it builds from $\mathcal{D}$ a sequence of Q-functions where each element is obtained by regression of the optimal Bellman operator given the previous Q-function. Specifically, at the N-th iteration, given the Q-function approximated at the previous iteration, $Q_{N-1}(s,a)$ $\forall (s,a)$, Q_N is trained on the set:

$$TS_{FQI} = \{(i^k, o^k) \,|\, k = 1, 2, ..., |\mathcal{D}|\}$$

where the input is a state-action pair (i.e., $i^k = (s_t^k, a_t^k)$), and the output is the application of the optimal Bellman operator to Q_{N-1}:

$$o^k = r_{t+1}^k + \gamma \max_{a \in \mathcal{A}} Q_{N-1}(s_{t+1}^k, a).$$

FQI can be understood intuitively as expanding the optimization horizon at each iteration. However, as the number of iterations increases, the approximation errors from the Bellman equation propagate and the sequence of Q-function may not converge to $Q^\star$.

Persistent FQI [35] extends the original algorithm to take into account the possibility of persisting actions for more than a single step. Action persistence involves repeating each action for a certain number $\rho \geq 1$ of consecutive steps. It allows to tune the control frequency which plays a fundamental role in the learning process. Indeed, when a continuous-time problem is transposed into a discrete-time MDP framework, a time discretization and therefore control frequency is introduced. A higher frequency gives the agent more control opportunities but also decreases the signal to noise ratio and negatively impacts sample complexity. Thus, higher frequency agents could potentially earn greater returns but may have more difficulties during learning. Lastly, persistence has an effect on the optimization horizon. A higher persistence agent requires fewer iterations to reach the same horizon than a less persistent agent. Therefore persistence is another tool useful to deal with the trade-off between control capacity and learning ability.

2.3 Online Learning

Online learning with expert advice. In online learning with expert advice [9], the agent (also referred to as the learner) has to guess the outcome $y_t \in \mathcal{Y}$ based on the past sequence $y_1, y_2, ..., y_{t-1}$ of events that occurred in the outcome space $\mathcal{Y}$. To do that, it is allowed to choose among the suggestions of a set of experts $\mathcal{E}$. At each step in the prediction game, every expert $e \in \mathcal{E}$ predicts an

Algorithm 1: Optimistic Adapt ML Prod

Initialize: K experts
Set $w_0^k = \frac{1}{K}$, $l_0^k = 0$, $\eta_0^k = \frac{1}{4}$, $\forall k \in [K]$
for $t = 1, 2, ..., T$ **do**
  Update $\tilde{w}_{t-1}^k = w_{t-1}^k e^{\eta_{t-1}^k m_t^k}$
  Update $p_t^k = \frac{\eta_{t-1}^k \tilde{w}_{t-1}^k}{\langle \eta_{t-1}, \tilde{w}_{t-1} \rangle}$
  Sample an expert according to $\mathbf{p}_t$, then receive loss vector $\mathbf{l}_t$
  Update $\eta_t^k = \min \left\{ \frac{1}{4}, \sqrt{\frac{\ln K}{1+\sum_{s\in[t]}(r_s^k - m_s^k)^2}} \right\}$
  Update $w_t^k = \left[w_{t-1}^k e^{\eta_{t-1}^k r_t^k - (\eta_{t-1}^k)^2 (r_t^k - m_t^k)^2} \right]^{\frac{\eta_t^k}{\eta_{t-1}^k}}$

element $a_{e,t} \in \mathcal{A}$ and incurs a loss $f(a_{e,t}, y_t)$. After its choice, the agent suffers a loss corresponding to the selected expert, but it is also allowed to observe all the other losses, differently from the *bandit* [28] and RL settings. Another important difference with the MDP formulation consists in not taking into account the process dynamics, thus in the lack of the concept of state. In the *adversarial* setting [9], the environment, by knowing the decision a_t of the agent, can choose deterministically the outcome y_t maximizing the loss $f(a_t, y_t)$. When the environment is not a real adversary, this setting may be overly conservative, thus, milder assumptions should be considered instead.

Adversarial and stochastic non-stationary settings. In the aforementioned online learning framework, the non-stationarity is due to an adversarial process since the environment can adapt to the behaviour of the agent. We refer to it as *adversarial non-stationary* setting. Instead, in the *stochastic non-stationary* setting, the dynamics evolve independently from the behaviour of the agent. This difference can be summarized in the following way. In the stochastic non-stationary setting, the environment chooses the (non-stationary) dynamics for the entire process to come at the beginning of the process. In the adversarial non-stationary setting instead, at each step of the process, the environment chooses the dynamics for the step to come. Clearly, this second setting is much more demanding.

Optimistic Adapt ML Prod. The Optimistic Prod with multiple adaptive learning rates (Optimistic Adapt ML Prod or OAMP) [50] is an expert learning algorithm that is focused on learning the optimal policy in non-stationary environments, including both switching and drifting dynamics. Furthermore, it is a parameter-free algorithm in which the learning rates η_t are adjusted dynamically, as explained in [50]. Finally, being an online learning algorithm, no training phase is needed. Analyzing in detail the steps of Optimistic Adapt ML Prod described in Algorithm 1, we can notice that the weights $\mathbf{w}_t$, and consequently the probabilities $\mathbf{p}_t$ associated with each expert are updated based on the instantaneous regret $\mathbf{r}_t$ and an appropriate estimate $\mathbf{m}_t$ of the former defined as follows:

$$r_t^k = \langle \mathbf{p}_t, \mathbf{l}_t \rangle - l_t^k, \qquad m_t^k = \langle \mathbf{p}_t, \mathbf{l}_{t-1} \rangle - l_{t-1}^k, \tag{3}$$

where $\mathbf{l}_t$ is the vector of losses measured by the experts at time t. It is not straightforward to calculate $\mathbf{m}_t$, fortunately we can approximate it efficiently as explained by the authors in [50].

3 RELATED WORKS

In this section, we first present a brief, general overview of both RL techniques and online learning algorithms applied to trading. Then, we focus on some RL approaches developed to deal with non-stationary environments.

3.1 RL for Trading

The use of RL in trading has received increasingly more attention for its goal being well aligned with trading objectives [3, 20, 34]. The first applications to trading using recurrent RL (RRL) have shown promising results [22, 37]. In [14] the authors moved forward by introducing adaptive RL, a three-layer trading system consisting of an RRL algorithm as base layer, a risk management overlay and a dynamic utility optimization layer. Later works have confirmed this encouraging direction in a variety of contexts, including high frequency trading using order book information [7] with a policy based approach or stock trading using OHLCV data [45] with DQN [36].

Focusing on the FX setting, different approaches can be found in the literature. In [22] an application of techniques from the seminal work of [37] was considered in a different setting, with several currency pairs. Q-learning has been first employed in [15] to trade GBP/USD. Later in [16] Q-learning was combined with a genetic algorithm to trade GBP/USD, USD/CHF, and USD/JPY. More specifically, the role of the genetic algorithm was to select an optimal subset of technical indicators to build the environment state. More recently its deep version, DQN has also been employed in [8, 25, 44, 49]. In particular, the authors of [44] showed that their trained agent outperforms an experienced trader when considering the EUR/USD pair. On the other hand, [49] applied DQN together with PPO to optimize the Sure-Fire statistical arbitrage policy. Three different currency pairs were considered, namely EUR/USD, GBP/USD and AUD/USD, and exchange rate times series were encoded using the Gramian Angular Field method. While, in principle, DQN could implicitly tackle non-stationarity being an online algorithm, in practice its divergence issues [1] risk to be exacerbated by drifts. Offline approaches have also been employed by [5, 43]. While these works share similarities with our approach, in their case the model validation is performed with a fixed dataset preceding the test set. Whenever a drift happens between the validation and the test, such approaches select a sub-optimal model.

3.2 Online Learning for Trading

Online learning has been used extensively in trading, though with a focus on multi-asset trading, specifically portfolio optimization. This led to a stream of the literature called Online Portfolio Optimization (OPO) [29]. The first algorithm in this field is UP [12], followed by many others such as ONS [2, 23] and OGDM [46]. The ONS algorithm has been shown to provide good performance in terms of regret, as well as feasible computational complexity. OGDM focuses on keeping transaction costs under control. More recently, the Conservative Projection (CP) [4] was devised to tackle the problem of beating a benchmark in portfolio optimization. The main differences of OPO, compared to the expert learning approaches considered in this paper is that OPO algorithms have regret guarantees with respect to the best constant rebalancing portfolio, while in

the expert learning case the objective is to have regret guarantees with respect to the best expert.

3.3 RL in Non-Stationary Environments

Non-stationary RL The work in [6] considers the possibility for the environment's dynamics to change in either abrupt or smooth manner. Part of the literature considers shifts in the dynamics in between episodes of interacting with the environment. In this case, the different dynamics are referred to as tasks. The same dynamics may be experienced multiple times by the agent. In [32] a novel algorithm is proposed, namely Restarted Q-Learning with Upper Confidence Bounds (RestartQ-UCB), which adopts a simple but effective strategy to reset the memory of the agent according to a calculated schedule. More specifically, the approximation of an optimistic action-value function is re-initialized after a certain number of episodes based on an extra optimism term (in addition to the standard Hoeffding/Freedman-based bonus) that takes into account the shift of the transition kernel (and/or of the reward function) across different episodes, which is assumed to be either abrupt or gradual. However, dynamics may also shift in between steps inside an episode. This setting is more challenging since the agent must take into account future non-stationarity to measure the effect of its actions. In the extreme case, the agent may not even experience some portions of the environment twice. Some approaches tackle these problems by implementing *change detection* methods to track the non-stationarity of the environment [13, 39], learning task generic together with task specific coefficients for the policy [19, 33], or adapt the policy online by policy iteration [10, 31]. Online learning for RL has been studied more theoretically in a setting where rewards may evolve over time [17, 30]. In [51] a black-box reduction is designed that turns an RL algorithm with optimal regret in a (near-)stationary environment into another algorithm with optimal dynamic regret in a non-stationary environment. The high-level idea behind this method is to schedule multiple instances of the base algorithm with different durations based on a carefully-designed randomized scheme. Then, based on the rewards collected by each of these instances, a sort of regret analysis is performed to check whether the MDP's dynamics changed, in which case the training phase needs to be restarted. However, despite their theoretical guarantees, these approaches are rarely used in practice. A more in-depth look on these approaches can be found in [26] and [38].

4 PROBLEM FORMULATION

In this section, we first formalize the FX trading task as an MDP and, then, we discuss its non-stationary extension.

4.1 The Forex MDP

Episodic setting. We decided to model the Forex trading problem as an episodic MDP, where each episode corresponds to a trading day. A trading day begins at 08:00 and ends at 18:00 CET, the interval in which most of the markets are open and so one can expect a good liquidity, furthermore it enables the possibility of real-time monitoring of the algorithms. The agent must take an action every ρ minutes, where ρ corresponds to the persistence. At the end of the episode, the agent is forced to close its position. There are two reasons behind this choice: first of all, considering fixed-length episodes simplifies the learning problem and allows to work in the undiscounted setting (i.e., $\gamma = 1$); secondly, it allows to neglect overnight funding costs.

State space. We consider two main components of the state: the agent's portfolio and the market information. For the former, we allow three possible portfolio allocations: *Short*, *Flat*, and *Long* positions, denoted, respectively, with -1, 0, and 1. This component evolves in a deterministic way, according to the selected action. For the latter, we recall that, in order to properly model the Forex trading environment as an MDP, it is fundamental to ensure that the Markov property holds. This means that each new state should be independent from its predecessors given the last. To ensure this property, the state should, in principle, include all the information related to past market observations that may have affected the transition to the next state. However, in practice, only a finite window of market information can be given as input to the agent. Following [5, 42], we consider the last 60 exchange rate variations[1] between consecutive minutes, the current time of the day and the current day of the week. While time features are deterministic quantities, rate variations are conditioned to the market regime.

Action space. The action consists of the portfolio allocation that the agent wants to keep for the next ρ minutes. The action set $\mathcal{A}(s)$ is identical for each state $s \in \mathcal{S}$ except for the last step of the trading day where the agent is forced to close any open positions. We further make the assumption that trades have a relatively small size and do not cause an impact on the market such as slippage.

Reward function. Given the current portfolio allocation x_t, the current exchange rate p_t, the action taken a_t the next exchange rate p_{t+1} and the fee rate ϕ, the reward received by the agent at persistence ρ is defined as:

$$r_{t+1} = a_t(p_{(t+\rho)} - p_t) - \phi|a_t - x_t|. \tag{4}$$

In other words, the reward is obtained by two terms: the first one consists in the gain (or loss) associated with the exchange rate variation, whereas the second one corresponds to the transaction fee that has to be paid to change portfolio allocation. We assume that ϕ is equal to a fixed percentage of the total amount of base currency traded (i.e., $\phi = 1e^{-5}$).

4.2 The Non-Stationary Environment

The main drawback of the previous formulation consists in defining the trading process as a single MDP, which implies a stationary transition kernel. This amounts to assuming that price patterns evolve according to the same probability distribution. Such a view can be challenged by practitioners, who often experience abrupt transitions between different market regimes. Market volatility reflects, indeed, traders' attitude toward risk, which is deeply influenced, in turn, by external factors, such as economics or politics.

In this work we move towards a more realistic setting, in which price dynamics are modelled as non-stationary, hence, a time-dependent transition kernel $\mathcal{P}_t(\cdot|s, a)$ needs to be taken into account. We have assumed trades to be small-sized, the dynamics are therefore not influenced by the agent's actions. We further

[1] Computed as the differences between the price at a certain time-step and the previous one, normalized by the value of the former one.

assume that dynamics are constant within each day, that is, for each trading day d, the agent is presented with a trading task $\mathcal{T}_d := \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_d, R, \gamma \rangle$ where $\mathcal{P}_d$ remains constant. This assumption is not very restrictive since market regimes, when there are no external events influencing the price process (e.g., Fed rate hike announcement) usually have longer time-scales than the single day. Moreover, we expect RL policies to be able to distinguish between intra-day periodical regimes, as shown in [5]. As we discuss in the next section, this hypothesis is important to justify our approach.

Market regime drifts are difficult to predict, since they typically depend on a combination of the aforementioned external causes. Therefore, differently from other works in the non-stationary RL field, we do not make any further assumption on the task distribution or the market regime dynamics such as smoothness of non-stationarity [10], known task patterns [39] or task drawn from an i.i.d. fixed distribution [33].

5 METHODS

Without explicitly modelling neither task distribution or dynamics, it is difficult to apply state-of-the-art non-stationary RL approaches (see Section 3.3). On the other hand, for a practitioner, forcing unrealistic assumptions can be even more detrimental from the P&L viewpoint. In this section, we develop an alternative approach by combining expert RL policies with an online validation procedure by means of an expert learning algorithm. The employed architecture is shown in Figure 1. At the intra-day level, each expert, independently from the others, takes an action, which is then weighted by its daily budget assigned to an online learning algorithm, namely OAMP (see Section 2.3). At the daily level of interaction, OAMP gathers an expert feedback, by decoupling the different contributions of each

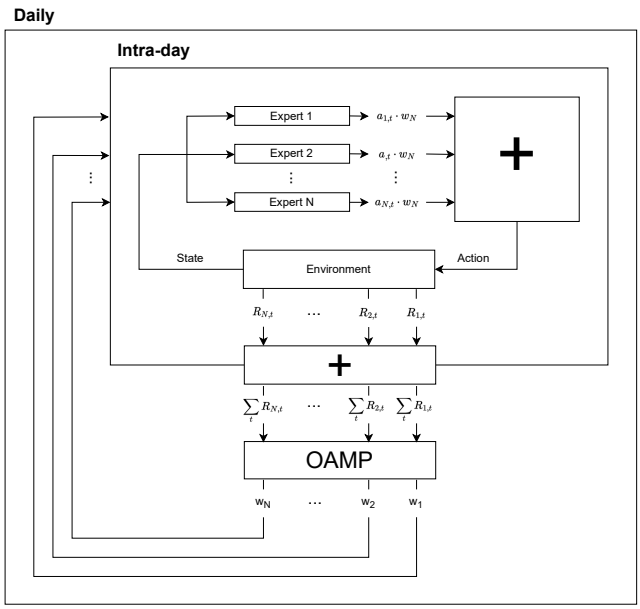

Figure 1: The inner box represents the intra-day interaction with the environment, the outer one represents the daily one. Blocks with the + symbol stand for component-wise sum.

expert in order to update its weights.[2] In what follows we give the details about how we built both levels of this architecture.

5.1 FQI Training

The experts given as input to the online learning algorithm are trading agents trained with FQI. The FQI training set is generated starting from 2 years of market data observations, which we believe is a long enough time interval to include a variety of market regimes. All the trading agents are trained by performing multiple runs of the FQI algorithm to take into account the randomness of the regression method chosen to estimate the Q-function. Each of these agents is characterized by a particular combination of hyperparameters and action persistence. Therefore, similarly to the single-expert setting analysed in [5, 43], the trading performances of the trained agents are successively validated to determine the best set of parameters. Specifically, the agents are ranked based on the average cumulative return achieved at the end of the validation year. Instead of choosing a single strategy, we select the best three agents as experts for the online learning algorithm, with the only purpose of reducing the number of experts given as input to the external layer. As observed in [43], a single RL model may overfit the validation set, hence, perform poorly in the following test phase. Therefore, selecting multiple agents allows us to be robust w.r.t. this possibility.

5.2 Online Learning with FQI Experts

Training RL policies with different hyperparameters has the effect of specializing experts for different conditions. The purpose of the online learning layer of our architecture is to automatically select the expert that best fits the current regime, based on its performance. We employed OAMP algorithm described in Section 2.3. We stress two important points about our implementation of the method.

First, we decided not to update the experts' weights after each trading step but to do that once at the beginning of each trading day and to maintain them unvaried during the rest of the episode. The reason behind this choice is related to the assumptions we made about the non-stationarity of the environment. Since we expect that the transition kernel will more likely drift between two consecutive trading days rather than in the middle of an episode, we believe that it is more appropriate to evaluate the experts' performances (and update their weights accordingly) on a daily basis. Therefore, we defined the vector of daily losses $\mathbf{l_t}$ as the negative mean cumulative return earned by the experts during the trading day. On contrary, allowing the agent to change expert during the day would have modified the portfolio held by the current expert, acting as a state for the next one, thus violating one the hypothesis of the online learning framework.

The second aspect regards the online expert selection procedure. One possibility was to first sample an expert at the beginning of each episode based on the probability distribution given by the vector of weights and then trade according to its policy. Luckily, this further element of randomness can be avoided, by assigning instead to each expert e a portion of the total budget proportional to its weight $p_{t,e}$ according to OAMP. At each trading step, all the actions suggested by the experts are first collected, then combined together to execute

[2] The decoupling can be done thanks to the hypothesis of negligible impact on the market.

a unique trade on the market. The action proposed by an expert is in turn given by the combination of the actions proposed by that expert when trained with different seeds. To summarize, at each trading step the action executed on the market is given by:

$$a_t = \sum_{e=1}^{K} p_{t,e} a_{t,e} = \sum_{e=1}^{K} p_{t,e} \frac{1}{S} \sum_{s=1}^{S} a_{t,e,s}$$

where S corresponds to the number of different seeds used to train each expert, while $a_{t,e,s}$ represents the action suggested at trading step t by the expert e trained with seed s. As a consequence, while each FQI expert is limited to take an action in the discrete set $\mathcal{A}(s) = \{-1, 0, 1\}$, the action effectively executed on the market may vary continuously between -1 and 1.

6 EXPERIMENTS

In this section, we first introduce the experimental settings and discuss the preliminary steps required to apply the Optimistic Adapt ML Prod algorithm. Then, we briefly describe the baseline strategies against which we compare the performances of the online learning algorithm. Finally, we analyze the experimental results we obtained.

6.1 Dataset Generation

Market observations of two different currency pairs, GBP/USD and AUD/USD, were collected from 2017 to 2021 from an open dataset downloaded from HistData[3]. As introduced in Section 5.1, the first two years were used to train our FQI trading agents, whereas the third one to validate these models and select the best experts. Finally, market data from 2021 were used to assess the performance of the Optimistic Adapt ML Prod algorithm and compare it with the results obtained by the baseline strategies described in 6.3.

As explained in Section 2.2, the FQI training set is derived from the experience gathered from the past, which is represented by the set $\mathcal{D} = \{(s_t^k, a_t^k, s_{t+1}^k, r_{t+1}^k) \mid k = 1, 2, ..., |\mathcal{D}|\}$. Therefore, starting from the collected market data, we first filtered them in order to focus on the selected time window defined in Section 4.1. Then, we built the vectors of market features by computing the set of exchange rate variations between consecutive minutes and adding the time of the day and the day of the week. Finally, for each of these vectors, we simulate all the possible (x_t, a_t) combinations and compute the reward r_{t+1} according to Equation 4.

6.2 Model Selection

In order to select the best set of trading experts to give as input to the online learning algorithm, we decided to train two different families of FQI agents, each of them characterized by a certain action persistence (i.e., 5 and 10) and composed of a set of models trained with different combinations of FQI hyperparameters.

Due to its great performances in terms of accuracy, but above all because of its high scalability and computational efficiency when dealing with very large training sets, we chose XGBoost [11] as the regression method to approximate the Q-function at each iteration of the FQI training algorithm. Based on our experience and following what is suggested by [21], given a reasonable value for almost all the algorithm hyperparameters, it is sufficient to tune the *min child weight* hyper-parameter to regulate the model complexity. Typically, the higher the threshold is, the simpler the trained model becomes, as trees are forced to consider a greater number of samples to perform a split, hence complicated patterns are excluded. On the other hand, a low *min child weight* allows for more complex models, but it also increases the risk of overfitting the training data.

[3]https://www.histdata.com/download-free-forex-data/

Table 1: Trading performances in validation (2020) of the best FQI iteration for each value of *min child weight* and for each action persistence. Performances are measured in terms of P&L (mean ± standard deviation) expressed as a percentage of the total amount of base currency invested.

Hyperparameters		GBP/USD		AUD/USD	
Pers.	**MinChildWeight**	**Iter.**	**P&L**	**Iter.**	**P&L**
5	500	2	8.25 ± 2.93	2	4.99 ± 2.64
	1000	5	9.00 ± 3.31	1	9.74 ± 0.51
	5000	4	12.39 ± 0.82	9	8.86 ± 0.30
	10000	**6**	**17.27 ± 4.02**	8	6.92 ± 1.74
	20000	**4**	**18.23 ± 2.05**	**1**	**10.77 ± 1.16**
	40000	**1**	**16.86 ± 0.27**	**3**	**14.47 ± 2.63**
	60000	1	16.06 ± 1.20	**1**	**19.95 ± 1.13**
	80000	2	14.80 ± 1.23	2	6.05 ± 0.39
10	500	8	13.28 ± 0.65	9	14.50 ± 3.02
	1000	9	14.55 ± 1.72	**4**	**16.24 ± 0.76**
	5000	2	21.19 ± 0.68	**9**	**18.01 ± 3.66**
	10000	6	22.03 ± 0.07	2	14.87 ± 1.33
	20000	**5**	**23.62 ± 1.27**	**10**	**16.15 ± 0.68**
	40000	**1**	**24.70 ± 1.99**	7	12.42 ± 3.28
	60000	**3**	**25.65 ± 0.41**	1	14.68 ± 0.22
	80000	7	17.11 ± 0.23	1	11.74 ± 1.80

As mentioned in Section 2.2, along with the regression method parameters, one has to tune the number of FQI iterations. As it grows, the optimized horizon increases, allowing the model to learn longer-term patterns. However, iterating the Q-function fitting procedure leads to the propagation of approximation errors.

Summing up, for each value of persistence, we have to tune the XGBoost *min child weight* parameter and the number of FQI iterations. Due to limited computational resources, we select a finite set of different *min child weight* values and train each model with ten iterations of FQI. Moreover, we perform three different runs of the training algorithm to take into account the randomness of the XGBoost regression method. As discussed in Section 5.1, to determine the best set of FQI trading experts we compare the performance of the trained agents on the validation set. For each persistence value, we first determine the best iteration for each *min child weight* value (see Table 1), then, we select the three models that attained the highest average cumulative return at the end of the validation year.

6.3 Baselines

In both trading scenarios, we decide to evaluate the performance of the OAMP algorithm comparing them with the results obtained by two benchmark strategies: the Buy&Hold and the Sell&Hold. Both are passive strategies that consist in keeping a constant portfolio position, respectively, long or short. Moreover, we consider as baseline the expert that would have been chosen in the single-expert setting, that is, the one which attained the highest average cumulative return at the end of the validation year.

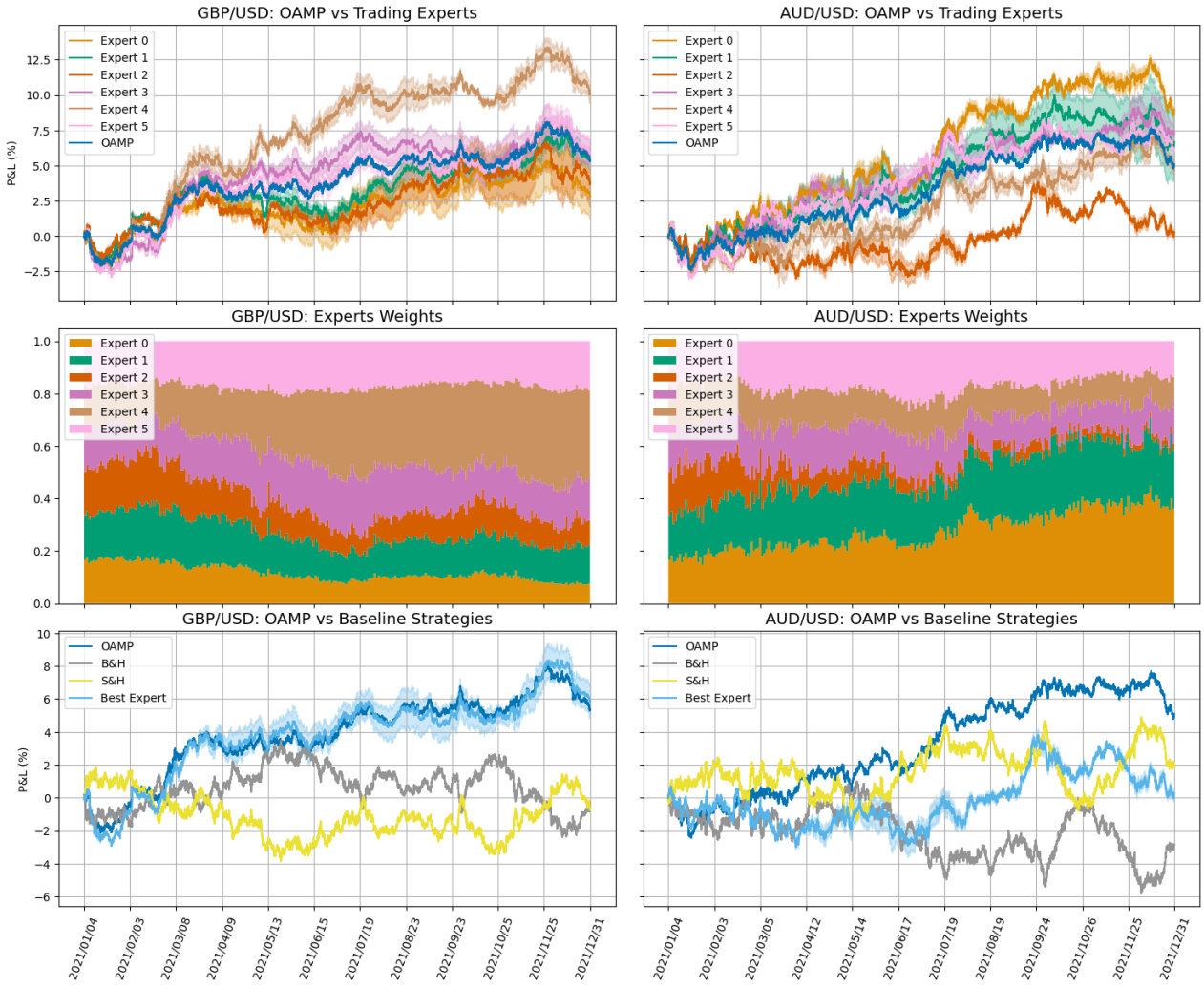

Figure 2: The left column reports results for GBP/USD and the right one for AUD/USD. Then, from top to bottom: cumulative return obtained by OAMP in test (2021) compared to the trading experts; evolution of the experts' weights during test (2021); cumulative return obtained by OAMP in test (2021) compared to baseline strategies, i.e., Buy&Hold (B&H) and Sell&Hold (S&H), and the expert that best performed in validation. Performances are reported as percentages w.r.t. the invested amount. Experts' cumulative returns are reported together with the 95% confidence intervals.

6.4 Results

Multi-expert online learning vs experts. In Figure 2, at the first row, the performance of the OAMP algorithm is compared with those achieved by the FQI trading experts. As expected, the P&L generated by the online learning algorithm in both trading scenarios is close to the average of the cumulative returns earned by all the experts in the test set. The second row of the same figure shows the evolution of the experts weights during the same year instead. As desired, the experts that obtained the highest yearly returns (i.e., *Expert 4* in the GBP/USD trading scenario and the *Expert 0* in the AUD/USD trading scenario) are those characterized by the highest weights at the end of 2021. On the other hand, experts clearly performing worse than all the others, as *Expert 2* in the AUD/USD and GBP/USD trading scenarios, reach a negligible weight at the end of the year. Finally, it can be observed that, some of the other weights do not significantly move away from the initial value (i.e., 1/K). This behaviour is due to the similar performances obtained by such experts in the analyzed period, and it is perfectly sound w.r.t. our goal: indeed, whenever experts perform equally well in some regime, there is no reason to weight one more than another one.

Multi-expert online learning vs baselines. The last row of Figure 2 compare our approach with the aforementioned baselines. It should be noted that our method always guarantees the best performance among the considered strategies for the year in exam. As shown in the figure, it earned a P&L significantly higher than the B&H and S&H strategies in both trading scenarios. Moreover, the online learning algorithm ensured greater or equal returns than

those obtained by the experts that would have been chosen through the offline validation procedure. This highlights the advantage of using an online learning algorithm as a model selection approach. While in GBP/USD the model selected with the validation set performs equally well w.r.t. our strategy, the same cannot be said for the AUD/USD scenario. In this case, our approach obtains a positive return, whereas the baseline chosen by means of validation ends the year without obtaining any profit.

7 CONCLUSIONS

Financial markets represent a difficult challenge for AI-based trading due to their intrinsic non-stationary nature. Reinforcement Learning (RL) algorithms' effectiveness strongly depends on the accuracy of the model validation and selection procedure, which often consists of back-testing the candidate strategies on the most recent available historical data. However, this approach is biased by the assumption that the trading patterns the selected model will see in the test phase are the same as those observed in the validation set. Markets are indeed subject to regime shifts that violate the stationarity assumption of the standard RL setting.

In this paper, we propose a novel technique to tackle the non-stationarity of financial markets based on a two-step approach, where an algorithmic layer is added on top of an RL method to allow a profit-driven online validation of the models produced by the latter. In practice, the first step consists of training a set of different trading experts by employing an offline RL algorithm, i.e., Fitted Q-Iteration (FQI). These experts are possibly trained under multiple market regimes and with different levels of complexity. Then, in the second step, an online learning algorithm, i.e., OAMP, is in charge of updating, at the beginning of each trading day, the experts' weights based on the P&L they obtained the day before. These weights directly determine the portion of the total budget that is assigned to each expert to be invested.

We decided to apply this approach to the foreign exchange (FX) market, especially to the AUD/USD and GBD/USD pairs, because of low transaction costs and ease of historical market data download. Experimental results show that the P&L generated by the OAMP algorithm in both trading scenarios is close to the average of the cumulative returns earned by all the experts. The OAMP algorithm manages to overperform all the baselines considered, and, more significantly, it ensured greater or equal returns than those obtained by the experts that would have been chosen through the offline model selection procedure. This empirically prove the advantage of using an online learning algorithm to dynamically validate and select the best model.

Even though the intended objectives have been reached and significant advancements have been made, different aspects of our approach can still be improved. Future research may indeed focus on enhancing either the first or the second layer of the proposed trading pipeline. For what concerns the RL layer, more can be done to produce experts which are representative of different market regimes, in order to guarantee a greater level of diversity among the experts themselves. To achieve this goal, an explicit partition of the historical dataset may be necessary, for instance by means of change point detection approaches. Regarding the external layer, despite its strong theoretical guarantees on drifting scenarios, OAMP may suffer from a lack of responsiveness in some cases. While loss engineering can fix the issue in most situations, it would be interesting to study the problem more in-depth from a theoretical perspective. A further direction of work could consist in extending the presented framework to deal with intra-day drifts. To do that, we would need to allow the second layer to act at a higher frequency, modifying the internal state after each weight balancing. A good fit for this setting could consist in applying a hierarchical RL approach, substituting the external online learning layer with a high level RL agent.

ACKNOWLEDGMENTS

The research was conducted under a cooperative agreement between Intesa Sanpaolo IMI Corporate & Investment Banking Division and Politecnico di Milano.

REFERENCES

[1] Joshua Achiam, Ethan Knight, and Pieter Abbeel. 2019. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894* (2019).
[2] A. Agarwal, E. Hazan, S. Kale, and R.E. Schapire. 2006. Algorithms for portfolio management based on the Newton method. In *International Conference on Machine Learning*. ICML, Pittsburgh, Pennsylvania, United States, 9–16.
[3] Vangelis Bacoyannis, Vacslav Glukhov, Tom Jin, Jonathan Kochems, and Doo Re Song. 2018. Idiosyncrasies and challenges of data driven learning in electronic trading. *arXiv preprint arXiv:1811.09549* (2018).
[4] Martino Bernasconi de Luca, Edoardo Vittori, Francesco Trovò, and Marcello Restelli. 2021. Conservative online convex optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 19–34.
[5] Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Gianmarco Reho, Nico Montali, Marcello Restelli, and Cristiana Corno. 2020. Foreign Exchange Trading: A Risk-Averse Batch Reinforcement Learning Approach. In *Proceedings of the First ACM International Conference on AI in Finance* (New York, New York) *(ICAIF '20)*. Association for Computing Machinery, New York, NY, USA, Article 26, 8 pages. https://doi.org/10.1145/3383455.3422571
[6] Bruce Lee Bowerman. 1974. Nonstationary Markov decision processes and related topics in nonstationary Markov chains. (1974).
[7] Antonio Briola, Jeremy Turiel, Riccardo Marcaccioli, and Tomaso Aste. 2021. Deep Reinforcement Learning for Active High Frequency Trading. *arXiv preprint arXiv:2101.07107* (2021).
[8] João Carapuço, Rui Neves, and Nuno Horta. 2018. Reinforcement learning applied to Forex trading. *Applied Soft Computing* 73 (2018), 783–794.
[9] N. Cesa-Bianchi and G. Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press.
[10] Yash Chandak, Georgios Theocharous, Shiv Shankar, Sridhar Mahadevan, Martha White, and Philip S Thomas. 2020. Optimizing for the Future in Non-Stationary MDPs. *ICML 2020* (2020).
[11] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.
[12] T.M. Cover. 1991. *The Kelly Capital Growth Investment Criterion*. World Scientific, Singapore. 181–209 pages.
[13] Bruno C. da Silva, Eduardo W. Basso, Ana L. C. Bazzan, and Paulo M. Engel. 2006. Dealing with Non-Stationary Environments Using Context Detection. In *ICML 2006*.
[14] Michael Dempster and V. Leemans. 2006. An automated FX trading system using adaptive reinforcement learning. *Expert Systems with Applications* 30 (04 2006), 543–552. https://doi.org/10.1016/j.eswa.2005.10.012
[15] Michael AH Dempster, Tom W Payne, Yazann Romahi, and Giles WP Thompson. 2001. Computational learning techniques for intraday FX trading using popular technical indicators. *IEEE Transactions on neural networks* 12, 4 (2001), 744–754.
[16] M. A. H. Dempster and Y. S. Romahi. 2002. Intraday FX Trading: An Evolutionary Reinforcement Learning Approach. In *Proceedings of Third International Conference on Intelligent Data Engineering and Automated Learning - IDEAL 2002 (Lecture Notes in Computer Science, Vol. 2412)*, Hujun Yin, Nigel M. Allinson, Richard T. Freeman, John A. Keane, and Simon J. Hubbard (Eds.). Springer, Manchester, 347–358.
[17] Travis Dick, Andras Gyorgy, and Csaba Szepesvari. 2014. Online learning in Markov decision processes with changing cost sequences. In *International Conference on Machine Learning*. PMLR, 512–520.

[18] Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 6, Apr (2005), 503–556.

[19] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[20] Thomas G Fischer. 2018. *Reinforcement learning in financial markets-a survey*. Technical Report. FAU Discussion Papers in Economics.

[21] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning* 63, Apr (2006), 3–42.

[22] Carl Gold. 2003. FX trading via recurrent reinforcement learning. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.* IEEE, 363–370.

[23] E. Hazan, A. Agarwal, and S. Kale. 2007. Logarithmic Regret Algorithms for Online Convex Optimization. *MACH LEARN* 69 (2007), 169–192.

[24] Yang Hongyang, Liu Xiao-Yang, Zhong Shan, and Walid Anwar. 2020. Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. In *ICAIF '20: ACM International Conference on AI in Finance.*

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

[26] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. 2020. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490* (2020).

[27] Jeremias Knoblauch and Theodoros Damoulas. 2018. Spatio-temporal Bayesian on-line changepoint detection with model selection. In *International Conference on Machine Learning*. PMLR, 2718–2727.

[28] Tor Lattimore and Csaba Szepesvári. 2020. *Bandit algorithms*. Cambridge University Press.

[29] B. Li and S. Hoi. 2014. Online portfolio selection: A survey. *ACM COMPUT SURV* 46, 3 (2014), 35.

[30] Yingying Li and Na Li. 2019. Online learning for markov decision processes in nonstationary environments: A dynamic regret analysis. In *2019 American Control Conference (ACC)*. IEEE, 1232–1237.

[31] Pierre Liotet, Francesco Vidaich, Alberto Maria Metelli, and Marcello Restelli. 2022. Lifelong Hyper-Policy Optimization with Multiple Importance Sampling Regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7525–7533.

[32] Weichao Mao, Kaiqing Zhang, Ruihao Zhu, David Simchi-Levi, and Tamer Basar. 2020. Near-Optimal Regret Bounds for Model-Free RL in Non-Stationary Episodic MDPs. *CoRR* abs/2010.03161 (2020). arXiv:2010.03161 https://arxiv.org/abs/2010.03161

[33] Jorge Mendez, Boyu Wang, and Eric Eaton. 2020. Lifelong policy gradient learning of factored policies for faster training without forgetting. *Advances in Neural Information Processing Systems* 33 (2020), 14398–14409.

[34] Terry Lingze Meng and Matloob Khushi. 2019. Reinforcement learning in financial markets. *Data* 4, 3 (2019), 110.

[35] Alberto Maria Metelli, Flavio Mazzolini, Lorenzo Bisi, Luca Sabbioni, and Marcello Restelli. 2020. Control frequency adaptation via action persistence in batch reinforcement learning. In *International Conference on Machine Learning*. PMLR, 6862–6873.

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

[37] John Moody and Matthew Saffell. 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks* 12, 4 (2001), 875–889.

[38] Sindhu Padakandla. 2021. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–25.

[39] Sindhu Padakandla, Prabuchandran KJ, and Shalabh Bhatnagar. 2020. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence* 50, 11 (2020), 3590–3606.

[40] Martin L Puterman. 1990. Markov decision processes. *Handbooks in operations research and management science* 2 (1990), 331–434.

[41] Refinitiv. 2021. The future of trading: Technology in 2024. https://www.refinitiv.com/en/resources/special-report/automation-trading-and-future-technologies

[42] Antonio Riva, Lorenzo Bisi, Pierre Liotet, , Luca Sabbioni, Marco Pinciroli, Marcello Restelli, Michele Trapletti, and Edoardo Vittori. 2021. Learning FX Trading Strategieswith FQI and Persistent Actions. In *Proceedings of the Second ACM International Conference on AI in Finance*. 1–9.

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

[44] Sutta Sornmayura. 2019. Robust forex trading with deep q network (dqn). *ABAC Journal* 39, 1 (2019).

[45] Thibaut Théate and Damien Ernst. 2021. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications* 173 (2021), 114632.

[46] Edoardo Vittori, Martino Bernasconi de Luca, Francesco Trovò, and Marcello Restelli. 2020. Dealing with Transaction Costs in Portfolio Optimization: Online Gradient Descent with Momentum. In *ICAIF*.

[47] Edoardo Vittori, Michele Trapletti, and Marcello Restelli. 2020. Option hedging with risk averse reinforcement learning. In *Proceedings of the First ACM International Conference on AI in Finance*. 1–8.

[48] Minh T Vo. 2009. Regime-switching stochastic volatility: Evidence from the crude oil market. *Energy Economics* 31, 5 (2009), 779–788.

[49] Chun-Chieh Wang and Yun-Cheng Tsai. 2019. Deep Reinforcement Learning for Foreign Exchange Trading. *CoRR* abs/1908.08036 (2019). arXiv:1908.08036 http://arxiv.org/abs/1908.08036

[50] Chen-Yu Wei, Yi-Te Hong, and Chi-Jen Lu. 2017. Tracking the Best Expert in Non-stationary Stochastic Environments. *CoRR* abs/1712.00578 (2017). arXiv:1712.00578 http://arxiv.org/abs/1712.00578

[51] Chen-Yu Wei and Haipeng Luo. 2021. Non-stationary Reinforcement Learning without Prior Knowledge: An Optimal Black-box Approach. *CoRR* abs/2102.05406 (2021). arXiv:2102.05406 https://arxiv.org/abs/2102.05406

[52] Annie Xie, James Harrison, and Chelsea Finn. 2021. Deep reinforcement learning amidst continual structured non-stationarity. In *International Conference on Machine Learning*. PMLR, 11393–11403.

[53] Zihao Zhang, Stefan Zohren, and Stephen Roberts. 2019. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing* 67, 11 (2019), 3001–3012.