Learning FX Trading Strategies with FQI and Persistent Actions

Antonio Riva*
antonio5.riva@mail.polimi.it
Politecnico di Milano

Lorenzo Bisi*
lorenzo.bisi@polimi.it
Politecnico di Milano

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

Automated Trading Systems are constantly increasing their impact on financial markets, but learning from historical data, detecting interesting patterns and producing profitable strategies are still challenging objectives for autonomous agents. This holds true especially in the intraday Foreign Exchange market, where prices are heavily affected by random noise and high non-stationarity. In this volatile market, opportunities are present at many time-scales, but not all of them can be easily learnt. The signal-to-noise ratio has, indeed, a critical impact on the ability of autonomous agents to learn effectively. In this paper, we formulate multi-currency trading as a Markov Decision Process and we train an agent via Fitted-Q Iteration, a Reinforcement Learning value-based algorithm. Focusing on a three-currencies framework, we study the importance of tuning the control frequency, in order to obtain effective trading policies. We backtest the developed approaches on real data from the FX market considering two currency triplets, comparing results employing either a single pair or both ones at the same time.

CCS CONCEPTS

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

KEYWORDS

Fitted Q Iteration, FX Trading, Reinforcement Learning

ACM Reference Format:
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 *2nd ACM International Conference on AI in Finance (ICAIF'21), November 3–5, 2021, Virtual Event, USA*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3490354.3494403

*Equal contribution

ICAIF'21, November 3–5, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9148-1/21/11…$15.00
https://doi.org/10.1145/3490354.3494403

1 INTRODUCTION

Trading is a sequential decision problem, where the agent needs to decide in an almost continuous fashion what position to hold in order to maximize returns. Changing allocation too often, though, can cause high transaction costs, hence, planning ahead is important. It is possible to model such a sequential decision problem in a discrete time setting as a Markov Decision Process (MDP), where the artificial trading agent observes at each time-step information from the market and decides which portfolio to hold. A changing position implies a trading cost, together with a profit or loss (p&l), which is generated depending on the position and the market movement. Since the dynamics of this process is unknown, this MDP have to be solved through the use of Reinforcement Learning (RL) [36]. The application of RL algorithms to financial trading, using only market information, has revealed to be successful since the seminal work of [31]. This approach aims at finding and leveraging patterns in the market, focusing on strategies that may span from a few minutes to a few hours, by directly training a machine learning algorithm to learn autonomously from historical data [15].

However, even if trading opportunities are present at many different time-scales, learning them is not equally difficult. While most of the autonomous agents that populate the market today operate at a very high frequency, their behavior is usually hard-coded, since at such a time-scales (milliseconds) the main opportunities are constituted by ephemeral arbitrages. On the other hand, operating at very low frequency (e.g. days, months) is impossible without including exogenous inputs regarding the market, such as economical data releases or, more generically, any news having a market impact. Operating with a middle frequency (e.g. minutes, hours) is, indeed, the more suitable scenario for a machine learning task. As we will further discuss, precisely determining the best time-scale for the learning process, is fundamental to learn profitable policies. In fact, while higher frequencies always allow, in principle, for better control, they may present a worse signal-to-noise ratio, which can deeply impact the learning performance [29].

We study the Foreign Exchange (FX) market: our goal is to apply RL techniques to the multi-currency setting, evaluating the impact of learning strategies with different frequencies. In particular, we focus on the application of a batch RL algorithm, called Fitted-Q Iteration (FQI), which allows for an efficient use of the historical data. From a financial viewpoint, the problem is tackled from a quantitative trading perspective: we operate with assets which are highly liquid in order to trade intraday without market impact, with the possibility of easily going long and short. In order to

reduce as much as possible execution delays and to deal with an execution schedule which can be pretty crowded, we assume the algorithm to be directly connected to the financial markets. Aiming at a realistic environment, we consider transaction costs, which are taken into account by the agent when constructing the trading strategies. We assume a relatively small trading size, which does not cause market impact (slippage), so to simplify the transaction costs definition. While the experimental analysis is totally focused on the FX scenario, the developed techniques can be easily extended to deal with any multi-asset task in a liquid market.

Contributions. The contribution of this paper is twofold and mainly empirical. Foremost, for the first time, we analyze the performance of FQI on a multi-asset trading scenario in a realistic setting *i.e.* with real data and transaction costs. Secondly, we experimentally study the impact of different trading frequencies on the performance of the learnt trading strategy.

Outline. This paper is organized as follows: in Section 2 we introduce the Reinforcement Learning background and FQI algorithm, with the inclusion of subsections dedicated to the definition of action persistence. In Section 3 we recall several works related to our framework, which is presented and formulated in Section 4: among the main contributions, we recall the problem formulation involving a multi-currency setting with transaction costs. The numerical results are presented in Section 5, then followed by some final considerations, in Section 6.

2 BACKGROUND

2.1 Reinforcement Learning

Reinforcement Learning is built over the concept of discrete-time Markov Decision Process (MDP) [33], describing the interactions between an agent and its environment. The MDP can be represented as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu \rangle$, where the (continuous) set $\mathcal{S}$ is the space of the possible states of the environment. The (continuous) set $\mathcal{A}$ gathers the actions that the agents can perform. The transition model $\mathcal{P}(\cdot|s,a)$ gives the probability to reach state s' for each state-action pair (s,a). The reward distribution $\mathcal{R}$ characterizes the reward $R(s,a)$ collected by the agent resulting by applying action a in state s. In this work, we assume that $\mathcal{R}$ is bounded. The discount factor $\gamma \in [0,1)$ drives the agent to balance instant rewards for future rewards. Finally, μ is the distribution of the initial state of the environment.

In RL 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. From it, we can define the goal of the agent as the maximization over its policy space of its expected discounted sum of reward over a trajectory of horizon T, also called return. For a policy π the return is defined as

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

In this work, we consider only the set of *stationary Markovian* policies. This is not restrictive since this set is proven to contain the optimal policy [33]. Another quantity of interest, which will be useful for this work is the action-value function associated to some policy π. This function gives the expected discounted future reward started from state s and taking a as a first action. It reads

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

Closely related to the function Q is the Bellman operator T^π associated to a policy π:

$$(\mathcal{T}^\pi Q)(s,a) = \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s' \sim \mathcal{P}(\cdot|s,a) \\ a' \sim \pi(\cdot|s')}} \left[Q(s', a') \right].$$

The two concept are linked in that Q_π is a fixed point of $\mathcal{T}^\pi$. Notably, the result holds also for the optimal policy π^* which is the fixed point of the optimal Bellman operator $\mathcal{T}^*$:

$$(\mathcal{T}^* Q)(s,a) = R(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \left[\max_{a' \in \mathcal{A}} Q(s', a') \right]. \quad (2)$$

By the Banach-Caccioppoli fixed point Theorem [3] we can thus obtain $Q^* = Q^\pi$ starting from any action-value function Q by recursively applying the optimal Bellman operator.

2.2 Fitted Q Iteration

Fitted Q Iteration (FQI) [14] is a RL algorithm which adopts the previous optimal Bellman operator. Its power lies in that it learns the action-value function over a space of functions by leveraging Supervised Learning techniques in order to generalize the knowledge from training dataset to unseen samples in $\mathcal{S} \times \mathcal{A}$. The training set is composed of 4-tuples (s, a, r, s'), with s the state, a the applied action, r the resulting reward and s' the next state. This set of examples can be collected once and for all at the beginning of the training since FQI is an offline algorithm. The Q-function learnt by FQI is initialized by approximating the rewards from the state-action pairs in the training set. Then from an iteration N to the next, the previous approximation Q_N is used to train Q_{N+1} to approximate the optimal Bellman operator with $r + \gamma \max_{\hat{a} \in \mathcal{A}} Q_N(s', \hat{a})$. Two conflicting phenomena appear when training FQI. On the one hand, the higher the number of iterations, the more the future outcomes are taken into account in the computation of the Q-value of each state-action. At each iteration, the horizon considered increases by one step. On the other hand the regression of the Q-function introduces errors, which are propagated through the value iterations, preventing the Q function from converging. Thus, a trade-off is usually observed in determining the best number of iterations.

2.3 Extra Trees

In this work, we use extremely randomized trees (Extra Trees) [17] as the regression algorithm used by FQI to learn the Q-function. Extra Trees are based upon decision trees [7] which work by partitioning the dataset, iteratively cutting each subset in two. The cutting threshold is randomized on two levels, hence the term extremely randomized. First, an attribute of the data is chosen at random then several thresholds for the cut are also chosen at random. Then the best threshold among them is selected under some criteria such as the Gini index or the mean-squared error. This double randomization makes the elementary regressors more uncorrelated with each other, reducing the variance of Extra Trees. This iterative process ends when the number of elements in each leave is

below a previously determined threshold called the *min_split*. Extra Trees uses a relatively important number of trees whose predictions are averaged in order to further reduce the variance of the final prediction.

An useful property of Extra Trees is that they produce an estimate of the importance of the features in predicting the target. The importance of a feature is measured as the normalized gain in terms of Gini impurity index obtain by partitioning the dataset using that feature. The higher, the more important the feature. As a consequence, features can be ranked in order to establish which are the most informative to predict the target.

2.4 Persistent action

In RL continuous time control problems are typically addressed by means of time discretization inducing a certain control frequency. On one hand a higher control frequency, especially in trading and finance, gives the agent more control opportunities; on the other hand a too fine time discretization shows several drawbacks, as for example an increase of sample complexity due to the reduced effects of the single actions. Moreover, as specified in Section 2.2, an increasing number of FQI iterations leads to a larger planning horizon and a propagation of regression errors. Hence there is a trade-off between the possibility to detect immediate opportunities and the learning capabilities. [29] introduces the idea of action persistence, which consists in the repetition of each individual action for a number of consecutive steps. Given a discrete-time MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu \rangle$ modeled at the highest possible control frequency, persistence can be seen as an environmental parameter k which can be configured to generate a family of related decision processes $\mathcal{M}_k = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_k, \mathcal{R}_k, \gamma^k, \mu \rangle$ in which, whenever an action is issued, the resulting transition lasts for k steps, with all the one-step rewards collected (with discount) in the new distribution $\mathcal{R}_k$.

3 RELATED WORKS

RL for trading. RL applications to finance have drawn more and more attention for its goal being well aligned with trading objectives [2, 15, 28]. First applications to trading using Recurrent RL (RRL) have shown promising results [18, 31]. Later works have confirmed this direction in a variety of contexts, including high frequency trading using order book information [8] with Proximal Policy Optimization [34] or stock trading using OHLCV (open, high, low, close and volume) data [37] with Deep Q Network (DQN) [30]. Of particular interest for our application, a number of works have focused on the trading of single Forex currency pairs. The approaches include RRL on several currency pairs [18], Q-learning for GBP-USD [11], DQN on EUR-USD and USDJPY [9, 35], FQI on EUR-USD [4], DQN on 12 currency pairs [21]. Experiments on the Forex are promising as highlighted by [35] whose agent outperforms an experienced trader on EUR-USD. Approaches consisting of value-based approaches such as DQN and FQI are closer to ours in essence.

RL for multi-asset trading. To the best of our knowledge, using RL to trade simultaneously more than one currency pair has not been evaluated on the Forex. However, RL framework has already been employed for other multi-asset trading tasks. For example, [24] applied Deep Deterministic Policy Gradient [25] using past price information to the allocation of a portfolio of 12 cryptocurrencies. In that case, the agent is allowed to interact with its environment every 30 minutes. Their results are inspiring since their algorithm surpasses a wide range of benchmarks from the literature. Adopting an approach similar to [24], [22] consider the daily re-balancing of 24 US stocks. Other works on multi-asset investment include [23] which learn a meta-policy with Q-learning to select which trader's allocation proposition to follow from a set of traders. [20] consider the daily trading 30 stocks from the Dow Jones. Interestingly, the authors consider an ensemble approach and select for the next testing period the RL algorithm which obtained the best Sharpe Ratio in the previous periods.

Trading time-scale. Due to the diversity of market participants and investment strategies [6, 27], it is reasonable to consider that the market is built upon different time-scales. This property is included inside the Adaptive Market Hypothesis (AMH) [1, 12] which extends the Efficient Market Hypothesis (EMH) to add the possibility that investors adapt their investment decisions based on new information. The AMH is often tested through the lens of multifractal analysis [26] to study the scaling laws of financial time series. If a time-series exhibits a mutlifractal behavior, then it does not have a characteristic scale. This implies that whatever the trading horizon, the scaled opportunities will be the same. In particular, experiments suggest that such is the case for time series from the Forex for a variety of scales [10, 16].

In the field of RL for trading, the study of the impact of the trading time-scale hasn't been a primary focus. Most works do not consider changing the frequency of interaction with the environment even though the variety of time-scales across papers ranges from high-frequency to daily to even longer periods. We note however that [32] compares quarterly, semi-annual and annual frequencies and find the latter to offer the best performance. This can be partly explained because the hyper-parameters have been tuned for the annual frequency but the author also suggests that it could be explained by the different probability distribution of the returns which would favor riskier but more profitable assets. Some previous works highlight the effect of assumptions on the trading frequency, such as transaction costs [13] or the agent's risk aversion [4]. The authors do not change the basis time-scale but the agent learns by itself to act with a lower frequency.

4 PROBLEM FORMULATION

4.1 MDP model for Forex trading

We model a generic trading task on a single asset as an MDP with a *discrete* action set. In the simplest case, *three* possible allocations are sufficient: *Long*, *Short*, or *Flat*. These actions are referred to a fixed quantity of an asset we want to trade. Concerning the reward, the following formula is used:

$$R_{t+1} = \underbrace{a_t(P_{t+1} - P_t)}_{\text{p\&l from market changes}} - \underbrace{f_t|a_t - a_{t-1}|}_{\text{transaction costs}} \tag{3}$$

where s is the portfolio, a is the action, P is the price of the asset expressed in some currency, and f is the fee multiplier. The first part consists in the gain (loss) derived from trades, and the second one corresponds to costs due to changing allocation.

While allocating a fixed amount of money could seem to be a limiting assumption, this is indeed sufficient when our goal is to maximize the expected return, as in a standard risk-neutral RL task. To be more specific, we can consider traders who believe the price of some asset is about to increase. In that case, there is no reason for them to buy only a fraction of the asset, hence, the discretization we use is sufficient. However, if they want to keep also the risk under control, then a continuous action space is desirable [5]. In mathematical terms, as soon as reward and return are essentially linear in the allocation, the problem is invariant by a global rescaling of the action. Instead, the introduction of non-linearities breaks the scale invariance by introducing dimensionful parameters either in the reward or in the return, so that reducing the allocation space implies a constrained optimization, with the possibility of reaching suboptimal extremes only.

The fact that our action space is discrete allows us to resort to value-based techniques otherwise unfeasible. We leave the extension to risk-averse trading to future research.

Two-currencies Forex trading. In the Forex market the traded asset is not a stock but a currency instead. This poses the question on how to value trades, leading to the definition of a *domestic* or *base* currency and a *foreign* currency, where we are interested in maximizing the domestic one (e.g. depending on which country a company is based in). Two options are then viable: trading a fixed quantity of the foreign currency for some variable amount of the domestic, or, vice versa, setting a fixed amount of base currency to trade. We are interested in the second case since, from a financial point of view, this allows an easier characterization of the risk exposition. With respect to the general model mentioned earlier, it is possible to see that this corresponds to treating the base currency as an asset, where in place of the price P_t we have the instantaneous exchange rate. As a result, rewards would be expressed in the foreign currency. However, in order to have an effective evaluation for the artificial agent's performances, it might be desirable to have all the returns expressed in the same measure unit, avoiding then to have them split in multiple currencies. Thus, we assume to convert the collected rewards on a daily basis. However, in order to avoid an a-posteriori conversion, we consider the following reward $\tilde{R}_{t+1} := \frac{R_{t+1}}{P_{t+1}}$, which uses the instantaneous exchange rate in place of the one at the end of the day. If we assume that exchange rates do not vary much during a trading day, and that the transaction cost for a daily conversion is negligible, we can consider the previous formula a fair approximation. We assume that the dynamics of the MDP is not controllable for what concerns the exchange rates. This means that the allocations are not large enough to move the market. The only feature of the state which is affected by the agent's actions is its current allocation. The considered episodes are only one business day long, with 1-minute long time-steps, hence, we use the undiscounted setting (i.e. we set $\gamma = 1$).

Since the state of the market is clearly impossible to observe, the following features will be used instead:

- the **last 60 exchange rate variations**[1] between consecutive minutes;
- the corresponding **time of the day**, expressed in minutes;
- the current **portfolio position** w.r.t. the currency pairs, which may assume the same values as the actions.

[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.

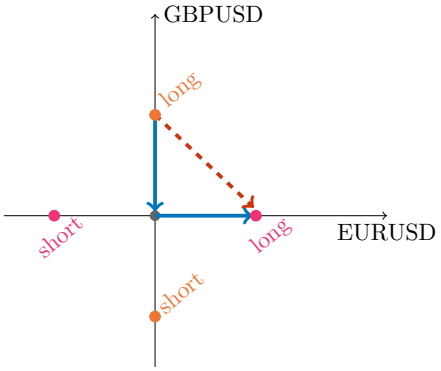

Figure 1: Three currency model. In this case, the domestic currency is USD, while EUR and GBP are the foreign ones. The dots (including the origin) represent the possible portfolio positions. In order to switch from a long position in GBP-USD pair to the same position applied on EUR-USD, the agent needs to pay twice the transaction fees for closing one position, and opening the new one (blue path).

Three-currencies Forex trading. In this work we consider also a three-currencies scenario, with two foreign currencies and a base (domestic) one. This scenario can be seen as a trading task in which two assets can be traded, hence, we have to adapt the corresponding MDP accordingly. Concerning the allocations, we disallow the positions involving simultaneous allocations on different foreign currencies. This is not restrictive since we are pursuing a risk-neutral objective. In practice, this means that it is only possible to be long/short w.r.t to one pair at each timestep[2]. Therefore, we allow the agent to take the 5 possible positions which correspond to being long (or short) w.r.t. each of the foreign currencies, or to being flat w.r.t both, as shown in Figure 1. The agent can switch from a currency pair to the other one in just one step. However, this transition is considered as the composition of two operations: the transition from a_t to the flat allocation, and the transition from the latter one to a_{t+1}. Consequently, such operations would involve a doubled transaction cost. We expect the three-currency scenario to be more profitable, since the agent, at each timestep, has more instruments to choose among, hence, it may exploits trade opportunities on both sides.

5 EXPERIMENTS

In this section, we describe how we applied FQI to the three-currency setting based on real Forex Data and we compare the results we obtained with the performances of the two-currencies models. Market data were collected from 2017 to 2020 from the Histdata platform [19]. A fixed $100k$\$ allocation was considered and, based on that, the fee f_t has been set to 1\$. Performances are shown as percentages of the invested amount.

[2]Similarly to what we mentioned before, splitting the allocation between two currency pairs is inefficient if the goal is return maximization.

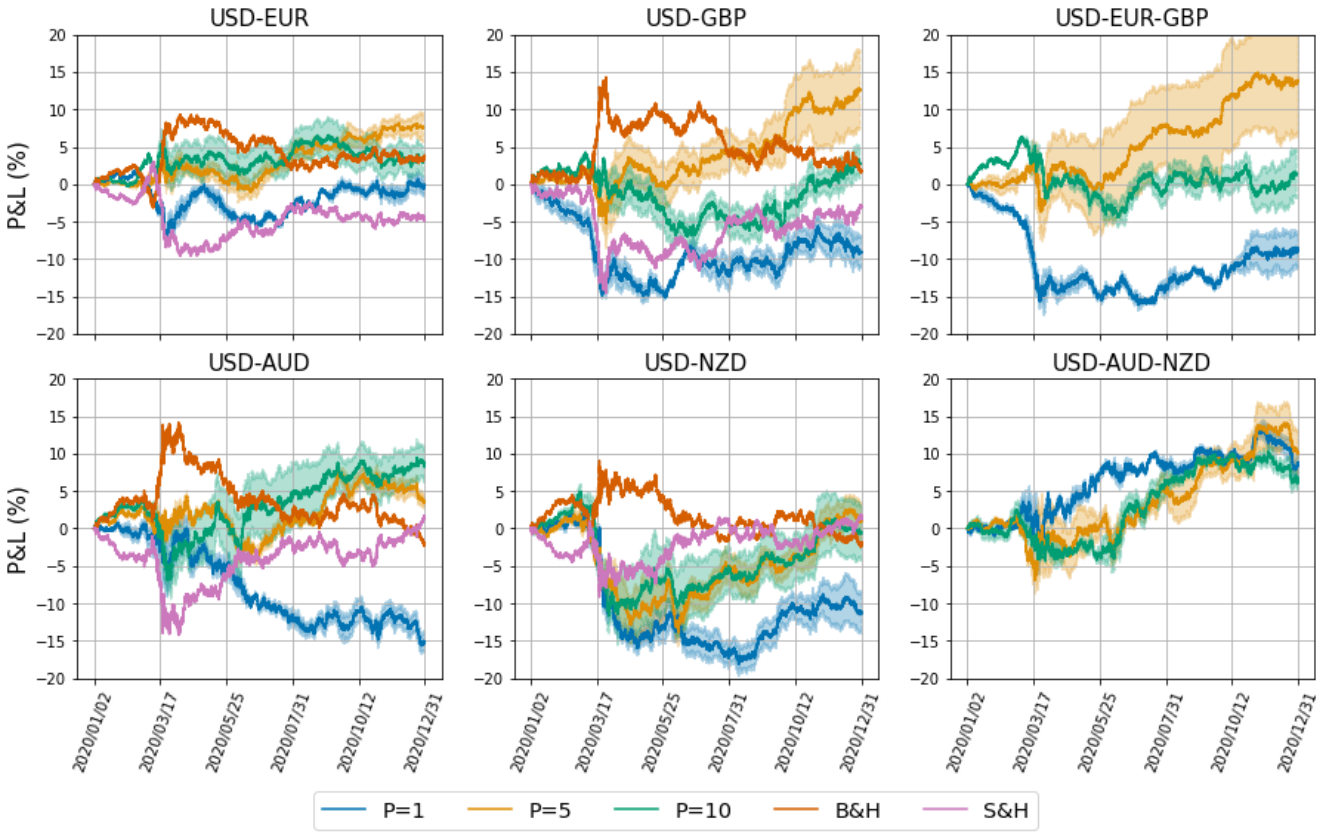

Figure 2: For each of the different currency pairs combinations, the cumulative returns on 2020 (test) of the best model are reported for each persistence value. Results are shown together with B&H and S&H baselines. Performances are reported as percentages w.r.t. the invested amount.

5.1 Dataset Generation

As explained in Section 2.2, the FQI training set is composed by a series of tuples, each of which contains the current state, the action of the agent, the next state and the reward. To build the dataset, starting from the collected market quotations, we first filtered them in order to focus on the Europe daily time window from 8:00 to 18:00 CET.[3] We then added the 60 consecutive normalized rate differences and the time of the day to each state. Finally, we associated to each pair (s, s') all the possible portfolio-action configurations and the correspondent rewards, computed using Equation 3.

5.2 Model Selection

In order to select the best FQI model, we had to tune both the hyperparameters related to the Extra Trees regressors and the ones which characterize the general training algorithm.

Extra Trees are defined by different main parameters: the number of trees, the minimum number of leaves, the minimum number of features randomly selected before each split and the minimum number of data points a node must have in order to be split again. Based on our experience and following what is suggested in [17], given a reasonable number of trees that guarantees a good trade-off between high computational time and low variance of the estimation, only the minimum sample size of nodes has to be tuned to regulate the model complexity. Typically, the higher this so-called *min_split* threshold is, the simpler is the trained model, because trees are forced to use a greater number of samples to perform a split, hence complicated patterns are excluded. On the other hand, a low *min_split* threshold allows for more complex models, but it also increases the risk of overfitting the training data.

The training algorithm, instead, is characterized only by the number of iterations. As the number of iterations 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 the approximation error. Therefore, we have to deal with the trade-off between extending the optimization horizon and propagating approximation errors through iterations.

In order to analyze the impact of persistence on the algorithm performances, we chose to consider three different values (1, 5 and 10) w.r.t. a 1-minute sampling frequency, both in the multi-currency setting and in the single-currency one. Then we train each model

[3] Central European Time. This choice is motivated by the fact that two out of the three considered currencies are European, and that this is the time with the highest traded volumes; the remaining time has been excluded for the lower trading volumes, to make the approach more consistent and robust.

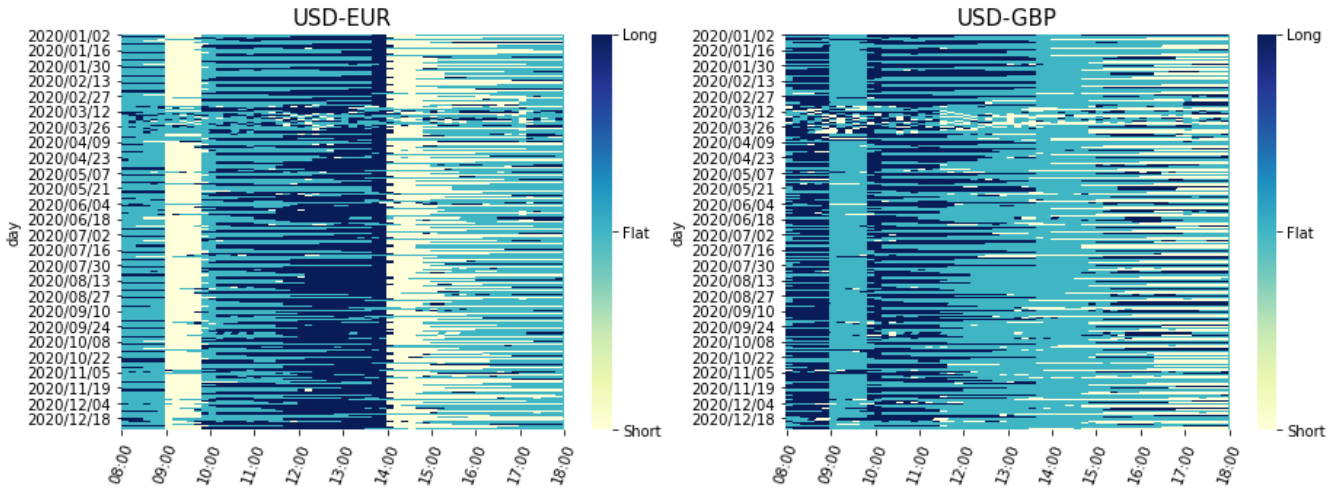

Figure 3: Portfolio allocation chosen by the agent of the three-currencies model trained with persistence equal to 10. Each row corresponds to a different business day, and each column is specific for a trading minute.

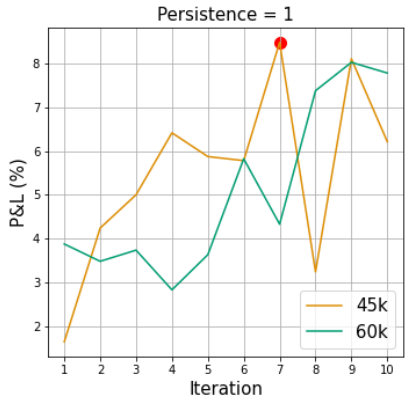

Figure 4: Validation performances for the USD-EUR case with persistence equal to 1 are shown. The final cumulative return, averaged over two seeds, is reported for each iteration, and for the different values of the *min_split* parameter. The selected model is highlighted with a red dot. Performances are reported as percentages w.r.t. the invested amount.

using two different *min_split* thresholds and fixing the maximum number of iterations to 5. Moreover, in order to take into account the randomness of Extra Trees regressors, we perform 2 different runs for each set of hyperparameters. Given the models trained on 2017-2018 data, we validate their performances using 2019 rates, in order to select the best *min_split* and iteration. For each value of the persistence, we selected as best hyperparameters set the one with the highest average cumulated return. This procedure is exemplified for the USD-EUR case in Figure 4.

Finally we test the performance of these models in 2020.

5.3 Results

In order to evaluate the selected models, we present two complementary analyses: first, we focus on the impact of the persistence both in the two-currencies setting and in the three-currencies one, by comparing the cumulative returns obtained by the trained models with some baseline strategies; then we move to the comparison between the performances of the three-currencies models and the two-currencies ones, for every value of the persistence. We summarize the performance of the best models selected for each triplet in Table 1, and we show the cumulative returns through time for each setting in Figure 2.

Impact of the persistence. We decide to compare the performance of the two-currencies models with two benchmark strategies: the Buy&Hold and the Sell&Hold. Both are passive strategies that consist in keeping a constant position, respectively, long or short. As shown in Figure 2, excluding the Australasian three-currencies setting where all the models yield nearly the same cumulative return on 2020, in the European framework, all models trained with persistence equal to 5 and 10 outperform the ones trained with persistence equal to 1 and both the benchmark strategies. The poor performances of the models with persistence equal to 1 can again be explained by the worse signal-to-noise ratio which deeply affects learning using high frequencies. Only the performances of the models trained to trade the USD-NZD currency pair are particularly poor for every value of the persistence. This is probably due to the fact that this is the less liquid currency pair among all the ones considered in this paper, which is translated into a major difficulty for the agent to find profitable patterns during the trading hours.

Looking at the policies learned by the models, another relevant fact to notice is that the higher the persistence is, the better the agent exploits temporal patterns. As we can observe in Figure 3, these patterns can be identified by looking for vertical stripes of the same color in the allocation heatmaps. These stripes becomes much more evident as the persistence increases. For instance, the agent

Table 1: performances of the selected models (best *min_split* MS and iteration Ite) for each persistence (Pers). The measures are P&L (mean ± standard deviation), Sharpe ratio and maximum drawdown (MDD) as a percentage of the $100k$ \$ allocation. Left: performance of the models EUR-USD, GBP-USD, and EUR-USD-GBP (Both). Right: performance of the models AUS-USD, NZD-USD, and AUS-USD-NZD (Both).

	Pers	MS	Ite	P&L (%)	Sharpe Ratio	MDD (%)
EUR	1	45k	7	−1.45 ± 1.16	−0.22	9.28
	5	**60k**	**1**	**6.91 ± 2.63**	**1.34**	**4.83**
	10	45k	6	1.65 ± 1.99	0.27	6.16
GBP	1	45k	1	−11.30 ± 3.05	−1.37	14.89
	5	**45k**	**2**	**14.29 ± 4.65**	**1.93**	**7.66**
	10	45k	9	6.43 ± 1.57	0.63	10.54
Both	1	75k	3	−10.12 ± 3.64	−1.45	15.63
	5	**60k**	**1**	**14.83 ± 7.34**	**2.02**	**7.96**
	10	75k	7	3.00 ± 3.40	0.33	11.07

	Pers	MS	Ite	P&L (%)	Sharpe Ratio	MDD (%)
AUS	1	45k	6	−15.14 ± 1.22	−1.74	16.46
	5	45k	7	3.63 ± 0.84	0.40	8.60
	10	**45k**	**1**	**8.83 ± 2.11**	**0.88**	**10.97**
NZD	1	60k	1	−11.17 ± 2.69	−1.18	−20.09
	5	**60k**	**1**	**0.90 ± 2.33**	**0.10**	**15.76**
	10	45k	2	−0.45 ± 3.57	−0.05	16.63
Both	1	90k	1	8.42 ± 1.24	0.87	7.47
	5	**75k**	**3**	**10.09 ± 2.74**	**1.13**	**8.56**
	10	75k	1	6.80 ± 1.08	0.81	6.90

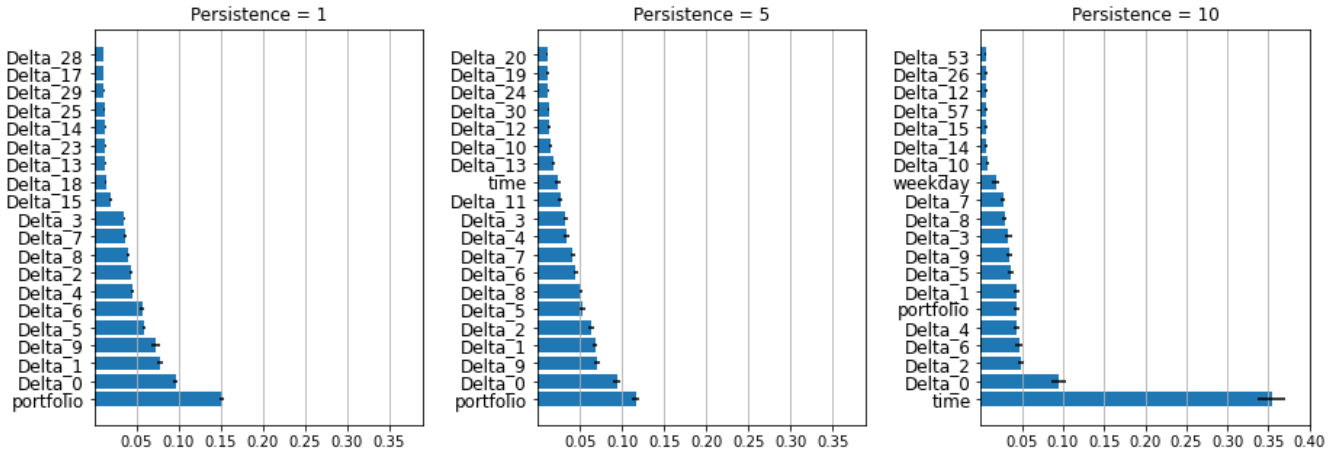

Figure 5: Feature importance of USD-EUR models trained with different persistences. The importance of a feature is measured as the normalized gain in terms of Gini impurity brought by that feature. Only the 20 most important features are shown for each setting.

of the European three-currencies model trained with persistence equal to 10 learned to be long w.r.t USD-GBP during the first hour of most of the days, then it usually changes the portfolio allocation moving to a short position w.r.t USD-EUR and keeping it until 10:00. Some of these patterns are associated with particular events which characterize the trading day: when american traders enter in the FX market around 14:00, the agent usually changes its position with respect to USD-EUR from long to short. The existence of a relationship between temporal patterns and the value of persistence is corroborated by the analysis of the feature importances. In fact, looking at the feature importances of USD-EUR models shown in Figure 5, three interesting facts can be observed. First, more recent exchange rate variations are more relevant than older ones independently of the value of persistence. Second, the importance of time-related features, (i.e., *time* and *weekday*), becomes significantly higher as the persistence increases. Finally, the importance of the *portfolio* feature is the highest in the first two plots, but it becomes less relevant for persistence equal to 10. This feature is important to predict costs, thus, its presence in the top positions of the feature importance histogram is indicative of how much costs impact a certain setting.

Besides the better performance obtained, a higher persistence allows also to have computational advantages. Given the same number of iterations, the optimization horizon becomes shorter as the persistence decreases. Therefore, higher persistences optimize on longer horizons, using the same number of iterations. Moreover, we also noticed that the mean time per iteration decreases with the persistence, even if the sample size of the FQI training set is the same. By inspecting the resulting models, we found out that the regressor trees obtained with persistence equal to 1 are characterized by almost double the number of nodes and leaves w.r.t. the ones with higher persistence. This may be due to the impact of the noise embedded in the data, which increases with lower persistences. These noisier samples make the fitting process harder, hence, a more complex model is needed.

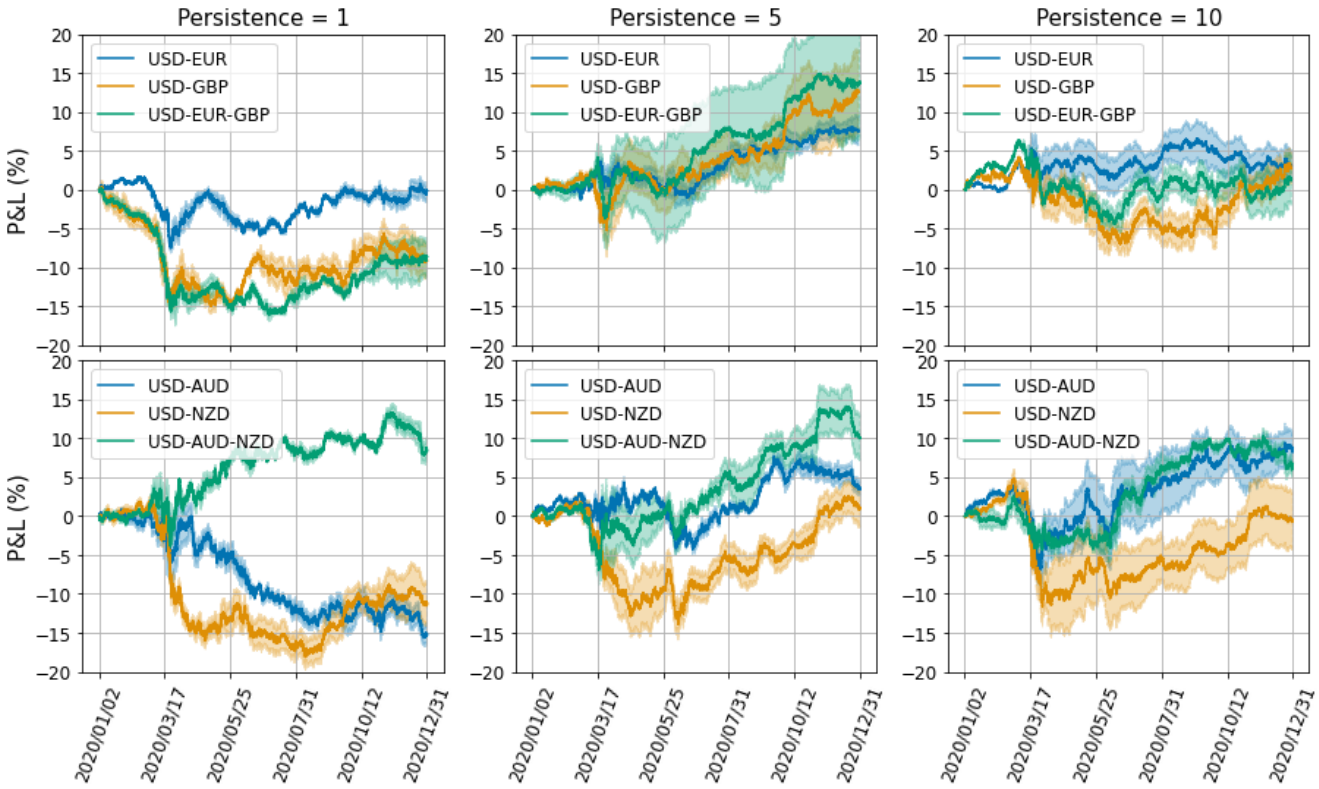

Figure 6: For each one of the different persistences, the selected models from each setting are compared w.r.t. their cumulative returns on 2020. Performances are reported as percentages w.r.t. the invested amount.

Three-currencies and two-currencies models. In Figure 6 we compared the performances of the three-currencies models and the two-currencies ones for every value of the persistence. Although they do not consistently outperform the two-currencies models through the whole year, it can be observed that three-currencies models, they are the ones which give the best average returns on both settings (see also Table 1). Moreover, it is notable that the three-currency model obtains positive performance in the Australasian setting even with persistence equal to 1.

Finally, it is worth noting that the performances of all the models, both in the two-currencies setting and in the three-currencies one, are strongly affected by multiple drawdowns registered between March and May of 2020, which might be related to the high volatility and unpredictability of the Forex market due to the spread of the Covid-19 pandemic. This strong impact of the pandemic can also be observed by looking at the portfolio allocations displayed in Figure 3, where it can be easily noticed how the solid temporal patterns learned by agent do not hold during the whole month of March, when the pandemic exploded. Nevertheless, higher persistence models were able to recover from the drawdown, ending up with a positive cumulated return. Furthermore, for what concerns the Sharpe Ratio, we can notice from Table 1 that single currency pair selected model obtain acceptable values, and the two-pairs one has an even better performance.

6 CONCLUSION

The Foreign Exchange market represents a major challenge for AI-based trading, because of its intrinsic difficulty in detecting profitable patterns due to the volatility and the non-stationarity of the exchange rates. In this paper we have developed an Automated Trading System based on Fitted Q-Iteration, a batch Reinforcement Learning algorithm where the agent, while observing the rates on a 60-minutes time window, can evaluate the effects of his possible portfolio allocations on a historical dataset. The formulation of the model is built upon a multi-currency framework, where multiple currency pairs can be analyzed in such a way that the agent can leverage the best trade opportunities he finds, with the transaction fees forcing it to plan the portfolio allocations with a larger horizon. Consequently, this raises some interesting questions regarding the optimal trading frequency, since the agent experiments a trade-off between the choice of a higher control frequency, which can bring more control opportunities, and a lower one, leading to a gain in terms of sample complexity and in planning horizon. Indeed, the results show that the agent acting once every 5 minutes can detect patterns more easily than the one acting every minute and the one keeping (persisting) its position for 10 minutes.

There are several possible future research directions directly related to our work. First of all, in our formulation, the three-currency framework is modelled as a portfolio with two assets: when the

agent chooses to change asset, it is forced to pay twice the transaction costs. If we keep into consideration the fact that we are dealing with currencies, we can take into account the missing pair of the triplet, which is never involved in this work. Secondly, in order to have a more realistic setting, many of the assumptions we took can be dropped. Transaction costs, instead of being fixed, should reflect the market behaviour, hence, they should include the instantaneous bid-ask spread for each portfolio adjustment. Execution times could be taken into account, evaluating the impact of a delay in the allocations. Future works could explicitly take into account non-stationarity, by employing, e.g., sliding-window approaches. Finally, there is the need for financial traders to measure the risk of their portfolio positions: a rich stream of Reinforcement Learning literature is devoted to study risk-aversion, and it has developed algorithms to optimize different risk-measures. Current performances are deeply conditioned by the presence of drawdowns, which a risk-averse optimization may avoid. Taking into account risk, however, implies the need to consider partial allocations, possibly in a continuous way. Actor-critic algorithms may represent the right candidates to put together the advantages from the value-based techniques and the flexibility of policy gradient approaches, in order to obtain effective risk-averse Automated Trading Systems.

ACKNOWLEDGMENTS

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

REFERENCES

[1] W Andrew. 2004. Lo. The adaptive markets hypothesis. *The journal of portfolio management* 30, 5 (2004), 15–29.
[2] 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).
[3] Stefan Banach. 1922. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. math* 3, 1 (1922), 133–181.
[4] 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
[5] L Bisi, L Sabbioni, E Vittori, M Papini, and M Restelli. 2020. Risk-Averse Trust Region Optimization for Reward-Volatility Reduction. In *29th International Joint Conference on Artificial Intelligence, IJCAI 2020*. 4583–4589.
[6] Jean-Philippe Bouchaud and Marc Potters. 2003. *Theory of financial risk and derivative pricing: from statistical physics to risk management*. Cambridge university press.
[7] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and regression trees*. CRC press.
[8] Antonio Briola, Jeremy Turiel, Riccardo Marcaccioli, and Tomaso Aste. 2021. Deep Reinforcement Learning for Active High Frequency Trading. *arXiv preprint arXiv:2101.07107* (2021).
[9] João Carapuço, Rui Neves, and Nuno Horta. 2018. Reinforcement learning applied to Forex trading. *Applied Soft Computing* 73 (2018), 783–794.
[10] Marco Corazza and A Tassos G Malliaris. 2002. Multi-fractality in foreign currency markets. *Multinational Finance Journal* 6, 2 (2002), 65–98.
[11] 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.
[12] Tiziana Di Matteo, Tomaso Aste, and Michel M Dacorogna. 2003. Scaling behaviors in differently developed markets. *Physica A: Statistical Mechanics and its Applications* 324, 1-2 (2003), 183–188.
[13] Thomas Elder. [n.d.]. Creating Algorithmic Traders with Hierarchical Reinforcement Learning MSc Dissertation. ([n. d.]).
[14] Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 6, Apr (2005), 503–556.
[15] Thomas G Fischer. 2018. *Reinforcement learning in financial markets-a survey*. Technical Report. FAU Discussion Papers in Economics.
[16] Matthieu Garcin. 2019. Fractal analysis of the multifractality of foreign exchange rates. (2019).
[17] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning* 63, 1 (2006), 3–42.
[18] 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.
[19] HistData.com. [n.d.]. *Free Forex Historical Data*. https://www.histdata.com/download-free-forex-data/
[20] 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*.
[21] Chien Yi Huang. 2018. Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787* (2018).
[22] Miquel Noguer i Alonso and Sonam Srivastava. 2020. Deep Reinforcement Learning for Asset Allocation in US Equities. *CompSciRN: Other Machine Learning (Topic)* (2020).
[23] O Jangmin, Jongwoo Lee, Jae Won Lee, and Byoung-Tak Zhang. 2006. Adaptive stock trading with dynamic asset allocation using reinforcement learning. *Information Sciences* 176, 15 (2006), 2121–2147.
[24] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. 2017. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059* (2017).
[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
[26] Renaud Lopes and Nacim Betrouni. 2009. Fractal and multifractal analysis: a review. *Medical image analysis* 13, 4 (2009), 634–649.
[27] Rosario N Mantegna and H Eugene Stanley. 1999. *Introduction to econophysics: correlations and complexity in finance*. Cambridge university press.
[28] Terry Lingze Meng and Matloob Khushi. 2019. Reinforcement learning in financial markets. *Data* 4, 3 (2019), 110.
[29] 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.
[30] 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).
[31] John Moody and Matthew Saffell. 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks* 12, 4 (2001), 875–889.
[32] Parag C Pendharkar and Patrick Cusatis. 2018. Trading financial indices with reinforcement learning agents. *Expert Systems with Applications* 103 (2018), 1–13.
[33] Martin L Puterman. 1990. Markov decision processes. *Handbooks in operations research and management science* 2 (1990), 331–434.
[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[35] Sutta Sornmayura. 2019. Robust forex trading with deep q network (dqn). *ABAC Journal* 39, 1 (2019).
[36] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[37] Thibaut Théate and Damien Ernst. 2021. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications* 173 (2021), 114632.