Foreign Exchange Trading: A Risk-Averse Batch Reinforcement Learning Approach

Lorenzo Bisi[*][†]
Pierre Liotet[*]
Luca Sabbioni[*]
lorenzo.bisi@polimi.it
Politecnico di Milano
ISI Foundation

Gianmarco Reho[‡]
Nico Montali[§]
Marcello Restelli
Politecnico di Milano

Cristiana Corno
Advanced Global Solutions

ABSTRACT

Automated Trading Systems' impact on financial markets is ever growing, particularly on the intraday Foreign Exchange market. Historically, the FX trading systems are based on advanced statistical methods and technical analysis able to extract trading signals from financial data. In this work, we explore how to find a trading strategy via Reinforcement Learning by means of a state-of-the-art batch algorithm, Fitted Q-Iteration. Furthermore, we include a Multi-Objective formulation of the problem to keep the risk of noisy profits under control. We show that the algorithm is able to detect favorable temporal patterns, which are used by the agent to maximize the return. Finally, we show that as risk aversion increases, the resulting policies become smoother, as the portfolio positions are held for longer periods.

CCS CONCEPTS

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

KEYWORDS

Fitted Q Iteration, Extra-trees, FX-Trading, Risk-Aversion, Multi Objective, Reinforcement Learning

ACM Reference Format:
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 *ACM International Conference on AI in Finance (ICAIF '20), October 15–16, 2020, New York, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3383455.3422571

[*]These authors contributed equally to this research.
[†]Contact author
[‡]Now at Likke.
[§]Now at Waymo.

1 INTRODUCTION

The Foreign Exchange (FX) market is the largest financial market in the world, even larger than the stock market, with a daily volume of $6.6 trillion, vs. $84 billion for equities worldwide, according to the 2019 Triennial Central Bank Survey of FX and OTC derivatives markets. Having such a large trading volume can bring many advantages to traders. Market participants use FX to hedge against international currency and interest rate risk, to speculate on geopolitical events, and to diversify portfolios, among several other reasons. Thanks to the fact that high-frequency data is easy to obtain, together with the increase in computational power of the machines, the development of new algorithms and machine learning applications in this field is not surprising. One of the most interesting applications to the trading problem is the creation of autonomous systems able to outperform human traders, often based on technical analysis or statistical tools aimed at detecting particular patterns.

Reinforcement Learning [34] deals with the problem of an agent having the goal of maximizing the (discounted) cumulative rewards, obtained through actions performed in interaction with an environment in a sequential decision-making process. Trading can be framed as an application of RL techniques which, in recent years, are achieving outstanding results in other fields [2, 31].

The ability to obtain a policy starting from a set of raw data without having to rely on any economic or financial assumptions makes RL a valid alternative to the approaches typically adopted in the study of the financial time series. In this paper, we begin by defining the trading activity as a Markov Decision Process, where agents can choose to change positions every minute in a day; the reward is represented as the profit generated by the adopted trading strategy. A non-trivial, but realistic property modeled in our environment is the presence of transaction fees, which makes the problem more difficult to solve. We carried out an analysis of the importance of each feature, in order to adopt, in a second instant, one of the state-of-the-art RL algorithm, Fitted Q Iteration (FQI) [15].

Financial traders need to always measure the risk related to the positions held in their portfolios, leading to the concept of Risk Aversion [24]. Hence, it is possible to consider different policies associated with different risk aversions in the agents' behaviors. In this paper, we consider as a risk measure the *reward volatility* [4, 39], which evaluates the uncertainty about the step-by-step rewards obtained from the environment. Profit maximization along with risk minimization create a multi-objective optimization problem which, in this paper, is tackled by means of Multi-Objective Fitted Q-Iteration (MOFQI) [9], a generalization of FQI. We provide an empirical evaluation of the aforementioned techniques on different historical datasets using data from 2014 to 2019. We show that policies have learned to follow some temporal patterns that repeat over the days and become smoother as risk aversion increases.

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 definitions of the risk measures adopted and Multi-Objective Reinforcement Learning. In Section 3 we recall some works related to our problem, presented and formulated in Section 4. The results of our numerical simulations are presented in Section 5 and compared to baseline strategies. Some final comments and considerations are shown in Section 6.

2 BACKGROUND

2.1 Reinforcement Learning

A discrete-time Markov Decision Process (MDP) [28] is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu \rangle$, where $\mathcal{S}$ is the (continuous) state space, which is the set of all observations that can be obtained from an environment. $\mathcal{A}$ is the (continuous) action space, i.e. the set of all possible actions that the agents can perform w.r.t. the environment. $\mathcal{P}(\cdot|s,a)$ is a Markovian transition model that assigns to each state-action pair (s,a) the probability of reaching the next state s', $\mathcal{R}$ is the distribution of rewards which can be seen as the (stochastic) immediate gain obtained by the agent from the environment as a consequence of the action chosen. This distribution is bounded by hypothesis. $\gamma \in [0,1)$ is the discount factor, a parameter close to the economic field, quantifying the benefits of having immediate rewards, instead of delaying them in the future. Finally, μ is the initial state distribution. The policy of an agent is characterized by $\pi(\cdot|s)$, which assigns to each state s the density distribution over the action space $\mathcal{A}$. Hence, the policies we are dealing with are *stationary* (i.e. they do not depend on t) and *Markovian*, meaning that the distribution does not depend on the past states. We define the set of all Markovian, stationary policies as Π, which can be shown to contain always the optimal one [28].
Following a trajectory $\tau := (s_0, a_0, s_1, a_1, s_2, a_2, ...)$ with horizon T, the return is defined as the discounted cumulative reward encountered along the trajectory: $G_\tau = \sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t)$. and the expected return (or *performance*) J_π is defined as the expected value of G w.r.t. the probability distribution of the trajectories following π.

Sometimes, the (discounted) state-occupancy measure induced by π will be used:

$$d_{\mu,\pi}(s) := (1-\gamma) \int_S \mu(s_0) \sum_{t=0}^{\infty} \gamma^t p_\pi(s_0 \xrightarrow{t} s) \, \mathrm{d}s_0,$$

where $p_\pi(s_0 \xrightarrow{t} s)$ is the probability of reaching state s in t steps from s_0 following π.
For each state s and action a, the action-value function is defined as:

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

which can be recursively defined by the following Bellman equation:

$$Q_\pi(s,a) = \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s' \sim \mathcal{P}(\cdot|s,a) \\ a' \sim \pi(\cdot|s')}} \left[Q_\pi(s', a') \right].$$

For each state s, we define the state-value function of the stationary policy $\pi(\cdot|s)$ as:

$$V_\pi(s) := \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim \mathcal{P}(\cdot|s_t,a_t)}} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) | s_0 = s \right] \quad (2)$$

Fitted Q-Iteration is a value-based algorithm: this means that the goal is to find the optimal value function, i.e.

$$Q^*(s,a) := \sup_{\pi \in \Pi} Q_\pi(s,a) \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A} \quad (3)$$

A policy π is *greedy* w.r.t. a Q-function if it plays only greedy actions, i.e. $\pi(\cdot|s) = \arg\max_{a \in \mathcal{A}} Q(s,a)$. An *optimal policy* $\pi^* \in \Pi$ is any greedy policy w.r.t. Q^*.
In order to retrieve the optimal value function, great importance is given to the following definition:[1]

Definition 1. *The* Bellman optimality operator $T^* : \mathcal{B}(\mathcal{S} \times \mathcal{A}) \to \mathcal{B}(\mathcal{S} \times \mathcal{A})$*, applied on an action value function* $Q \in \mathcal{B}(\mathcal{S} \times \mathcal{A})$*, is such that:*

$$(T^* Q)(s,a) = R(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \left[\max_{a' \in \mathcal{A}} Q(s', a') \right] \quad (4)$$

This Bellman operator is a γ-contraction and optimal action-value function is its fixed point. Therefore it is possible to apply the Banach-Caccioppoli fixed point Theorem [1] to show that the fixed point is unique and that $\lim_{k \to \infty} (T^*)^k Q = Q^*$.

2.2 Fitted Q Iteration

FQI [15] is a model-free, off-policy, and offline algorithm. It is designed to learn a good approximation of the optimal action-value function $Q^*(s,a)$ by exploiting the Bellman optimality operator. Its innovative approach consists in the application of Supervised Learning techniques as, for example, Extra Trees, widely used in our work. The algorithm considers a full dataset F containing the information collected from experience. In particular, each row represents an interaction with the environment composed of a 4-tuple, containing current state and action performed, and the values received from the environment, i.e., the reward and the next state.[2]
In brief:

$$F = \{(s_t^k, a_t^k, r_{t+1}^k, s_{t+1}^k) | \; k = 1, 2, \ldots, |F|\} \quad (5)$$

In complex real problems the possible combinations of actions and states is huge (or infinite in a continuous problem), hence the dataset does not contain all possible outcomes; FQI tries to generalize over the whole space $\mathcal{S} \times \mathcal{A}$ applying regression techniques over F. Once the regressor is trained, it can estimate the value function. Specifically, at each iteration of the algorithm, the horizon considered increases of one step; given $Q^*_{N-1}(s,a) \; \forall (s,a)$, the training set $TS = \{(i^k, o^k)\}_{k=1,\ldots,|F|}$ is built, where each input is equivalent to the state-action pair (i.e. $i^k = (s_t^k, a_t^k)$), and the target is the

[1] where $\mathcal{B}(\mathcal{X})$ denotes the space of bounded measurable functions over $\mathcal{X}$. The definition actually holds for any function in $\mathcal{B}(\mathcal{S} \times \mathcal{A})$.
[2] For simplicity, we here denote the observed reward with lower case r_{t+1}^k instead of $R(s_t^k, a_t^k)$.

result of Equation 4: $o^k = r^k_{t+1} + \gamma \max_{a\in\mathcal{A}} Q_{N-1}(s^k_{t+1}, a)$. In this way, the regression algorithm adopted is trained on TS to learn $Q^*_N(s,a)$. One might think that the algorithm obtains ever better results as $N \to \infty$; however, when performing the estimation of the optimality operator obtained with the regressor, an estimation error is introduced. These errors are cumulated over the iterations; hence, the performance of FQI may decrease with iterations. For this reason, the first parameter analyzed in the model selection explained in Section 5.2, is the number of iterations N.

2.3 Extra Trees

The regression algorithm adopted to perform the feature selection are the extremely randomized trees (or Extra Trees) [18]. This method is based on random forests [5], a combination of decision tree predictors [6], which iteratively produce hyper-planes to split the data space, until the points inside each set of the partition are relatively homogeneous w.r.t. the target. In other words, at each internal node, one attribute is selected to split training samples into two subgroups, maximizing a measure of similarity of the targets among data. If the number of elements belonging to each leaf (i.e., the cardinality of each subgroup) is below a certain threshold, called *min_split*, the branching procedure ends.

A Random Forest regression algorithm is an ensemble of a large number of Decision Trees with a random selection of features at each split. The idea is to use many learners as much uncorrelated as possible, which together can strongly reduce the variance of the estimation. Extra Trees [18] is a variant that further exploits the randomness of forests: when a random subset of features is selected to perform a split, also the cut-points are chosen completely randomly. Then, the split occurs using the best cut-point.

An important property of Extra Trees is that they produce an estimate of the importance of the features in predicting the target. Indeed Decision Tree methods perform splits by determining which one will most effectively discriminate groups of data with similar targets. Therefore, in Random Forests, the importance of a feature is measured as the global improvement (in terms of Gini index, information gain or mean squared error), hence the features can be ranked in order to establish which are the most informative to predict the target.

There are some hyperparameters that must be chosen to perform regression with Extra Trees, such as the number of Trees, which can be used to reduce the variance of the estimation, the number of features considered for each split, affecting the robustness of the predictions, and the minimum sample size *min_split* to perform a split. As explained in Section 5.2, we will fix all hyperparameters (following the suggestions in [18]) but the *min_split*, which is the second term (the first being the number of iterations of FQI) considered to perform model selection.

2.4 Multi-Objective

Learning to trade-off between different (possibly conflicting) goals leads to consider Multi-Objective MDP (MOMDP) [30]. In this setup, the reward R outputs a (bounded) probability distribution over $\mathcal{R}^d$, where d is the number of objectives. This in turn implies that, unlike standard RL, quantities such as the return or the value-function of a policy π are vectors in $\mathcal{R}^d$. As a consequence, it is not possible to define quantities such as $\max_{a\in\mathcal{A}} Q(s,a)$ in general.

A common approach is to reduce the problem to a single objective framework by performing a weighted combination of the different rewards. The weights λ are chosen in the $(d-1)$-dimensional simplex Λ. It is unlikely for a policy to be optimal for every $\lambda \in \Lambda$ since some of the objectives may conflict. Hence the notion of Pareto-optimality: a policy is Pareto-optimal if no other policy can collect better returns on one of the objectives without degrading any other.

A different approach is to learn the Pareto frontier directly, as in Multi-Objective FQI (MOFQI) [9]. The idea in MOFQI is to enlarge the state by the weight vector λ before applying FQI in order to take it into account during the approximation of the action-value function $Q^*(s,\lambda,a)$. Ideally, if the learning is made over a significant set of weights $(\lambda_i)_{1\le i\le n}$, the policy will successfully generalize to unseen values of λ. As the state is enlarged, MOFQI requires more training tuple examples to reach FQI-like performance on a single objective. However, the reward being function of λ, for each tuple in the original training set, n tuples can be constructed for the MOFQI without needing to interact with the environment. Furthermore, as pointed out in [9], MOFQI can be computationally more efficient than running multiple single-objective FQI when the number of such objectives gets significant.

2.5 Risk Aversion

Another far-reaching branch in Reinforcement Learning for finance is risk aversion. Traders are not only interested in returns but also in keeping risk under control. Different risk measures are considered in the literature, as summarized in Section 3. Usually, risk measures are related to the distributions of returns (Variance, CVaR). We are, however, interested in the distribution of rewards: for this reason, we take into consideration the definition of *reward volatility* ν^2_π in [4], which is defined as the variance of the reward under the state-occupancy measure generated by π:

$$\nu^2_\pi := \mathop{\mathbb{E}}_{\substack{s\sim d_{\mu,\pi}\\ a\sim\pi(\cdot|s)}}\left[\left(\mathcal{R}(s,a) - \frac{J_\pi}{1-\gamma}\right)^2\right]. \tag{6}$$

Unlike other risk measures, this term accounts for uncertainties in the short period: the more rewards are oscillating, the higher is the volatility. This is due to the fact that FX prices are very noisy: for this reason, we want to keep under control not only the final return, but also the smoothness of the trajectories. The volatility measure allows us to define the mean-volatility trade-off,

$$\eta_\pi = J_\pi - \lambda\nu^2_\pi, \tag{7}$$

that can be maximized by means of a transformation of the reward:

$$R^\lambda_\pi(s,a) := R(s,a) - \lambda(R(s,a) - J_\pi)^2. \tag{8}$$

Hence, by varying the *volatility aversion* λ, it is possible to obtain different solutions. However, it should be noted that this reward is policy-based, preventing the possibility to use off-policy algorithms, such as FQI, to minimize the volatility. To comply with the off-policy framework, we consider an exponential reward transformation which represents a first-order approximation of R^λ_π

$$\widetilde{R}^\lambda_\pi(s,a) := \frac{1 - e^{-\lambda R(s,a)}}{\lambda}. \tag{9}$$

To make the connection to finance, R^λ_π can be seen as a Constant Relative Risk-Aversion (CRRA) utility function of parameter $\lambda + 1$.

3 RELATED WORKS

In recent years, the financial trading world has been highly interested in AI applications, with a particular focus on Reinforcement Learning [16, 29, 36]. One of the founding works in this direction is [24], where the authors built one of the first Automated Trading Systems (ATS), using Recurrent RL. Starting from this, many other authors proposed different systems, e.g. by using Genetic Algorithms [20, 38] or adding adaptive control layers to govern the investment policies [12]. In particular, the latter is dedicated to FX Trading, as well as [19] and the more recent [14]. In all these works, the goal is the optimization of the Sharpe Ratio, defined as the ratio between the mean and standard deviation of the reward. This is one of the several risk measures considered in a rich RL literature dedicated to Risk Aversion. Another well-known measure is the return variance [33], with its related, nonlinear Bellman equation [13, 27, 32], and the CVaR, which is a quantile of the return distribution [11, 25]. The only value-based approaches in this setting are [35], using the variance of the returns, and [17] for the Sharpe ratio. The distribution of rewards has been considered for risk aversion only recently, for example in [39], where the authors consider the variance of the *per-step* reward, closely related to the reward volatility studied in [4].

Other financial works consider Q-learning for the maximization of the return, as [3, 14, 21] or [22], in which the financial framework as a multi-agent system. Other works address the problem of Optimal Trade Execution, which consists of a different environment in which the goal is to understand the optimal price and volume for a trading request [26].

A completely different stream of literature is dedicated to Multi-objective Sequential Decision Making [30], with a few works considering a value-based approach [8, 9, 37]. At the best of our knowledge, there is no attempt of joint research including both Risk Aversion and Multi-Objective RL, with the goal of building the Pareto Frontier of different financial objectives.

4 PROBLEM FORMULATION

The complete dataset is made of €/$ exchange rate per minute from Monday to Friday, with 1230 observations per day, considering a time window from 00:00 to 20:30.[3, 4] At each time t, we take into account only the *open* price, denoted as p_t. This dataset is used to build an MDP as follows:

State. The state is composed of the last 60 open prices over the last hour, normalized with respect to the first daily price. In order to build a real Markov process, the current state must provide all the necessary information so that it is independent from previous states. Including past data into the current state is a known way to approximate Markovianity. The previous price alone gives too little information, while the previous 60 minutes prices provides a better approximation of the running dynamics. Furthermore, the state also includes the current time and the portfolio position of the previous step: in particular, the current time is in $[0,1]$, denoting the fraction of time remaining until the end of the episode (a trading day). Finally, the portfolio position x_t in $\{-1,0,1\}$ is the only feature from the state which is related to the agent's behavior and is not retrieved from market data. As a consequence, as explained in Section 5.1, it must be generated in order to train FQI models.

Action. The action consists of the portfolio position that the agent wants to keep for the next minute; hence as the previous portfolio position contained in the state s, the set of possible actions $\mathcal{A}$ is $\{-1,0,1\}$; corresponding respectively to short, neutral, or long position. We make the unrealistic assumption of infinite liquidity, which allows the trades selected by the agent to be immediately effective at the current market price. In a real scenario, however, there might be operational delays, which can affect the performances.

Transition Probability. The transition probability $\mathcal{P}(s'|s,a)$ is the probability of moving to state s' given current state-action pair. The action affects only the portfolio feature, hence $x_{t+1} = a_t$; all the other features are exogenous, hence their value is not affected by the action. The current time transition as well as the already seen prices have a deterministic transition, with only one new stochastic addition from market data.

Reward. Given the features vector s_t (which contains the current position x_t), the previous price p_t, and the action taken a_t, the reward can be directly computed once the new price p_t is known. Indeed, it is modeled as follows:

$$r_t = a_t(p_{t+1} - p_t) - f|a_t - x_t|. \tag{10}$$

Basically, the first member of the Equation is related to direct profit, obtained from the price difference and the chosen portfolio position. On the other hand, the second term consists of a fee f that the agent has to pay when he changes her position. We consider for this environment movements of 10^5\$, with fees equal to 2\$. The budget is considered as fixed, so the observed profits are not reinvested nor the losses reduce the budget. The inclusion of fees related to the actions chosen is common in Reinforcement Learning literature for Financial Trading [19, 24].

Discount factor and episode length. Since the last 60 prices are needed to build the state vector, the agent starts with neutral position $x_0 = 0$. The MDP is considered as episodic, where each episode consists of a trading day (at the end of the day the agent closes her position), composed of 1230 steps, in which the rewards are not progressively discounted, hence $\gamma = 1$.

5 EXPERIMENTS

In this section, we describe how we applied FQI to the described setting on real Forex data. The collected data ranged from 2014 to 2019. In order to be more robust, we chose to focus on a training set composed of two consecutive years rather than on a single year. Concretely, we considered the pairs: 2015-2016, 2016-2017, 2017-2018. In order to select the models that better generalize over the following years, for each training dataset, we considered the

[3]Considering the Central European Standard Time - CET.

[4]The remaining time has been excluded for the lower trading volumes, to make the approach more consistent and robust.

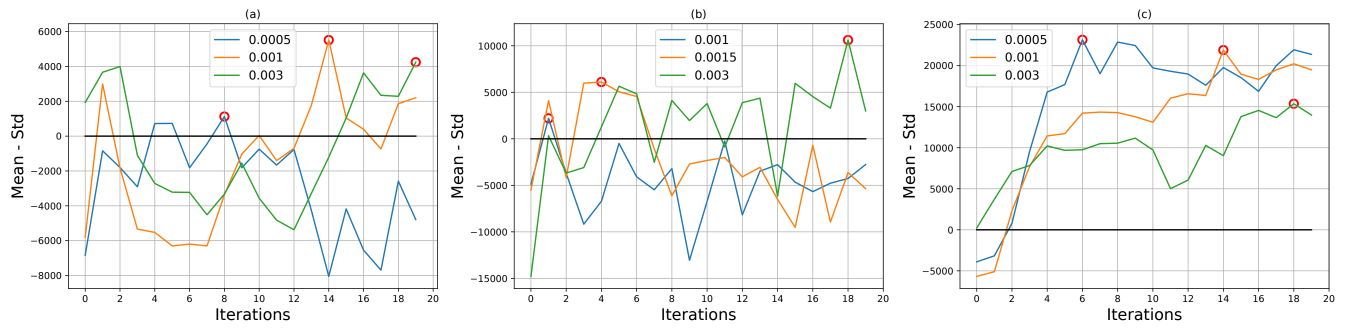

Figure 1: Validation for different datasets: for each of them, lower bound (mean minus standard deviation) is shown and the maximum of each curve is highlighted with a red circle. Different colors correspond to different min-split used. The chosen model corresponds to the higher circle min-split.
The shown plots have been obtained, respectively, validating on years 2014 (Figure (a), with models trained on 2015-2016), 2015 (Figure (c), with models trained on 2016-2017), and 2016 (Figure (c), with models trained on 2017-2018)

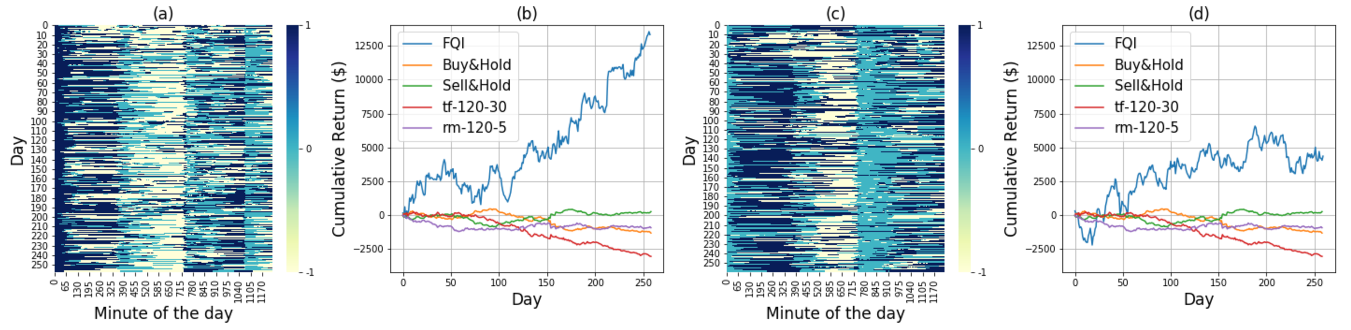

Figure 2: Test evaluation of the performance of two different agents. In the first two figures the agent is trained in 2015-16, validated on 2014 and tested on 2017. In the last two figures, the agent trained in 2016-17, and validated on 2015 is tested on 2018. In Figures (a) and (c) the actions chosen by the agents in test are shown. Each row corresponds to a different business day, and each column is specific for a trading minute (the first column is the first trading minute, for a total of 1230 steps). Figure (b) and (d) show the daily cumulative returns in the test year related to a 100K$ investment, compared with the baselines, where rm-120-30 corresponds to the best active benchmark, with a mean reverting strategy with 120 steps, and 30 steps moving averages.

years before and after the training set: the obtained policies were validated in the former and tested in the latter. This unusual choice was made to take into account the non-stationarity of FX market values: long temporal distances can more easily lead to unseen patterns, thus degrading the performance of the agents.

5.1 Dataset Generation

To assess the pertinence of the time window, a *Feature Importance* procedure has been applied. Using Extra Trees as a predictor, the state features have been used to fit the immediate reward. The results showed the importance of the prices from the last hour to predict the reward (while older prices have negligible impact). From the original sequence of market quotations, the FQI training set (F_{FQI}) is built by composing a series of tuples defined in Equation (5), containing current state, action, reward, next state, with the addition of a Boolean value indicating whether the state is a final state: in our case, only the final quotation of each day is considered a terminal state. As specified in Section 4, the state contains market prices information, and a feature denoting the current portfolio position x_t. To allow the Extra Trees to properly learn the action-value function $Q^*_N(s,a)$, the training set must include a maximum number of scenarios. We treat the market as an uncontrollable component of the state, by assuming that the agents' actions do not modify the market, but only the current portfolio. Therefore, all possible combinations of actions can be generated given a portfolio state, for each of the available data points. Thus, since we only have three possible actions (buy, hold, and sell), the collected market dataset is repeated nine times to include all configurations.

The MOFQI training set (F_{MOFQI}) is built for a given range of risk aversion coefficients $\lambda \in \Lambda_{train}$. It is built from the same tuples in F_{FQI} by adding the risk-aversion coefficient λ to the state and transforming each reward r to $\widetilde{r}^{\lambda}$ according to Equation (9). As a consequence, $|F_{MOFQI}| = |\Lambda_{train}||F_{FQI}|$ and the computational time is drastically increased. To avoid this bottleneck, at the expense of the final performance, the choice has been made to uniformly sample one value in Λ_{train} for each tuple from the FQI dataset. In this way MOFQI training set is built with the same size as the one considered for FQI.

5.2 Model Selection

To choose the best FQI model, we can tune both the hyperparameters related to the regressor and the ones characterizing the general algorithm. As already explained in Section 2.3, Extra Trees have different main parameters: the minimum number of leaves, the minimum number of features to consider, the minimum number of data points necessary for doing a split, and the number of trees. Based on our experience and following what suggested in [18], given a sufficient number of trees (we used 50 trees), it is only necessary then to tune the minimum number of splits, without constraining the features or the number of leaves, in order to regulate model complexity. Typically, a high value for this variable corresponds to build a simpler model, since we are forcing the trees to use a great number of samples to choose the split, hence, excluding more complicated patterns. A small split threshold allows for more complex models, but it increases also the risk of overfitting the in-sample data. As the algorithm produces new iterations, the optimized horizon increases, allowing for longer-term planning, however, the noise contained in the data also propagates, by repeating alternate steps of projection on the approximation space and the Q-function maximization. Therefore, we have also to deal with the trade-off between extending the optimization horizon and propagating errors through iterations.

We trained each dataset using 3 min-splits, and, to check the randomness of the initialization of the trees, we performed 5 different runs for each of them. Since our objective is to find the model that generalizes better over years, we validated the obtained models on the year preceding the training set, in order to test the models on the year following the training. The criterion for selecting the models based on their performance on the validation set was as follows: for each value of min-split, the best iteration for that min-split was selected by choosing the one with the best mean value of the cumulated return, subtracting also its standard deviation. In this way, we try to favor iterations that behave consistently better with different initializations. The latter values are used to select the best min-split value. This procedure is illustrated in Figure 1.

In the MOFQI setting, leveraging the results from risk neutral training, the best value of min-split has been used. In order to compare and choose over the iterations of a given training, the criteria is the *antiutopia hypervolume* [7, 40]. For each risk aversion coefficient $\lambda \in \Lambda_{train}$, the model optimizes a different objective, obtaining a different volatility and average reward. The set of the resulting values can be represented in a graph as in Figure 4c. Ideally, the best model Pareto-dominates the other frontier approximations. To compare Pareto optimality, we compute the overall worst empirical volatility and mean reward, called antiutopia. We then compute the hypervolume with respect to the antiutopia for each of the iterations of the model. The selected best iteration is the one that maximizes this hypervolume on the validation set.

Table 1: Performance obtained testing the selected models and averaging over different runs, with their related standard deviation. Best min-split (as fraction of dataset) and iteration selected from validation are reported.

Training	Validation	Test	Min-split	Iteration	Mean ± Std
2015-16	2014	2017	0.001	14	11218 ± 1785
2016-17	2015	2018	0.003	18	646 ± 2497
2017-18	2016	2019	0.0005	6	5219 ± 1275

5.3 Results

In order to evaluate the selected models, we show the obtained cumulative returns, comparing them with some baselines and providing some insights on the learned behaviors.

Comparison with benchmark strategies. Once selected the best model according to the validation, we need to test it on unseen data. Using the selected min-split and iteration, it is possible to train a new model and use it on the new dataset. For simplicity, we have used the same trained models, whose performance in test are reported in Table 1. We consider a setting in which each action corresponds to a fixed 100K\$ investment; therefore, rewards are rescaled accordingly. As can be seen, apart from 2016-2017, this criterion allows us to choose model that generalizes well in the following years.

Figures 2 and Figure 4 show the results for some of these models on the following year (w.r.t. the training set). Cumulative returns are compared with some benchmark strategies. The first, passive strategies are the *Buy&Hold* and the *Sell&Hold*, consisting in keeping a constant position, respectively long or short. The other strategies considered are active and based on a moving average trend analysis [10, 14]. In particular, the moving average is computed on the previous market prices, using two different time windows. The shorter time window can take into account the previous 5, 15 or 30 minutes, The longer one computes the average of the last 30, 60 or 120 prices. When the shorter moving average is greater than the "long-term" average (relatively long, since it considers a 2 hours window as maximum), then the Trend Following (tf) strategy holds a long position, and vice versa. On the contrary, the Mean Reverting (mr) strategy chooses the opposite action. Among all the possible combinations of time windows and tf/mr strategies, only the best one is presented. Our approach presents, for the selected cases, a steeper improvement in the cumulative rewards. Furthermore, the average cumulative reward for 2015-2016 and 2017-2018 (see Table 1) outperforms the baselines at the end of the year.

Analysis of Temporal Patterns. In Figure 2 and Figure 4a actions chosen in different timesteps (columns) and days (rows) are shown in different colors. It is evident from these plots that there are some temporal patterns that are repeated every day: this is a typical case of *intraday seasonality*, adopted also by human traders, related to specific moments in the trading day. For example, the agent seems more willing to start the day with a long position, which is very often changed when the European market opens for operations. Around step 720 the agent usually chooses to opt for a long position, and this is due to the fact that American Markets are

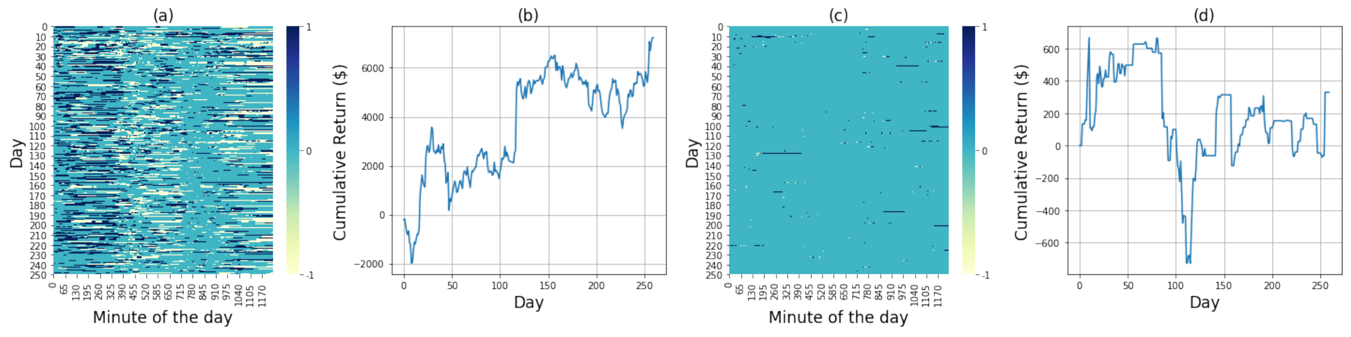

Figure 3: Test on year 2018 of two agents trained on 2016-17 (and validated in 2015) with different risk aversion coefficients. Figure (a) and (b) correspond to an agent acting under a smaller risk aversion with respect to Figure (c) and (d). In Figure (a) and (c) the actions are represented as in Figure 2. They have respectively a standard deviation of 0.57 and 0.09. Figure (b) and (d) show the cumulative return.

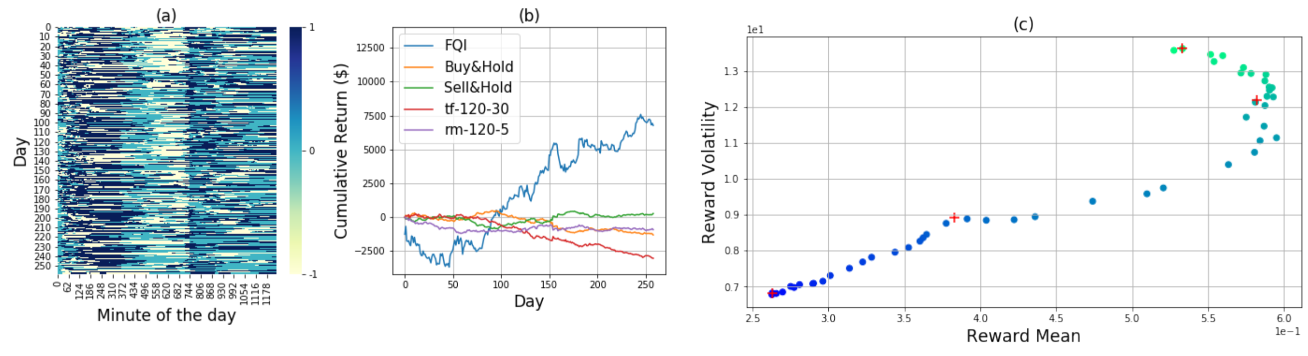

Figure 4: In Figure (a) and (b), the agent trained in 2017-18 and validated on 2016 is tested on 2019. See Figure 2 for an explanation of the plots. Curves rm-120-5 and tf-120-30 correspond to the best active benchmarks: a mean reverting strategy with 120 and 5 steps moving averages, and trend-following, with 120 and 30 steps moving averages.
In Figure (c) the Approximated Pareto Frontier obtained from training on 2016-2017 is showed. The dataset was generated with 4 λ values, while 50 different values have been obtained using the learned Q-function. The 4 original coefficients are marked by red cross.

opening. Finally, the opening of Asian markets (around step 1080) can be detected by looking only at the actions in Figure 2a, and not in 2c or 4a. This might mean that its influence on €/$ is milder. This behavior is intriguing, since these patterns were automatically learned, without any "expert" providing previous information. In any case, other, more complex patterns are leveraged by the agent and need further studies, such as the fact that the models obtained at later iterations have a more complex behavior.

Effect of Risk-Aversion. MOFQI has been trained with a set of 4 different risk-aversion coefficients, tuned manually to produce diversified risk management strategies. In our setting, it corresponds to values of *lambda* ranging from 10 to 500. The algorithm is then tested on 50 new value of *lambda* in the same range. Figure 4 shows the Pareto frontier for these new values on the training set. The results clearly highlight the mean-volatility trade-off. Remarkably, the more risk-averse the agent, the closer it is to the optimal Pareto frontier. A possible explanation is that learning in a highly risk-averse environment is easier as portfolio positions are held longer, resulting in a simplified optimization problem. This last statement is supported by Figure 3 which represents the actions and cumulative return in test for two opposite risk profiles. Unsurprisingly, the most risk-averse agent mostly adopts the flat action.

6 CONCLUSIONS

The implementation of Efficient High-frequency Automated Trading Systems is one of the most interesting challenges for Artificial Intelligence in the financial field, especially in the Forex Exchange Market, where prices are heavily affected by random noise. The particular structure of the problem allowed us to exploit a semi-generative configuration, in which we were able to produce all the possible outcomes of the controllable part of the system since they can be deterministically computed. This has allowed us to have a dataset that was rich enough to apply FQI to the FX trading context. The results, obtained on three different temporal frameworks, show that this algorithm allows developing artificial traders able to develop effective strategies. This is due to the fact that the regression performed by the Extra Trees tries to detect interesting patterns that can be exploited to optimize the return; some simple ones, as the opening/closing time for world markets, are automatically learned; some other may be more complicated, similar to the technical analysis adopted by human traders. These results show that this kind of technique can be useful also for human traders to discover novel patterns that can be exploited to find novel trading opportunities. An interesting addition to our approach was the generalization to a Multi-Objective setting, where the agent faces a trade-off between maximizing the return and the task of keeping the uncertainty under control, measured as the volatility of the collected rewards. At the best of our knowledge, this is the first attempt of considering Risk Aversion in Reinforcement Learning as a Multi-Objective MDP. We showed that the algorithm was able to learn a complete spectrum of trade-offs between risk and expected performance, generalizing the training carried out on a few risk-aversion values. Many future developments are possible, both from an algorithmic and application point of view. The approach could be extended to the online setting, e.g., by using DQN [23] instead of FQI. Another possible research line regards the development of an exact FQI version for the Mean-Volatility objective. Finally, from the application point of view, the problem could be generalized to consider multi-currency portfolios and more realistic scenarios, e.g., taking explicitly into account operational delays.

ACKNOWLEDGEMENTS

The research was conducted under a cooperative agreement between ISI Foundation, Intesa Sanpaolo IMI Corporate & Investment Banking Division and Intesa Sanpaolo Innovation Center.

REFERENCES

[1] Stefan Banach. 1922. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. math* 3, 1 (1922), 133–181.

[2] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680* (2019).

[3] Francesco Bertoluzzo and Marco Corazza. 2012. Testing different reinforcement learning configurations for financial trading: Introduction and applications. *Procedia Economics and Finance* 3 (2012), 68–77.

[4] Lorenzo Bisi, Luca Sabbioni, Edoardo Vittori, Matteo Papini, and Marcello Restelli. 2020. Risk-Averse Trust Region Optimization for Reward-Volatility Reduction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20.* 4583–4589. https://doi.org/10.24963/ijcai.2020/632 Special Track on AI in FinTech.

[5] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[6] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and regression trees.* CRC press.

[7] Yongtao Cao, Byran J Smucker, and Timothy J Robinson. 2015. On using the hypervolume indicator to compare Pareto fronts: Applications to multi-criteria optimal experimental design. *Journal of Statistical Planning and Inference* 160 (2015), 60–74.

[8] Andrea Castelletti, Francesca Pianosi, and Marcello Restelli. 2011. Multi-objective Fitted Q-Iteration: Pareto frontier approximation in one single run. In *2011 International Conference on Networking, Sensing and Control.* IEEE, 260–265.

[9] Andrea Castelletti, Francesca Pianosi, and Marcello Restelli. 2012. Tree-based fitted Q-iteration for multi-objective Markov decision problems. In *The 2012 International Joint Conference on Neural Networks (IJCNN).* IEEE, 1–8.

[10] Ernie Chan. 2009. *Quantitative trading: how to build your own algorithmic trading business.* Vol. 430. John Wiley & Sons.

[11] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. 2017. Risk-constrained reinforcement learning with percentile risk criteria. *JMLR* 18, 1 (2017), 6070–6120.

[12] Michael AH Dempster and Vasco Leemans. 2006. An automated FX trading system using adaptive reinforcement learning. *Expert Systems with Applications* 30, 3 (2006), 543–552.

[13] Dotan Di Castro, Aviv Tamar, and Shie Mannor. 2012. Policy Gradients with Variance Related Risk Criteria. *ICML* 1 (06 2012).

[14] Damien Ernst et al. 2020. *An Application of Deep Reinforcement Learning to Algorithmic Trading.* Technical Report. arXiv. org.

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

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

[17] Xiu Gao and Laiwan Chan. 2000. An algorithm for trading and portfolio management using Q-learning and sharpe ratio maximization. In *Proceedings of the international conference on neural information processing.* 832–837.

[18] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning* 63, 1 (2006), 3–42.

[19] 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.

[20] A Hryshko and T Downs. 2004. System for foreign exchange trading using genetic algorithms and reinforcement learning. *International journal of systems science* 35, 13-14 (2004), 763–774.

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

[22] Jae Won Lee, Jonghun Park, O Jangmin, Jongwoo Lee, and Euyseok Hong. 2007. A multiagent approach to q-learning for daily stock trading. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 37, 6 (2007), 864–877.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

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

[25] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. 2010. Nonparametric Return Distribution Approximation for Reinforcement Learning. In *ICML.*

[26] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. 2006. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning.* 673–680.

[27] L. A. Prashanth and Mohammad Ghavamzadeh. 2014. Actor-critic algorithms for risk-sensitive reinforcement learning. *arXiv preprint arXiv:1403.6530* (2014).

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

[29] Reuters. 2019. *How to train your machine: JPMorgan FX algos learn to trade better.* https://www.reuters.com/article/us-jpm-trading-machines/how-to-train-your-machine-jpmorgan-fx-algos-learn-to-trade-better-idUSKCN1S61JG

[30] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* 48 (2013), 67–113.

[31] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[32] Matthew J. Sobel. 1982. The variance of discounted Markov decision processes. *Journal of Applied Probability* 19, 4 (1982), 794–802.

[33] Marc C Steinbach. 2001. Markowitz revisited: Mean-variance models in financial portfolio analysis. *SIAM review* 43, 1 (2001), 31–85.

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

[35] Aviv Tamar, Dotan Di Castro, and Shie Mannor. 2016. Learning the variance of the reward-to-go. *The Journal of Machine Learning Research* 17, 1 (2016), 361–396.

[36] Financial Times. 2017. *JPMorgan develops robot to execute trades.* https://www.ft.com/content/16b8ffb6-7161-11e7-aca6-c6bd07df1a3c

[37] Marco A Wiering, Maikel Withagen, and Mădălina M Drugan. 2014. Model-based multi-objective reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL).* IEEE, 1–6.

[38] Jin Zhang and Dietmar Maringer. 2016. Using a genetic algorithm to improve recurrent reinforcement learning for equity trading. *Computational Economics* 47, 4 (2016), 551–567.

[39] Shangtong Zhang, Bo Liu, and Shimon Whiteson. 2020. Per-Step Reward: A New Perspective for Risk-Averse Reinforcement Learning. *arXiv preprint arXiv:2004.10888* (2020).

[40] Eckart Zitzler and Lothar Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation* 3, 4 (1999), 257–271.