POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Exploiting Continuous Action Space in Fx Trading with Reinforcement Learning

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Vito Alessandro Monaco

Advisor: Prof. Marcello Restelli

Co-advisor: Lorenzo Bisi, Luca Sabbioni

Academic year: 2022-2023

1. Introduction

The Foreign *Exchange market* (also called Forex Market) is considered the largest and most liquid market in the world, thanks to these qualities and great availability, it attracts a growing number of potential investors every day. Through the improvement of computational resources and the growing knowledge and popularity of Machine Learning (ML), artificial intelligence (AI) techniques have become a tool capable of developing powerful automatic trading systems. These techniques offer strategies able to generate large profits and in the same way, mitigate losses due to the market's inherent stochasticity. One of the possible approaches for automatic trading consists of modelling the Forex market as a Markov Decision Process (MDP, [12]) and solving it through the application of Reinforcement Learning (RL) techniques. This thesis is driven by the necessity to explore and expand current reinforcement learning (RL) approaches, enabling them to interact with a more realistic environment. This environment should integrate the use of continuous actions and be able to process orders of different magnitudes, allowing to use transaction costs that depend on the size of each order. Additionally, the objective is to integrate risk-averse approaches that can effectively manage market uncertainty. In order to do this, we explored the use of an actor-critic architecture to combine the benefits of value-based and policy-based approaches. The contributions provided in this thesis can be summarized as follows: we developed the Fitted Natural Actor-Critic algorithm (FNAC, [7]) from scratch and, based on our knowledge, we applied it for the first time in a real trading environment. Moreover, we modelled the interaction between the agent and the environment through the use of a continuous action space. This allowed us to model transaction costs in a more realistic way, reflecting the real costs brokers charge for large orders. Finally, we used the concept of persistence and integrated risk-aversion approaches to induce more effective behaviours in our models.

2. Reinforcement Learning

Reinforcement Learning (RL) is one of the branches of ML, and can be defined as a computational trial and error approach for training agents to learn an optimal behaviour to achieve a specific goal. The learning process is driven by the sequential interaction between an agent and an external environment, modelled

through the MDP framework. An MDP is a stochastic dynamical process defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \mu \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, P is the state-transition probability function defined as $P(s'|s, a)$, R is a reward function, $\gamma \in [0, 1]$ is a discount factor for future rewards and finally μ is the distribution of the initial state. The chosen action is retrieved through the policy π, which assigns a probability distribution over the action space for each state. In this work, we focus on stationary Markovian policies, where the distribution depends only on the current state and not on the history of the previous states. In general, the goal of an agent is to maximize the return collected through the interaction with the environment. When the agent follows a policy π, we can define the State-Value Function $V^\pi(s)$ that represents the expected return that an agent can achieve starting from a particular state. Similarly, the Action-Value function $Q^\pi(s, a)$ represents the expected return starting from a particular state s and taking an action a under a policy π. Solving an RL task involves finding the optimal policy that ensures the maximum expected return. This process can be performed using techniques derived from the *Bellman Optimality Equation*, which exploits the Dynamic Programming approach for decomposing the problem between the policy evaluation and the policy improvement tasks. When the full knowledge of the MDP is not fully available, techniques such as *Monte-Carlo Learning* and *Time-Difference Learning* (TD) techniques can be employed. TD methods are model-free approaches, which can learn directly from episodes of experience and update the new estimates using the previously computed one (using the so-called *bootstrapping* approach). Unfortunately, in real-world problems, it is common to encounter continuous or extremely large state-action spaces. As a result, there might be no available information on the vast majority of the state-action space. These issues highlight the necessity to generalize the experience gathered from the collected samples encountered in the past. For this reason, it is necessary to use function approximation approaches in order to approximate the value function and the agent's policy. Usually, the value function approximator relies on the minimization of the mean squared value error, obtained by the mean squared error between the approximate value function and the true value function. Instead, the policy approximator commonly utilizes the *Policy Gradient theorem* [16] in order to update the parameters and optimize a parameterized policy. One possible approach for implementing this update is by employing the natural gradient. Unlike vanilla gradients, natural gradients follow the steepest ascent direction in a space defined by the *Fisher Information Matrix*. This approach enables stable and efficient updates by opting for a more direct path to the optimal solution.

3. Actor-Critic Methods

Actor-Critic (AC) methods try to combine the best characteristics of value-based methods, which try to learn the optimal value function, and policy-based methods, which try to learn the optimal policy following the gradient in parameter space, using two components called *Actor* and *Critic*. The Actor represents the policy approximation, used for selecting the actions performed by the agents, and the Critic represent the value function approximator which, following a TD approach, evaluates the actor's choices to determine if the produced outcome is better or worse than expected.

3.1. Fitted Natural Actor-Critic

Fitted Natural Actor-Critic is an extension of the Natural Actor-Critic algorithm [11]: it modifies the TD-based critic to allow the use of general value function approximators, using the natural gradient to update the policy of the actor. The algorithm uses a dataset $\mathcal{D}$ of samples obtained from the environment: during each iteration, the critic component processes the samples to compute an approximation $\widehat{V}_\theta$ related to the policy π_θ. This approximation is then used to estimate an approximate form of the advantage function $\widehat{A}_\theta$ by employing a linear function approximation with compatible basis functions, which is used to perform the natural policy gradient update of the θ parameters of the actor component.

4. Related Works

One of the initially significant successes in applying RL to trading was the application of *Re-*

current Reinforcement Learning. In [10] RRL was applied to the USD/GBP currency training the model by maximizing a variant of the Sharpe Ratio. Afterwards, with the increasing popularity of *Deep Learning*, Deep Q-Network [9] algorithm was used in various settings, e.g. with OHLC Forex data and in different currencies, also obtaining surprising results thanks to its excellent generalization properties [15]. In order to control and deal with uncertainties in the market, Bisi et al. [1] employ the Fitted Q-Iteration (FQI, [5]) algorithm in the Forex market using a Multi-Objective formulation for keeping the risk of noisy profits under control and increment the risk-aversion behaviour of the algorithm. Lately, Riva et al. [13] use FQI integrating the concept of persistence for tuning the control frequency and handling in a better way the signal-to-noise ratio. In any case, the works in the financial context that is of most interest to us are the policy-based and actor-critic ones, in fact, unlike the previous ones, these methods are able to obtain a policy that supports actions in a continuous state, with which it is possible to integrate risk-averse approaches exploring the gradient nature of the actor component. Bisi et al. [2] incorporated the *mean-volatility* objective into the TRPO [14] algorithm to guarantee risk-averse behaviour throughout the learning process. This model was tested using data from S&P 500 index, by considering buy, sell, and flat as possible actions and utilizing daily price data from the 1980s to 2019, obtaining performance domination with regard to the reward-volatility and expected return frontier compared to the classical TRPO algorithm.

5. Problem Formulation

5.1. Forex Trading Enviroment

For modelling the Forex trading environment we use a EUR/USD dataset retrieved through the website HistData.com [6].[1] We formalize the problem as an episodic task, where an episode corresponds with a unique market opening day consisting of a total of 600 minutes between 8:00 a.m. and 6:00 p.m. (CET).

[1] We transform the spreads in the dataset by a factor equal to 0.5 to align the data with real-world markets.

5.1.1. State Formulation

To ensure the Markovian property we have included data from past observations in each state. More precisely, we have included the last 45 mid-price variations with one minute frequency, the values of the spread, the day of the week and the minute of the specific trading day. Moreover, we included the current portfolio allocation derived from the previously performed action, in order to take into account the possible costs incurred with the next actions. In the discrete action space setting this portfolio allocation is expressed by $x \in \{-1, 0, -1\}$ that corresponds to the actions *Short*, *Flat*, and *Long*. Instead in the continuous action space setting this feature is a real number $x \in \mathbb{R}$ such that $-1 \leq x \leq 1$, that expresses the percentual of the maximum traded amount available in the current portfolio, where in our formulation is equal to 1e5$.

5.1.2. Action Formulation

The action defines the portfolio allocation that the agent wants to keep in the next state and follows the same formulation of the allocation feature state. In this case, the sign represents the type of position, while the absolute value expresses the percentage of the subsequent position with respect to the maximum amount available in the portfolio. Furthermore, we decided to close all positions at the end of the trading day, resulting to have only action 0 (Flat) at the last minute of the day.

5.1.3. Reward Formulation

The reward received by the agent after performing its action is defined by the following equation:

$$r_t = a_t(p_{t+1} - p_t) - c_t|a_t - x_t|, \tag{1}$$

with

$$c_t = f + \frac{1}{2}\sigma_t. \tag{2}$$

where p_t is the current exchange rate at time t, $t + 1$ is the next timestep, with a frequency equal to one minute, x_t is the current portfolio allocation, f is a fixed transaction cost and σ_t is the current value of the bid-ask spread at time t. This formula can be divided into two parts: the first represents gain (or loss) due to the exchange rate variation, and the latest is the total cost charged by the FX brokers due to the allocation

changes performed by the agent. Thanks to the use of continuous actions the policy is able to execute orders with a variable size: this allows us to consider more realistic transaction costs that depend on the size of the order placed. We have formalized this concept by using a spread correction factor obtained by an auxiliary positive function $g : \mathcal{X} \to \mathcal{Y}, \mathcal{Y} \in \mathbb{R}^+$ where $\mathcal{X}$ defines the size of the performed order. By setting the maximum size allocation to $z = 1e5\$$, we can reflect real market conditions by adopting a spread step function defined as:

$$g(x) = \begin{cases} 1 & 0 \leq x < \frac{3}{4}z \\ \frac{11}{4} & x > \frac{5}{4}z \\ \frac{7}{4} & \text{otherwise} \end{cases} \tag{3}$$

5.2. Action Persistence

In the financial context, the frequency of interaction with the market is a fundamental aspect. In fact, higher frequencies of interaction allow better control opportunities with higher potential profits. On the other hand, the observation noise and the sample complexity might become too high to be able to learn profitable behaviour. In order to efficiently modify the control frequency and to obtain a better signal-to-noise ratio, we use the concept of *action-persistence* [8]. Finally, to be compliant with the action-persistence definition in which each action is repeated by a number of times equal to k, we have modified the general reward formula in Equation (1) to consider the next persisted exchange price p_{t+k}.

5.2.1. Algorithm Implementation

In order to create a model that can efficiently handle the previous Forex MDP formulation, we implemented from scratch the FNAC Algorithm described in Section 3.1. For its implementation, we use three main components: the main critic, for which we mainly use the *XGBoost* [4] regressor; the secondary critic, in charge of computing the w parameters for the natural gradient update, for which we adopt a Ridge Regressor. The final component is the actor, for which we take into account a *Feed-Forward Neural Network* (FFNN). The actor architecture built differs depending on the action space used: in the discrete actions space setting we used an FFNN with three hidden layers, with the final output layer composed of three neurons each corresponding to each available action; thanks to a softmax function, we normalize the output to obtain the probabilities from which each action will be selected. Instead in the continuous action space setting, we use an FFNN with four hidden layers, with the final output layer composed of two neurons, respectively corresponding to the mean μ and the standard deviation σ of a normal distribution. In order to restrict the output sample retrieved from the distribution and to ensure an efficient back-propagation, we substitute the normal distribution with the Truncated Normal distribution restricted in the range $x \in [-1, 1]$.

5.2.2. Risk-Aversion approaches

In order to keep the risk of potential losses under control, we extend the previous risk-neutral formulation modifying the objective function in order to optimize two different risk measures: the Reward Conditional Value-at-Risk (RCVaR, [3]) that is a reward-based version of the classic Conditional Value-at-Risk (CVaR) and the Mean-Volatility risk measure objective [2]. While the former risk measure is aimed at optimizing the worst-case scenarios, the latter is focused on a trade-off between the maximization of the expected return and the minimization of the variance of the observed rewards. In order to consider the RCVaR, we can use an equivalent MDP formulation using the following transformed reward:

$$\widetilde{R}(s, a) = \rho - \frac{1}{\alpha} \left(R(s, a) - \rho \right)^- , \tag{4}$$

where ρ is a risk aversion hyperparameter used to define the level of risky rewards to discard. Regarding the Mean-Volatility optimization, we use instead the following reward transformation:

$$R_\pi^\lambda(s, a) = R(s, a) - \lambda (R(s, a) - J_\pi)^2, \tag{5}$$

where λ is the risk-aversion level parameter. Instead, J_π is the normalized expected return, calculated by using a Monte-Carlo procedure under the current policy π.

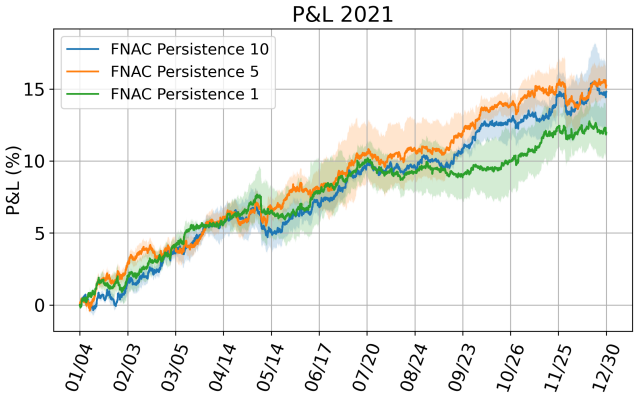
6. Experimental Results

In this section, we initially describe the model selection procedures used to train our algorithm.

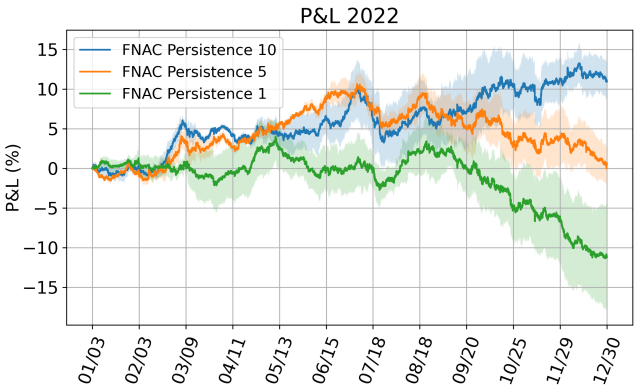
Next, we will provide the results obtained using Forex market data, showing the outcomes of the application of the fee function described in Equation 3 and of the risk-averse approaches. For all of the results shown, we will assume the fixed transaction costs f equal to 0.

6.1. Model Selection

Due to the actor-critic architecture of the FNAC algorithm, we used two different algorithms for implementing the primary critic and the actor components, with one additional critic represented by a linear regressor (with Ridge regularization) to compute the advantage estimation. As mentioned above, we have chosen to use the XGBoost algorithm for the value function approximation: this choice is motivated by its excellent capability to handle diverse and large datasets, along with its high prediction performance and efficient computational processing. Being an optimized ensemble implementation that combines gradient boosting and bagging approaches, its hyperparameters are mainly related to these latest techniques. Once we found an adequate number of trees to ensure a fair trade-off between the computational complexity and the generalization capabilities, we performed selection procedures only among a subset of all the available hyperparameters. This subset includes the learning rate and the *boosting rounds* related to the gradient boosting approach and the *min-child weight* that indicates the minimum sum of the weights of the leaf observations to perform the node splitting. In addition, to better control overfitting, we used an early stopping procedure to limit boosting rounds. Regarding the critic of the advantage function approximation, we use a Ridge Regressor, mainly tuning the coefficient for regulating the L2 term. Finally, since we used an FFNN for the actor component, we tuned the learning rate and the number of parameters of the network.
For the learning process, we decide to split the available data, which includes the observation between the years 2018 to 2022, into four parts: We used the years 2018-2019 for the training set, 2020 as the validation year to perform the early stopping on the XGBoost regressor, 2021 as validation for the actor component performance and finally the year 2022 as test set.



(a) FNAC Discrete validation year 2021

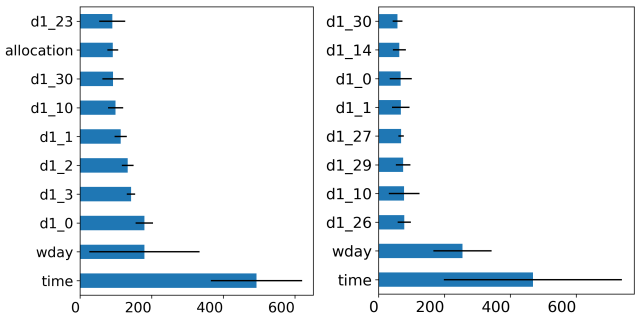


(b) FNAC Discrete test year 2022

Figure 1: Plot of the percentage P&L in the validation year 2021 and test year 2022, using FNAC discrete with different values of action persistence.

6.2. Discrete Setting

Firstly, we investigated the performance of FNAC in the discrete action space using the persistence values 1,5 and 10 (on a one-minute basis, this means that the actions are respectively performed every 1, 5 or 10 minutes). The results show how persistence helps the generalisation capabilities of the model. Indeed, while obtaining positive performances in the validation year 2021 (Figure 1a), models with lower persistence performed much worse in the test year 2022 (Figure 1b). To have a better interpretation of the results, we then analyzed the feature importance (Figure 2) and the heatmaps of the allocations chosen by the agents. We have observed how the most important features for the approximation of the value function are the temporal ones, useful for identifying temporal patterns within the trading day.



(a) Persistence 5 (b) Persistence 10

Figure 2: Bar Plot feature importance of the first 10 features using the number of split as a measure.

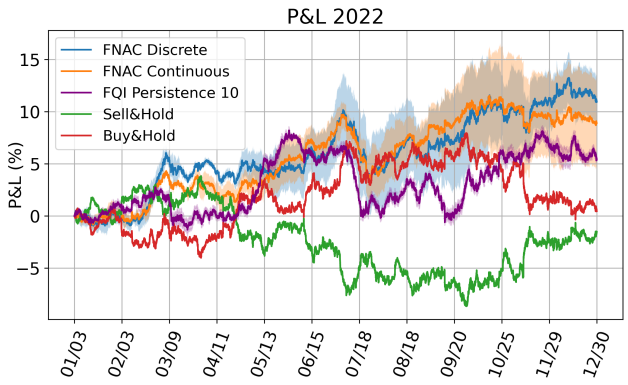


Figure 3: Plot of the percentage P&L in the test year 2022, using FNAC discrete and continuous models with persistence 10.

6.3. Continuous Setting

In the continuous action space settings, we analyzed only the results with persistence 10: this is because the results obtained with smaller values of persistence showed the generalization problems present in the discrete case. In Figure 3 we compare the results obtained in the test year 2022 from models trained with continuous and discrete action spaces, also including the results of FQI and from the baselines Buy&Hold and Sell&Hold. As we expected, due to the difficulty to handle a continuous actions space, we generally obtained slightly worse performances than the discrete case. Precisely, in the last 4 months of 2022, we noticed higher drawdowns and a significant rise in the standard deviation of the cumulative return. In any case, the training process showed greater stability than the discrete one, indicating a greater predisposition to decrease the *model-risk*.

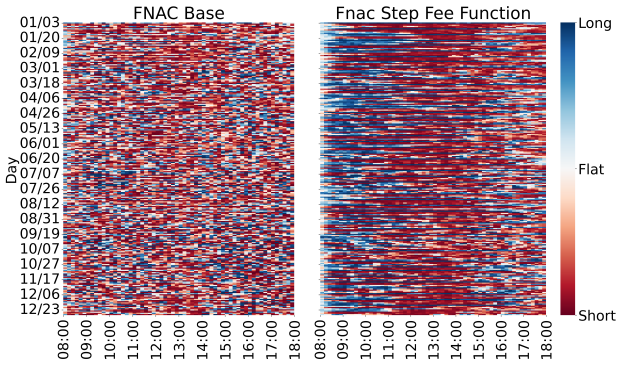


Figure 4: Heatmaps allocations of the base and the step fee function FNAC models in the test year 2022.

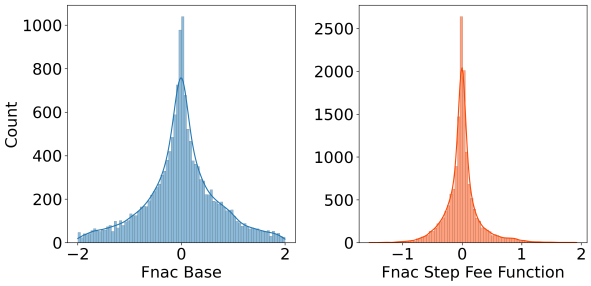
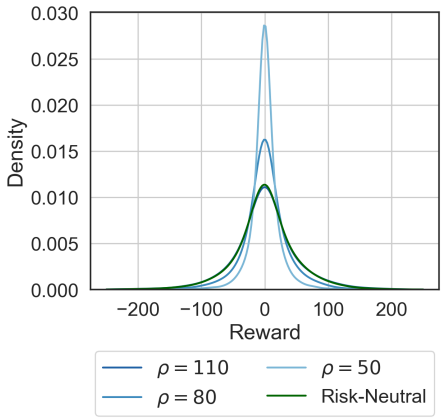


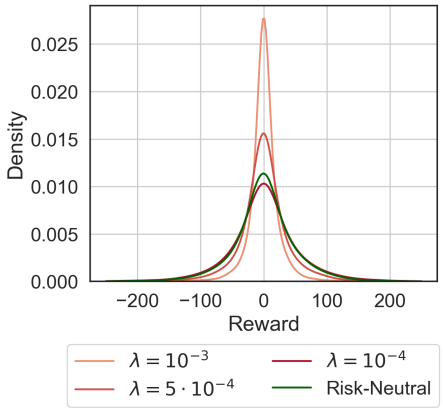
Figure 5: Histogram of the allocations changes in the test year 2022 using the base and the Step Fee function FNAC models.

6.3.1. Step Fee Function Setting

Using the step function in Equation (3), we study the learned models in a scenario with size-dependent transaction costs. The results obtained showed how the agent correctly learned to manage the size of the orders. From Figure 4 it can be observed the presence of continuous colour stripes, this provides a representation of how the model has acquired the capacity to incrementally modify the allocations, effectively decreasing the impact of high transaction costs. Additionally, this behaviour can be better observed in Figure 5, which shows the difference of the chosen allocations between two subsequent time steps, demonstrating how the model with the size-dependent function chooses to execute smaller orders. However, the performances follow the same trend compared to the base case, showing also here a noticeable and worse drop at the end of the test year.



(a) RCVaR



(b) Mean-Volatility

Figure 6: Kernel density estimation of the rewards obtained by the risk-averse models and risk-neutral model in the test year 2022.

6.3.2. RCVaR Setting

As previously mentioned, to use RCVaR as a measure of risk we used the transformation of rewards (Equation 4). We analyzed the distribution of the rewards obtained by the risk-neutral models to find proper values for the hyperparameters ρ; using the risk levels discovered, we trained and tested the models in the test year 2022. Through the performances obtained, we have verified how ρ modifies the actions chosen by the models. In fact, even if the RCVaR optimization is not fully effective, lower values of ρ induce a more conservative behaviour, determining which risky rewards to discard and decreasing the variability of the rewards obtained (Figure 6a).

6.3.3. Mean-Volatility Setting

Similarly to the RCVaR approach, we initially examine the mean-volatility of the risk-neutral results. Through this study, we find a range of potential values for λ around 10^{-3}. Analyzing the performances obtained by the model trained with this objective function, we have noticed that at high λ values, the model tends to have a more risk-averse behaviour. Unfortunately, the non-stationarity makes the performance in 2022 much worse than the risk-neutral model, especially for high values of the risk-aversion parameter. Finally, analyzing the distribution of the reward obtained by the risk-averse model, it is possible to notice how by imposing high λ values it is possible to decrease the variability of the rewards obtained (Figure 6b).

7. Conclusions

Reinforcement learning offers exciting opportunities for financial applications. The Forex market, thanks to its enormous liquidity and flexibility, is one of the places to design potentially profitable algorithms. Unfortunately, due to its non-stationarity and volatility, it presents difficult challenges to face. By implementing the FNAC algorithm, we modelled an agent capable of interacting with the market using continuous actions. Through the integration of a size-dependent fee function, we were able to train an agent capable of limiting order volume. Furthermore, through the use of risk measures, the trained agents developed risk-averse behaviour, decreasing the variance of the rewards obtained. Although positive performances were achieved, the non-stationarity of the data caused performance to deteriorate in the last months of the test year 2022. This can be attributed to the model's limited ability to generalize, which is partly due to the difficulty of creating optimal policies from different initializations of the policy and the use of training data with different trends from the test year. Several approaches can be considered to limit the impact of non-stationarity. One of these is the use of multi-expert learning algorithms, through which market drifts can be managed in a better way thanks to the experience of different models. Further possible optimizations result from the computational effort required for the training of the different components of FNAC. This can be done

by performing a more in-depth tuning of the hyperparameters used, which is however made difficult by the computational time required. Finally, the differentiable nature of the policy opens up possibilities for investigating transfer learning techniques on currencies other than EUR/USD. This is especially relevant for currencies with lower liquidity, where finding successful strategies can be more challenging.

References

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

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

[3] Massimiliano Bonetti, Lorenzo Bisi, and Marcello Restelli. Risk-averse optimization of reward-based coherent risk measures. *Artificial Intelligence*, page 103845, 2023.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

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

[6] Histdata.com. `https://www.histdata.com/`. Accessed: 2023-06-02.

[7] Francisco S Melo and Manuel Lopes. Fitted natural actor-critic: A new algorithm for continuous state-action mdps. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II 19*, pages 66–81. Springer, 2008.

[8] Alberto Maria Metelli, Flavio Mazzolini, Lorenzo Bisi, Luca Sabbioni, and Marcello Restelli. Control frequency adaptation via action persistence in batch reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 6862–6873. PMLR, 2020.

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

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

[11] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[12] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

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

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

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

[16] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.