POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Non-Stationary RL for Forex Trading with Automatic Market Regime Clustering

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Francesco Sammarco

Advisor: Prof. Restelli Marcello

Co-advisors: Pierre Liotet, Lorenzo Bisi, Luca Sabbioni, Antonio Riva

Academic year: 2021-2022

1. Introduction

The *Foreign Exchange Market* (Forex) is a global decentralized or over-the-counter (OTC) market for the trading of currencies. In terms of trading volume–the amount of security traded in a period of time–it is the largest market in the world. It implies that it offers lower transactions costs and higher volumes than other markets. This, in turn, explains why the Forex market draws the attention of researches to develop effective algorithms for *automated trading*. However, due to its dynamical nature, it represents a major challenge. Shifts in the market trends are abundant, due to unexpected exogenous events like Covid-19 pandemic or lower scale phenomenon dictated by social and economical reasons. In other words, Forex market is a highly **non-stationary** environment.
The main goal of this thesis is to deal with the non-stationarity which characterises the Forex market. More specifically, the idea is to find periods of time with common rates patterns and exploit them to develop more profitable strategies.
To this end, a novel approach is proposed, which should be capable of identifying and predicting financial regimes by exploiting trading strategies executed by reinforcement learning based algorithms in the past, and using this model to achieve higher return in the future.
The innovative idea introduced by this thesis is to achieve the proposed objectives by leveraging past *strategies* rather than the history of exchange rate evolution. To the best of authors' knowledge, this represent the first kind of approach which tries to deal with non-stationarity through reinforcement learning in this way.

2. Background

2.1. Reinforcement Learning

The basic model used to represent reinforcement (RL) learning problems is the discrete-time *Markov Decision Process* (MDP). An MDP is a tuple $< \mathcal{S}, \mathcal{P}, \mathcal{A}, \mathcal{R}, \gamma, \mu, >$, where $\mathcal{S}$ is the space of the possible state. $\mathcal{A}$ is the space of the possible actions that an agent may execute. $\mathcal{P}(\cdot|s,a)$ is a Markovian transition model which describes the probability of the next state s' given the transition pair (s,a). $\mathcal{R}$ describes the distribution which characterises the reward $R(s,a)$ that an agent receives as a consequence of executing action a in state s. Here, $\mathcal{R}$ is considered bounded. $\gamma \in [0,1)$ is called the *discount fac-*

tor, whose role is to quantify the advantage of acquiring immediate reward with respect to the future rewards. In this thesis case, $\gamma = 1$. Finally, μ is the distribution of the initial state of the environment.
A policy π defines a distribution over the action space $\mathcal{A}$ given some state. It defines how the agent selects its actions and the main objective of RL is to learn a policy in the space of all policies Π which maximizes the expected value of the *return*. The return is defined as $G_\tau = \sum_{t=1}^{T} \gamma^t \mathcal{R}(s_t, a_t)$.
We also define, for some $s \in \mathcal{S}$ and $a \in \mathcal{A}$, the action-value function–or Q-function:

$$Q_\pi(s,a) \doteq \mathop{\mathbb{E}}_{\substack{s_{t+1}\sim\mathcal{P}(\cdot|s,a) \\ a_{t+1}\sim\pi(\cdot|s_{t+1})}} \left[G_\tau | s_0 = s, a_0 = a \right], \quad (1)$$

and its optimal value $Q^*(s,a) \doteq \sup_{\pi\in\Pi} Q_\pi(s,a)$. This value can be found by iteratively applying the *Bellman optimality operator* $\mathcal{T}^*$,

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

whose fixed point is $\lim_{k\longrightarrow\infty} \mathcal{T}^*Q = Q^*$.

2.2. Fitted-Q Iteration

Fitted-Q Iteration (FQI) ([5]) is a model-free, off-policy, offline algorithm which employs the Bellman operator and *supervised learning* techniques to learn an approximation of Q^*. From a dataset of past transitions $(s_t^k, a_t^k, r_{t+1}^k, s_{t+1}^k)$, FQI learns the Q-function Q_n as an approximation to Equation (2) where the maximum is made over the previous iterate of the action value function $\max_{a'\in\mathcal{A}} Q_{n-1}(s_{t+1}^k, a')$. This approximation introduces an error which may increase as the number of iteration grows; thus, it is important to properly tune and select the number of iterations N.

2.3. XGBoost

XGBoost ([4]) is a supervised algorithm which may be employed as a regressor for Q_n in FQI. It is based on *Gradient boosting trees* (GBT): namely, it leverages various decision trees whose role is to iteratively enhance the approximation of a target y. From the current prediction $\hat{y} = \sum_{i=1}^{m} \hat{y}_i$, a new weak learner h_{m+1} is added to by minimizing the error $y - \hat{y} + \hat{y}_{m+1}$. GBT performs the minimization step with gradient descent. XGBoost adds a regularisation term to solve overfitting by coping with excessive confidence of the regressor.

2.4. Clustering

Clustering is an unsupervised learning approach whose aim is to partition a set of instances based on their similarity. The output of a clustering algorithm is a partition of the training set $\mathcal{X}$. The two following steps may be crucial for clustering.
Selecting the number of clusters D. The elbow method, which consists in identifying the lowest number from which the intra-cluster variance start decreasing slowly; *dedrogram* visualisation by applying a threshold based on the graph; maximisation of the *silhouette score*, a measure used to assess how similar an element is to its own cluster (cohesion) compared to other clusters (separation).
Selection of the algorithm. It can be done by measuring the silhouette score and visually inspecting the clusters to assess their goodness with respect to the problem considered. In case a ground truth is available, exstrinsic measure like the *adjusted rand index* (ARI) and the *V-Measure* can be used. The first assesses the agreement of two clusters results, while the seconds estimates the correctness of the clustering algorithm by combining a measure of how many times an unique label is contained in an unique cluster and viceversa.
These two processes are extremely delicate and are mostly based on domain knowledge. However, the described methodologies may support the decision in case this type of knowledge is not accessible.

2.5. Non-Stationarity

The concept of non-stationarity (NS) is mostly linked to time-series and processes. Intuitively, stationarity means that the distribution generating the series is not changing. If such condition does not hold, it is said to be *non-stationary*.
A NS may follow two pattern: *smooth*, if the changes to the underlying distribution are gradual; *abrupt*, if the properties of the underlying distribution changes suddenly and considerable effect may be perceived.
In RL, NS is mostly related to changes occur-

ring in the transition probability $\mathcal{P}$ or the reward distribution $\mathcal{R}$, and in rare cases to the action space $\mathcal{A}$ and observations (in this case, non-stationarity is induced by an adversary). In any case, this means that these elements now depend on time.
To model these changes, mainly three frameworks exist.
Switching modelling. It supposes the environment modeled as a finite or infinite number of MPDs $\mathfrak{M} = \{\mathcal{M}_1, \mathcal{M}_\in, ...\}$ whose elements are considered stationary. The NS trait is instead expressed by the way in which the environment switches from an MDP to the other. These MDPs are called *tasks*.
Drifting modelling. It assumes the existence of an unique MDP which model the environment, whose elements are subject to NS smooth bounded variations.
Adversary modelling. It supposes that an adversary is influencing the environment, thus causing NS behaviours.
Approaches to solve this problem are mainly divided in two classes: *passives* or *robust* approaches, which try to act in the best way possible ignoring the existence of the NS behaviour; *active* approaches, which acknowledge the existence of the NS behaviour and try to model it in order to predict or cope with its possible effects.

3. Problem Formulation

In this thesis, the Forex is modelled as a multi-task problem composed of an unknown number of regimes $\{\mathcal{M}_i\}_{i\geq 1}$, where each $\mathcal{M}_i$ may be described as MDPs following [2] formalisation.
Horizon. The Forex is modelled as a finite-horizon problem, i.e. it is divided in K episodes of H time-steps, each time-step being a minute. Each episode coincides with a market day, starting at 8:00 CET and ending at 18:00 CET. In this thesis, it is assumed that each episode k belongs to a single regime $\mathcal{M}_i$. $\mathcal{D}$ will be used to indicate the set of all the possible day indexes (i.e., the episodes).
State. The state is divided in two parts $s_h^k = (\phi_h^k, x_h^k)$. The first, ϕ_h^k, contains: the last 60 normalised variations between consecutive minutes of the exchange rate; the current time of the day; the spread; the encoding of the day (Monday = 1, ..., Friday = 5). The second, x_h^k, contains the previous allocation. At the start and the end of the day, the allocation is set to 0 so as to avoid overnight fees. The wealth of the agent is not present in the state as he may only trade a fixed quantity of $100K\$$.
Action Space. The agent may perform 3 actions: long, short and flat. They are represented with the number 1, -1 and 0. Assuming infinite liquidity, the action is always executed. Possible order delay is not taken into account.
Transition Probability. The transition probability $\mathcal{P}$ is simplified by assuming that only x_h^k is affected by the agent's actions. Instead, ϕ_h^k evolves according to historic market data.
Reward. The reward is modelled as:

$$r_h^k = a_h^k(p_h^k - p_{h-1}^k) - c_h^k|a_h^k - x_h^k| \qquad (3)$$

where $p_{hh\in\mathbb{H}}^k$ is the series of prices and c_h^k is a value dependent on the transaction fee f and the spread.

4. Related Works

4.1. Reinforcement Learning for Finance

In recent years, there have been many successful examples of RL applied to financial trading. Particularly relevant works for this thesis are [2, 12]. In [2], *Multi-Objective FQI* (MOFQI)–a modification of FQI–is used for learning to balance profitable and risk-averse trading strategies; its effectiveness is proven against basic trading strategies, such as *buy&hold* and *sell&hold*. In [12], the concept of *action persistence* [9] is used inside the FQI framework in order to tune the control frequency and modify the signal-to-noise ratio. Interestingly, [12] leverage this modified FQI to solve the Forex trading problem in a multi-asset environment, showing its ability to outperform the classic FQI on various currency pairs.

4.2. Non-Stationary Reinforcement Learning

Literature counts many examples that try to deal with NS in several contexts. Regarding the switching modelling literature, the one adopted in this thesis, [3] is particularly relevant. In this work, the authors propose the *Prognosticator*, a model-free RL approach which addresses smooth NS in rewards and transition distributions by forecasting future performances of a

policy leveraging historical data through importance sampling. The approach has been applied in this thesis to prove that accounting for these NS behaviours is not sufficient in Forex. Other approaches instead deal with NS by employing other methods, such as meta-learning ([11]) or factorisation methods ([6]).
Yet, only few works try to deal with NS in Finance with an active approach. On the *portfolio optimisation* problem, [7] model NS with the switching framework and leverage both meta-learning and evolutionary algorithms. For EUR/GBP and AUD/NZD currency pairs, [10] propose to automatically learn how to preprocess financial price series to remove NS traits and cast the problem back to classic RL. Finally, for EUR/USD, [8] employ a hyper-policy to deals with the NS by selecting which policy is best for each time-step, using past experience through importance sampling.
These last works are particularly important as they underline the necessity of new representations for financial series and new methods to address NS.

5. Proposed Approach

The proposed approach, called *Regime-Forecaster*, aims at identifying NS changes in the market by tracking the performance of different traders. In this case, the traders are RL agents trained via FQI based on the choice of a different set of hyper-parameters. If these performances are then compared, it is possible to assess if the market has changed with respect to previous days.
The Regime-Forecaster builds a non-stationary tracking framework which works at a high-level. By training a high number of policies, this approach gathers and simulates many different behaviours of traders on past data. These policies are used to describe days and measure the similarity between them. The similarity measure then drives a clustering algorithm which groups days supposed to belong to the same market regime. This new information is then leveraged both at train time to learn policies specialised on each regime, and at test time to build a prediction model of the next probable cluster regime.
The proposed approach may be roughly divided in five steps:

1. train and select a diverse set of policies;
2. from the previous policies' trading patterns, cluster days in market regimes;
3. train and test regime-specialised policies;
4. build a predictor of the next cluster;
5. predict next cluster and use proper regime-specialised policy.

5.1. Obtaining Representative Policies

The first step is to train and select policies for clustering. By training FQI with various hyper-parameters, one obtains an *ensemble* $\tilde{\Pi}$ of different trading strategies. The idea of using many policies is to provide the clustering algorithm with a complete representation of the day's trading patterns. In order to reduce the dimensionality of the features of a day, a *feature selection* step is applied to eliminate both similar policies–which would therefore add no information–and redundant features. To find similar policies, a clustering on $\tilde{P}i$ is performed. Concretely, for two policies π_i and π_j, the following similarity is computed,

- *Kullback-Divergence* (KD):

$$D_{KL}(\pi_i \| \pi_j) \doteq \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{a \in \mathcal{A}} \pi_i(a|h_k) \log \left(\frac{\pi_i(a|h_k)}{\pi_j(h_k)} \right)$$

  Where $\pi_i(h_k)$ returns the array $\{\pi_i(a|h_k), a \in \mathcal{A}\}$. Since it is not a symmetrical measure, the *Jensen–Shannon divergence* is instead used:

$$d_{KL}(\pi_i, \pi_j) = \frac{1}{KH} \frac{D_{KL}(\pi_i \| \pi_j) + D_{KL}(\pi_j \| \pi_i)}{2}$$

- *Total Variation Distance* (TV):

$$d_{TV}(\pi_i, \pi_j) = \frac{1}{KH} \sum_{k=1}^{K} \sum_{h=1}^{H} \max_{a \in \mathcal{A}} |\pi_i(a|h_k) - \pi_j(a|h_k)|$$

similarities are compared using various clustering algorithm, choosing the best one based on silhouette score metric and accordance to optimal cluster number chosen for each algorithm. The output of this step is a set $\bar{\Pi}$ containing the *representative policies*.

5.2. Clustering Days

Each policy $\pi_n \in \bar{\Pi}$ is used to obtain the *policy feature* Λ. A **policy feature** is a multi-

dimensional feature describing a specific day. Three possible representations may be chosen:

- the return G_k, which has lower dimensionality with respect to the other representations but may assume very different scale in the values of each days;
- the reward time series $\{r_h^k\}_{h\in\mathbb{N}}$, which retains the sequentiality trait of a day, is continuous and expressive of a trader profit evolution;
- the action time series $\{a_h^k\}_{h\in\mathbb{N}}$, which is able to fully represent the behaviour of a trader in terms of portfolio allocation, but is categorical and therefore hard to manage.

In the following, for policies in a set $\hat{\Pi} \subset \Pi$ and indices $\mathcal{G} = \{k \in \mathbb{N}\}$, $\Lambda(\hat{\Pi}, \mathcal{G})$ will indicate the set of features which represents the days with index in $\mathcal{G}$ obtained by policies in $\hat{\Pi}$.
Policies features $\Lambda(\bar{\Pi}, \mathcal{D}^{(train)})$ will be used to train a clustering model Υ. The latter is then applied to $\mathcal{D}^{(train)}$, $\mathcal{D}^{(val)}$ and $\mathcal{D}^{(test)}$ to obtain, through policy features Λ, the clusters on each set. The product of this step are three sets of form $\mathcal{C} = \{C_d | d \in [0, D]\}$, one for each set of $\mathcal{D}$.

5.3. Training Specialised Policies

In this phase, a set of *specialised policies* $\Pi_D = \{\pi_d\}_{0\leq d\leq D}$ is obtained by training each policy π_d on days in cluster $C_d^{(train)}$ and validated on cluster $C_d^{(val)}$). Additionally, a *general policy* π_* is trained on $\mathcal{D}^{(train)}$ and validated on $\mathcal{D}^{(val)}$. The idea of the specialised policies is to exploit the identified regime to develop ad hoc strategies that should, in theory, outperform the general policy π_* when used on days supposed to belong to such regime (though not being explicitly trained on it).
To test this assumption, each policy in π_d and policy π_* are tested on the cluster $C_d^{(test)}$. The result of the test is the return $G^{(d)}_{C_f^{(test)}}$ for the specialised policies and $G^{(*)}_{C_f^{(test)}}$ for the general policies, i.e. the total return obtained using policy π_d (π_*) on cluster $C_f^{(test)}$. If the return of the specialised policy is greater than the one of the general, it means that the clustering algorithm has identified truthful market regimes and the policy π_d was able to exploit such knowledge, enhancing the return.

5.4. Building the predictor

If the previous step has been successful, it is mandatory to obtain a prediction model Ψ able to predict the cluster of the next day $k+1$ given the history $k+1$.
To accomplish this objective, the clusters in $\mathcal{C}$ are used as a sequence of labels, building a set of form (k, d), where k is the day index and d is the cluster. Then, a classifier is trained to predict them. One of the most challenging problem is choosing a proper feature set $\boldsymbol{x}_{k+1}$ which can be used to represent each day $k+1$.
The choice of the features is a critical step since they must contain enough information in order to give accurate prediction on future clusters while being restricted enough to avoid noisy inputs and over-fitting. Obviously, it is not possible to use $\Lambda(\bar{\Pi}, k+1)$, as this cannot be accessed at the start of the day. For this reason, a set with feature available before trading in day $k+1$ and features of n days before $k+1$ were considered. In particular, the predictors considers

- the *day features*, encoding of the type of the day, the month value and the day value;
- the *price features*, containing: the open price for $k+1$ and its difference with the previous days; the open, high, low, mean and close price for n days before $k+1$ and their difference with previous days;
- four special policy features obtained by executing the policies in $\bar{\Pi}$ and Π_D on the 120 minutes before the start of trading day $k+1$ and 60 minutes after the start of the same day;
- the two policy feature obtained from $\bar{\Pi}$ and Π_D policies execution on the previous day k;
- the *Cboe Volatility Index* (VIX), which is a real-time index that measures the volatility in the Forex market;
- a window of n previous clusters. Combinations of these features have been considered, as to properly tune the predictor.

An important aspect is that an optimal predictor is not needed: in trading the final objective is to obtain an high return. Since policy π_d may perform well also on clusters $\hat{d} \neq d$, a *cost-sensitive* classifier is built, i.e. a classifier penalising each misclassification with a cost ci, j equal to the loss in performance of employing policy π_i on cluster j, which can be approximated with the quantity $G^{(i)}_{C_i^{(train)}} - G^{(i)}_{C_j^{(train)}}$.
The cost-sensitive loss selected for the task is

the one proposed in [1], which is expressed as:

$$L(M, z, S(\boldsymbol{x})) = \log\left(1 + \sum_{i=1}^{D} m_{z,i} e^{S_i(\boldsymbol{x}) - S_z(\boldsymbol{x})}\right) \tag{4}$$

where z is the true cluster, $m_{z,i} = c_{z,i}$ and $S(x) = \{S_0(x), \ldots, S_D(x)\}$ corresponds to the score given by the classifier to each class for the data point $\boldsymbol{x}$. This loss ensures the *guess-aversion* property, i.e. it encourages correct classification over random guess in multi-class classification.

5.5. Predicting Next Cluster

Once a suitable and reliable classifier Ψ is available, it can be used to infer the next day cluster d. When this is done, it is then possible to use π_d to obtain a theoretically higher return. To assess the goodness of the results, it is possible to measure the *expected return gain*, defined as the sum of the difference of the expected return from the specialised policies under the predictor Ψ and the return obtained via the general policy, i.e.:

$$\Delta G^{ex} = \sum_{k=1}^{K} \sum_{d=1}^{D} \left(\lambda_{d_k} G_k^{(d_k)}\right) - G_k^{(*)} \tag{5}$$

where λ_{d_k} is the weight assigned by Ψ to the cluster d in day k. Alternatively, also the *effective return gain* can be calculated, defined in the same way but replacing $\lambda_{d_k} = 0$ if d is the predicted cluster of k and $\lambda_{d_k} = 1$ if it is not.

6. Experiments

The proposed approach was experimented on two datasets: the EUR/USD dataset, containing the exchange rate history from 2017 to 2021, which is characterised by a yearly non-stationary behaviour and intra-day unit roots, but whose mean and variance may be considered almost stationary during a single market day; a synthetic dataset, generated using a *Vasicek model* whose mean can change every day based on a Markov chain model with a transition matrix as the one shown in Figure 1, where the letters indicate different values of the mean. The matrix was created giving high probability to remain in the same cluster, as a way to consider the empirical observation that regimes usually last more than one day.

Figure 1: Transition matrix used for generating the synthetic dataset.

6.1. Results of the EUR-USD dataset

For the EUR/USD dataset, the set $\tilde{\Pi}$ was formed with FQI by fixing the parameters of XGBoost to default values but the minimum child weight m_{min}, whose values were chosen in the interval $\{k \cdot 5000\}_{1 \leq k \leq 8}$, and the persistence values, whose possible values were $\{5, 10\}$. The policy were trained on 3 different sets, i.e. the one formed by taking two exchange rates year history from 2017, 2018 and 2019 and using the other one as a validation set, in order to ensure variability of the policy features Λ.
The feature selection step left 5 policies, selected by applying the TV distance with a hierarchical clustering, as the KL was not able to differentiate between different policies. The number of policies was selected using the dedrogram provided by hierarchical clustering.
The clustering of the day was executed by trying many different combinations of clustering methods on the action and reward representation, excluding the return one for the scale problem. Based on the silhouette score and on visualisation, the chosen algorithm was K-Means with the reward representation.
The specialised policies and the general policies were both trained on the clusters formed applying the clustering model Υ on the policy features of the training and the validation set. To hypertune the policies, persistence values in the set $\{2, 3, 5, 6, 8, 9, 10\}$ were tested, alongside m_{min} values in the set $\{k \cdot 2500\}_{1 \leq k \leq 11}$. Once the policies have been trained, they were tested against

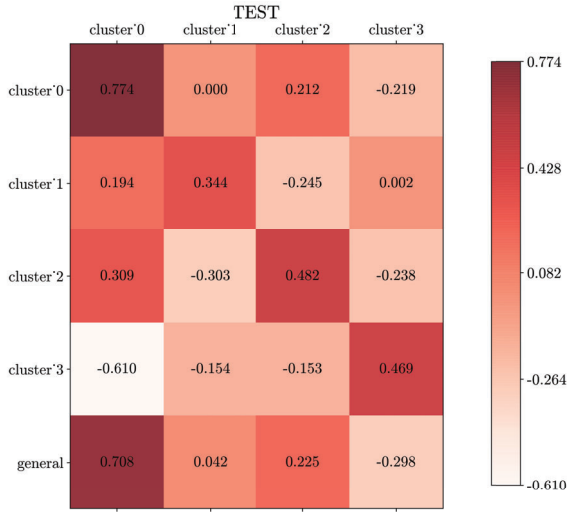

Figure 2: Test return matrix for EUR/USD.

the general policy by calculating the return obtained on the cluster of the test set, thus calculating $G^{(d)}_{C_f^{(test)}}$ on each pair of clusters index (d, f), and additionally $G^{(*)}_{C_f^{(test)}}$ for the general policy. Then this results where normalised by applying the formula:

$$\bar{G}^{(d)}_{C_f} = 250 \frac{G^{(d)}_{C_f}}{10^5 |C_f|} \tag{6}$$

to consider both the imbalance of the clusters and to turn this value in a percentage of gain over the initial invested sum. The result of this process is the *test return matrix*, which is portrayed in Figure 2.

As it is possible to see, the diagonal presents higher values compared to the last row, which means that the specialised policies obtained an higher return compared to the general policy π_*, confirming that the clustering methods found meaningful market regimes within the dataset.

At this point, the predictor was built. In a first attempt, a Markov chain model with various order was used, trained on the sequence of clusters. This was done in order to discover a pattern in the succession of clusters. However, experiments turned out to give unsatisfying results, with an accuracy of $\approx 31\%$, thus it was discarded.

Similar results were obtained by using the XGBoost classifier, fixing all hyperparameters but $m_{\min}$ and the max depth of the tree. In particular, it was tested both using the multi-categorical crossentropy (CCE) loss, maximising the accuracy of the classifier, and with the cost-sensitive loss shown in Equation (4).

6.2. Results of the Synthetic dataset

After the failure on the EUR/USD dataset, experiments continued by considering the synthetic dataset. Since the task was easier, it was decided to consider a different formalisation of the problem, reducing the state to the pair of the current price p^k_h and the current portfolio allocation x^k_h. This kind of representation was the optimal one, as the day were inherently stationary. This allowed to retain process memory. In this case, the set $\tilde{\Pi}$ was formed with FQI choosing persistence values $\{5, 10\}$ and $m_{\min}$ in the set $\{k \cdot 2500\}_{4 \leq k \leq 9}$. On this dataset, KL gave a more meaningful clustering of the policies, obtaining a silhouette score of ≈ 0.313 versus the ≈ 0.11 of TV. Therefore, it was chosen as the distance measure with hierarchical clustering, leaving 5 policies

For the clustering of the days, the action representation proved to gave more meaningful results. By using the Gower distance and hierarchical clustering, a silhouette score of ≈ 0.46 was reached, which outperformed any other combination of distance measure and clustering algorithm. In this case, extrinsic measures were available: Gower turned out to be the best in terms of ARI and V-Measure, confirming the correctness of the approach. However, compared to other clustering algorithms, Gower did not identify 4 regimes but 3, although with a better correspondence with the real clusters.

Specialised policies and general policy were trained choosing values of persistence in the set $\{2, 5, 10\}$ and minimum child weight in the set $\{k \cdot 2500\}_{1 \leq k \leq 15}$. Testing the policies against the general policy created the test return matrix in Figure 3.

Once again, it is possible to assess the effectiveness of the method in identifying meaningful regimes, though it is unknown whether these regimes can fully refer to the generated regimes or are related to properties of the Vasicek model. Building the predictor turned out to provide better results than EUR/USD dataset. The most useful feature turned out to be the day opening price. Using a Markov chain model, an ex-

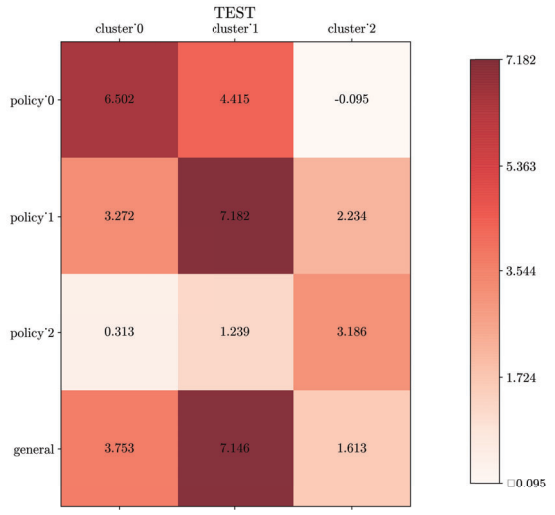

Figure 3: Test return matrix for synthetic dataset.

tremely good result of $\approx 74.5\%$ accuracy was reached with an order 3. Since the problem was easier, for the XGBoost model simpler features were used. Namely, a window of past labels and a window of open, close, high, low and mid price were used. XGBoost scored $\approx 80.2\%$ accuracy with CCE loss and 5.36 mean penalty with guess-averse loss. The positive results convinced to try out the last phase of the approach.
The Markov Chain and the XGBoost models were used as the predictor on the test set, and both the effective and expected return gain were calculated over 312 days. The normalised version of the return are reported in Table 1.

Ψ	$\Delta G^{ex}(\%)$	$\Delta G^{ef}(\%)$
Markov (1)	28.6	83.7
Markov (2)	31.6	83.7
Markov (3)	23.3	77.1
XGBoost (CCE)	50.8	118.4
XGBoost (Guess)	**110.0**	**122.8**
Real (Inf)	183.87	183.87
Real	434.62	434.62
Random	-14.74	-28.31

Table 1: Performance of *Regime-Forecaster* for the synthetic dataset.

XGBoost classifier outperformed any other predictor, obtaining 110.0% expected profit with respect to the general policy with guess-averse loss. The classifier weren't able to reach a score near the one obtainable from perfect, real cluster knowledge (the *Real* attribute in the table), but it is quite close to the result obtained from the availability of a perfect predictor of the next inferred cluster (the *Real (Inf)* attribute in the table). Nevertheless, the high scores mean that the experiment was successful, proving that on a simple task *Regime-Forecaster* is able to identify and exploit market regimes.

7. Conclusions

Based on the results, it is clear that *Regime-Forecaster* wasn't able to recognise the dynamics of regime switching in the EUR/USD dataset. This kind of result may be the consequence of various factors, related to the various steps of the approach: clustering methods may have failed to catch temporal patterns due to inexpressive policies or inadequate clustering algorithms for the task; important information for the regime clustering task were removed from FQI pre-processing on the financial series, and this process may have deprived them of their memory; the features chosen for the XGBoost predictor could be not informative enough.
However, the results on the synthetic dataset clearly show that the methodology of *Regime-Forecaster* has some relevance and effectiveness. Although it was a simpler task, the fact that it was possible to enhance the return with this technique shows that clustering methods are able to catch regimes within a financial time series, given the proper settings.
Therefore, there are various possibilities for future researches. Different RL algorithms may be tested, in order to assess if they are able to provide more informative policies feature; different clustering methods may be tried, which are able to take into account the temporal dimension when clustering is performed, thus enhancing predictability of the resulting clusters; more sophisticated features or classifier may be tested, able to exploit the sequential structure of the regimes, as *recurrent neural networks*.
In any case, *Regime-Forecaster* firmly stands as a milestone, being the first active NS approaches to tackle regime identification via high-level rep-

resentation, and may be used as a first step towards research in this promising direction.

8. Acknowledgements

Un grazie al Professor Restelli, per avermi dato la possibilità di condurre questo lavoro di tesi e per i preziosi consigli forniti durante il mio percorso. Un grazie ai miei correlatori Pierre, Lorenzo, Luca e Antonio, il cui aiuto e confronto è stato fondamentale per questa chiusura di percorso. Un grazie alla mia famiglia, per esserci sempre stata ed avermi sostenuto. Specialmente un grazie a Lida, riferimento e scoglio a cui tenermi nelle difficoltà Un grazie ai miei amici, all'11S, ad Annalaura, Mauro e Domenico, a cui voglio un bene dell'anima.

References

[1] Oscar Beijbom, Mohammad Saberian, David Kriegman, and Nuno Vasconcelos. Guess-averse loss functions for cost-sensitive multiclass boosting. In *International Conference on Machine Learning*, pages 586–594. PMLR, 2014.

[2] 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.

[3] Yash Chandak, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip Thomas. Optimizing for the future in non-stationary mdps. In *International Conference on Machine Learning*, pages 1414–1425. PMLR, 2020.

[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*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

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

[6] Fan Feng, Biwei Huang, Kun Zhang, and Sara Magliacane. Factored adaptation for non-stationary reinforcement learning. *arXiv preprint arXiv:2203.16582*, 2022.

[7] Myoung Hoon Ha, Seung-geun Chi, Sangyeop Lee, Yujin Cha, and Moon Byung-Ro. Evolutionary meta reinforcement learning for portfolio optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 964–972, 2021.

[8] Pierre Liotet, Francesco Vidaich, Alberto Maria Metelli, and Marcello Restelli. Lifelong hyper-policy optimization with multiple importance sampling regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7525–7533, 2022.

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

[10] Angelos Nalmpantis, Nikolaos Passalis, Avraam Tsantekidis, and Anastasios Tefas. Improving deep reinforcement learning for financial trading using deep adaptive group-based normalization. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2021.

[11] Riccardo Poiani, Andrea Tirinzoni, and Marcello Restelli. Meta-reinforcement learning by tracking task non-stationarity. *arXiv preprint arXiv:2105.08834*, 2021.

[12] 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.