Chaire LUSIS - CentraleSupélec

Non-Markovian Dynamics for Automated Trading

Final Year Capstone Project

Seong Woo AHN, Rodolphe NONCLERCQ
InfoNum 2023

Supervision:

Fabrice Popineau	LISN/CentraleSupélec	Fabrice Daniel	LUSIS
Arpad Rimmel	LISN/CentraleSupélec	Xavier Farchetto	LUSIS
Bich-Liên Doan	LISN/CentraleSupélec	Timon Fugier	LUSIS
		Pierre Lourdelet	LUSIS

Contents

List of Figures

Chapter

1 Introduction

Automated trading has become an important area of research in the financial industry ([1], [2], [3]). The use of deep learning models in trading has gained much attention due to their ability to process and analyze large amounts of financial data quickly and provide valuable insights in making effective investment decisions, such as in the work done by Booth et al. in [1] where Random Forest models are trained on financial market data to learn seasonality effects and empirical regularities and predict price returns.

Reinforcement learning (RL) is a widely used approach in this field, which involves training agents to make decisions based on their actions and the feedback received from the environment. The approach we propose in this study builds upon the existing literature on RL-based asset allocation. Several studies have investigated the use of RL agents for trading in financial markets, such as in the article written by Zhang et al. in [4] where a portfolio selection method using deep reinforcement learning is proposed. One of the main challenges in RL-based asset allocation is the limited availability of data at a daily scale. For example, stock market data is typically available only at the daily frequency, which may not be sufficient to capture the dynamics and patterns of the market accurately. This can lead to poor performance of the RL agents in asset allocation, which can have significant financial implications. To address this issue, the work done by Yuan et al. in [5] introduced a data augmentation framework by using minute-candle data (open, high, low, close) to improve RL based agents that can beat the market in Sharpe ratio.

In our work, we propose a novel approach to synthetically generate more data using the Mori Zwanzig formalism and thus overcome the limited availability of data. This formalism provides a way to project a signal onto a Hilbert space of observables and extract Markovian and Non-markovian components from the signal. By projecting the base signal onto a space of other stock prices, we create alternative versions of the original signal. These synthetic signals could potentially capture unseen interactions between market signals, therefore enriching the baseline dataset. Hypothetically, the resulting increase in the amount of data helps to improve the performance of the RL agent's asset allocation. This is because the synthetic data provides the RL agent with a broader and more diverse set of market dynamics to learn from. Additionally, the Mori Zwanzig formalism allows us to extract useful information from the original signal that may not be evident at the daily scale.

The Mori Zwanzig formalism has been widely used in the field of statistical mechanics to analyze complex systems. It has been used in areas such as thermodynamics (Chu et al. [6]) and chemical physics (Li et al. [7]). However, to the best of our knowledge, this is the first study to apply the Mori Zwanzig formalism to financial market data.

In summary, this study proposes a novel approach to improve the performance of RL-based asset allocation by synthetically generating more data using the Mori Zwanzig formalism. The approach we propose is based on a mathematically sound formalism and has the potential to significantly improve the performance of RL agents in trading financial markets.

Chapter

2 Our model

2.1 Data augmentation

In this section, we propose a synthetic data augmentation method for deep reinforcement learning in financial trading, based on the Mori-Zwanzig projection. The Mori-Zwanzig projection is a powerful mathematical technique that can be seen as a generalization of the Koopman learning framework.

There exists two ways to describe a dynamical system. In the first method [8, 9], the system is characterized by a collection of physical-space state variables. For instance it could be the component of the velocity field at a a specific location in a fluid dynamical system or the position of an atom in a many-particle system. In the second, Koopman [10, 11] proposes to characterize a system by a collection of observables which are functions of the physical-space variables. It could be for example a component of the total angular momentum of a subset of all atoms in a particle system, or the locally averaged density in a fluid dynamical system. In Koopman's representation, observables evolve linearly in an infinite-dimensional Hilbert space $\mathcal{H}$ which is composed of all the possible observables. To derive a closed form solution in the Koopman theory, one has to identify a set of variables whose dynamics are invariant in a subspace which is linearly spanned by the set of the observables. [12] shows how the Mori-Zwanzig formalism provides a way to close the dynamics through the use of projection operators, and can be therefore considered as a natural generalization of the Koopman description of the dynamical system.

A major result of the Mori-Zwanzig formalism is the identification of parameters for the Generalized Langevin Equation (GLE). In [12], the authors provide an alternative derivation of the GLE which is based on the Koopman eigenfunctions and provides a more transparent representation of the Mori-Zwanzig operators. Given a function of observables $\mathbf{g}(t)$, we have the GLE:

$$\frac{d}{dt}\mathbf{g}(t) = \mathbf{M} \cdot \mathbf{g}(t) - \int_0^t \mathbf{K}(t-s) \cdot \mathbf{g}(s) ds + \mathbf{F}(t)$$

$\mathbf{M}$ is referred as the *Markov transition matrix*, $\mathbf{K}(s)$ as the *memory kernel*, and $\mathbf{F}(t)$ as the *orthogonal dynamics*. Intuitively, this equation tells us that the evolutionary equations of our chosen observables always depends on their (1) instaneous configuration, (2) their past history, and (3) an external "driving force" which depends on the initial configurations in the orthogonal space. However, given that we do not have access to the orthogonal space, its initial configurations are under-resolved and we often refer to this last component as *noise* due to its resemblance of a Langevin noise. From this equation we can extract the relevant operators. The operators $\mathbf{M}$ quantifies the interactions within the group of the selected observables, whereas the memory kernel $\mathbf{K}$ combines the effects of the interactions (with the under-resolved observables, and the interactions between the under-resolved observables themselves). This gives way to Mori's projection which projects our function of observables $\mathbf{g}(t)$ onto a subspace $\mathcal{H}_g$ which is linearly spanned by the set of selected observables $Span(\{g_i\}_{i=1}^M)$.

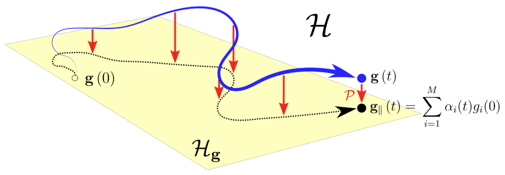

$\mathcal{H}$
$\mathbf{g}(0)$
$\mathbf{g}(t)$
$\mathcal{P}$
$\mathbf{g}_{\parallel}(t) = \sum_{i=1}^{M} \alpha_i(t) g_i(0)$
$\mathcal{H}_{\mathbf{g}}$

Figure 2.1: Schematic diagram of the Mori-Zwanzig formalism.. Mori's projection operator projects $\mathbf{g}(t)$ into $\mathcal{H}_g$

[12] proposes a data-driven learning algorithm which can be used to extract the different GLE operators based on time-series data. By using this Mori-Zwanzig algorithm, we were able to generate synthetic financial data that can be used to train deep reinforcement learning models with improved performance and generalization capabilities.

2.2 RL agent

The reinforcement learning model we chose to use in our experimentations is the SAC model [13]. Soft Actor Critic is a Reinforcement learning model used to optimizes a stochastic policy with continuous actions and observations. SAC uses a part of DDPG [14] and TD3 [15] approaches and adds some improvements. The particularity of SAC is the optimization of the entropy.

Reward

We decide to use a specific reward which varies depending on the gamma parameter used in Q-learning. The gamma parameter is a discount factor that determines the importance of future rewards in the learning process. It is a value between 0 and 1, with higher values indicating that future rewards are more important than immediate rewards. When Q-learning is applied, the agent uses the gamma parameter to calculate the expected value of the future rewards at each state-action pair. This value is then used to update the Q-values, which represent the estimated value of taking a certain action in a given state. A higher gamma value encourages the agent to take actions that lead to long-term rewards, while a lower gamma value makes the agent focus more on immediate rewards.

Case $gamma = 0$:

$$Reward = (V(n, n) - V(n - k, n))/V(n - k, n)$$

Case $gamma \neq 0$:

$$Reward = (V(n, n) - V(n - k, n))$$

with n the current time step, k a period and $V(n_1, n_2)$ the value of the portfolio at n_1 step with the market price at n_2 step.

It compares the current value of the portfolio with the value of the portfolio if no action was taken. Thus, this reward penalizes actions more than price dynamics. In order not to favor high portfolio values, a rate is calculated rather than a difference. In this first approach we do not consider the transaction fees.

Observations

SAC is based on Q-learning modelling, which implies that the problem must be markovian. But the market seems to have temporal dynamics in addition to noise. In order to describe the set of observations we decided to calculate RSI over 3 different periods.

Actions

There are several possibilities, the first would have been to normalize the action vector and distribute the value of the portfolio in proportion to this normalized action vector. The biggest flaw of this modelling is the inability of the model to produce a neutral action, the holding action.

We choose a specific modeling:

```
def execute_actions(self,actions):
        neg_actions = (actions < 0)*-actions
        pos_actions = (actions > 0)*actions
        if pos_actions.sum() >0:
            pos_actions = pos_actions/pos_actions.sum()
            val = np.sum(neg_actions*self.portfolio*market_price)
            new_portfolio = portfolio - neg_actions* portfolio +
                pos_actions*val/market_price
        return new_portfolio
```

This modelization keeps the possibility of a neutral action. Therefore, the general principle is to reinvest the money recovered (pos_actions) directly after the sale (neg_actions).

Chapter

3 Experiments

3.1 Generating synthetic data for financial assets

In this research project, our goal was to generate realistic synthetic financial data using data augmentation techniques. Specifically, we focused on stock market assets, with an emphasis on those related to the Tech domain. To achieve our goal, we conducted experiments using a dataset consisting of 32 assets selected from the S&P 500 stocks. These assets were chosen based on their relevance to the Tech domain and included consumer tech companies such as Apple, Google, and Microsoft, as well as telecommunications companies like ATT and Verizon, software companies such as Oracle, Salesforce, and Adobe, and hardware and semi-conductor companies like NVidia, Micron Technology, and NXP Semiconductors.

Data collection

We started off by collecting historical data on the daily adjusted closing price of these 32 assets over a 10-year period spanning from 2012 to 2022. We utilized the Yahoo Finance Python library for data collection. Once the data was collected, we performed data augmentation techniques to generate synthetic data that is realistic and plausible.

Code implementation

To generate the synthetic data, we used an open-source Python library called MZ-projection, which is an implementation of the data-driven learning technique. The MZ-projection library was developed by Shinya Maeyama from the University of Nagoya [16]. Our approach involved creating multiple versions of a given signal (stock) by leveraging the

Mori projection technique. We split the time data into training and test periods and separated the stock assets into a multivariate response variable $\mathbf{f}(\mathbf{t})$ and a multivariate explanatory variable $\mathbf{u}(\mathbf{t})$. Due to symmetry constraints with the MZ projection library, the following constraints had to be respected:

- The train period and test periods had to be of equal duration. Since we were dealing with time series data, we chose the training/fit period to be from 2012 to 2016, and the testing period to be from 2017 to 2022.
- $\mathbf{f}$ and $\mathbf{u}$ had to be comprised of the same number of variables. Since we were analyzing 32 tech assets, at each iteration we chose 16 to be part of $\mathbf{f}$ and the remaining 16 other to be part of $\mathbf{u}$.

Using the MZ-projection library, we extracted the matrix transition matrix $\mathbf{M}$ and memory kernel $\mathbf{K}$ operators from the training period data. These operators contained the complex Markovian and non-Markovian interactions between the response and explanatory variables.

Data augmentation strategy

To generate the synthetic data, we projected the extracted operators onto the space of explanatory variables for the test period. We evaluated the quality of the projection by comparing the generated signal with the actual historical signal. These steps were repeated by shuffling and permuting the assets that were chosen as explanatory variables. This effectively created n-versions of a given signal, thereby augmenting the original dataset. The main intuition behind this type of synthetic data generation is that we are creating alternative versions of a real signal by looking at its projection onto different spaces of observables. Our hypothesis was that by looking at a certain stock asset's evolution through its interaction with other similar assets, a part of the actual reality would be captured in the projection. This approach allowed us to create data that is realistic and potentially contained broader and more diverse information on the market dynamics from which the RL agent could learn.

To dive more into detail into how the implementation of the MZ library was actually done, the two main functions that were used were `split_long_time_series()` and `mzprojection_multivariate_discrete_time()`. The steps required to generate augmented data were the following:

1. Define `f` and u.
2. Split the data into training data and test data.
3. Split both the training and test time series data into smaller subsamples using the `split_long_time_series()` function.

4. Extract the Markov transition matrix **M** and memory kernel **K** from the subsampled training data using `mzprojection_multivariate_discrete_time()`.

5. Generate synthetic data for the observed signal f on the test period using the extracted projectors and data on the explanatory signal u during this period.

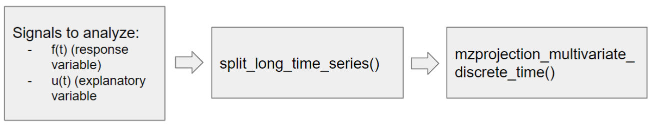

Signals to analyze:
- f(t) (response variable)
- u(t) (explanatory variable
split_long_time_series()
mzprojection_multivariate_discrete_time()

Figure 3.1: Algorithmic pipeline used for MZ projection

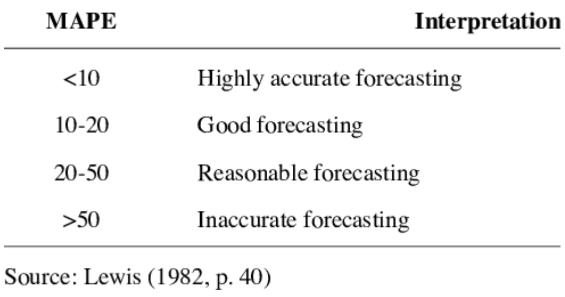
Difficulties

Some of the difficulties that were encountered during the implementation of the library was the lack of documentation that was given on its use. For instance, it took us a while to realize that the subsampling step was a necessary and inevitable step in order to the extraction of the projectors **M** and **K**. This required the data ingestion pipeline to be rigorously structured, which added overhead for the implementation. Another difficulty were the understanding of the parameters used in the function `split_long_time_series()` itself. In fact, there are 3 parameters, `ista`, `nperiod`, and `nshift`. However, since pratically no documentation was given, we had to infer their significance through empirical tests:

- `ista` corresponds to the initial time offset starting from which subsampling of the time series data is done
- `nperiod` corresponds to the time window of each subsample
- `nshift` corresponds to the time shift between subsamples

Augmentation evaluation

To evaluate the quality of the generated data, we considered two metrics. First, we looked at the mean absolute percentage error (MAPE) of the forecasting. This metric was chosen as it is intuitive to interpret. A reference point that was used to evaluate the quality of the data generation was to look at the benchmark 3.2 provided in [17]. Second, we evaluated the root mean square error (RMSE) on the derivative of the signal. Here, our objective was to see if the main dynamics were captured during the forecasting rather than the amplitude itself. Since our goal was to provide and create data that is realistic and plausible, we did not focus too much on the accurate forecasting of the signal. The goal was to provide additional data points from which the RL agent could learn.



MAPE	Interpretation
<10	Highly accurate forecasting
10-20	Good forecasting
20-50	Reasonable forecasting
>50	Inaccurate forecasting

Source: Lewis (1982, p. 40)

Figure 3.2: Interpretation of MAPE for signal forecasting

We discovered during our experimentation that the parametrization of the subsampling had an influence on the accuracy of the projection. By testing multiple values for the above mentioned parameters (`ista`, `nperiod`, `nshift`), we aimed to select those that would yield the best metric values on both MAPE and RMSE on the test data. Overall, we found that the following parameters showed projection results that were the most accurate:

- $\texttt{ista} = 0$
- $\texttt{nperiod} = 2days$
- $\texttt{nshift} = 2days$

3.2 Comparing RL agent performance on augmented dataset

For the implementation of the RL agent, we used a combination of the OpenAI `gym` API [18] and `Stable Baselines3` [19].

Since our augmented data spanned from 2017 to 2021, we decided to compare the performance of:

- A SAC RL agent trained on the baseline dataset.
- A SAC RL agent (same parameters) trained on the augmented dataset (10x).

For considerations of simplicity, we chose to test on a reduced number of tech assets, as taking the same 32 assets for RL training proved to be very time-consuming and complex. The arbitrarly chosen portfolio of assets was:

- GOOGL (Google)
- AMZN (Amazon)
- CSCO (Cisco)
- ADI (Analog Devices, Inc.)

- AAPL (Apple)

This allowed to go from an initial training dataset size of around 10,000 data points to about 100,000 data points with data augmentation.

Challenges

Training the RL model was very difficult, as the model converged very slowly and the observation did not represent the full state. In a RL model based on the Markovian approximation, this poses a problem as predictable components are limited in the world of trading.

Due to these difficulties, it was quite challenging to observe and compare the performances of a model without data augmentation and a model with data augmentation. Several training iterations showed results that were promising, with the data augmented version beating both the baseline agent and the unchanged portfolio in terms of value. However, results varied drastically from one iteration to another, making comparing general performances cumbersome.

Chapter

4

Results

4.1 Data augmentation

Here are some examples of MZ projection results that were obtained through the experiments.

For certain major stock assets such as Google, the projection yielded very positive results. This can be seen just by comparing the projection with the actual signal on a time window of 80 days (test set):

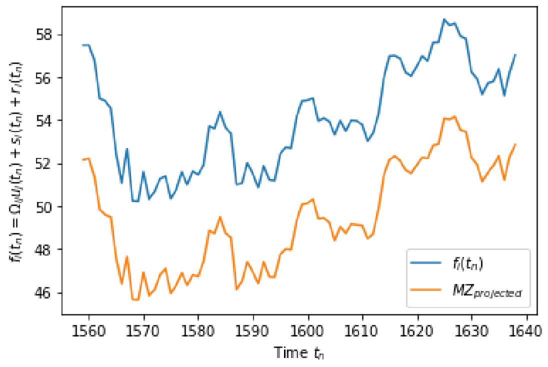


Figure 4.1: MZ projection for Alphabet class C GOOG (time scale is in days). In blue is the original signal, and orange the signal obtained through the projection.

Looking at the derivative, we also see that projection manages to capture the dynamics very well.

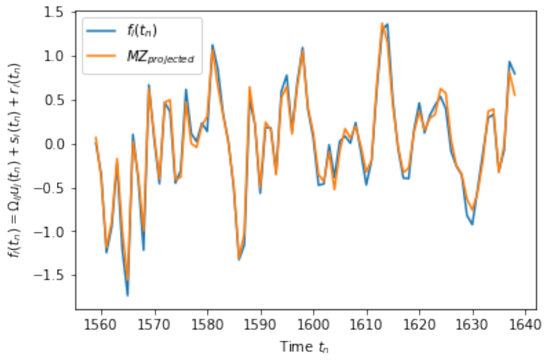


Figure 4.2: MZ projection for Alphabet class C GOOG (time scale is in days) - derivative

For others, the projection is less accurate, both for the signal and its derivative.

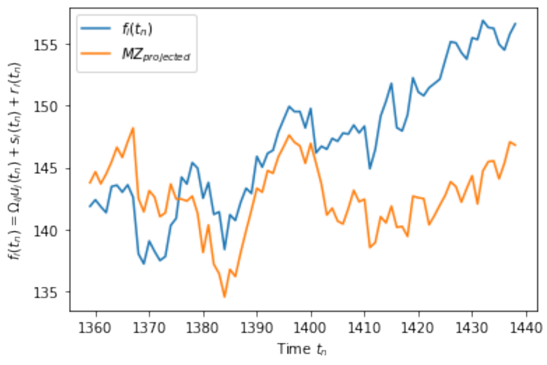


Figure 4.3: MZ projection for Adobe

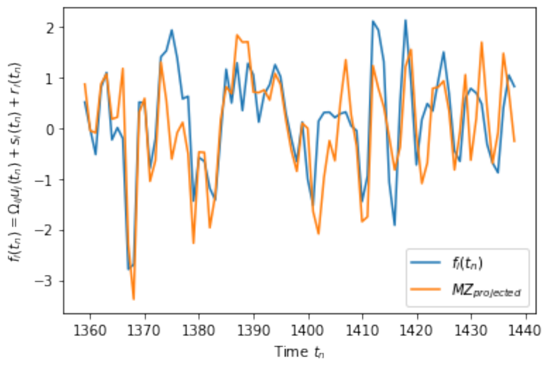


Figure 4.4: MZ projection for Adobe - derivative

The assets chosen as explanatory variables and response variables were permuted multiple times, and this allowed to get a total of 10 alternate versions of every asset, meaning that we were able to augment the original data by a factor of 10 for a time period that spanned from 2017 to 2021. Here are some examples of the total projection for a given permutation of assets:

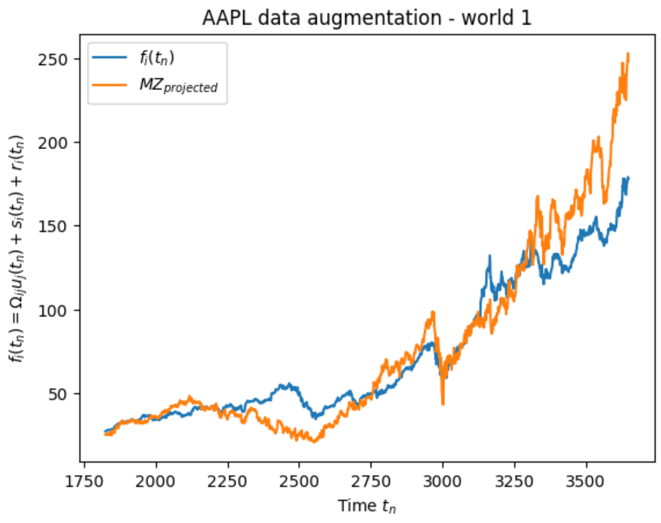


Figure 4.5: Data augmentation for AAPL (Apple)

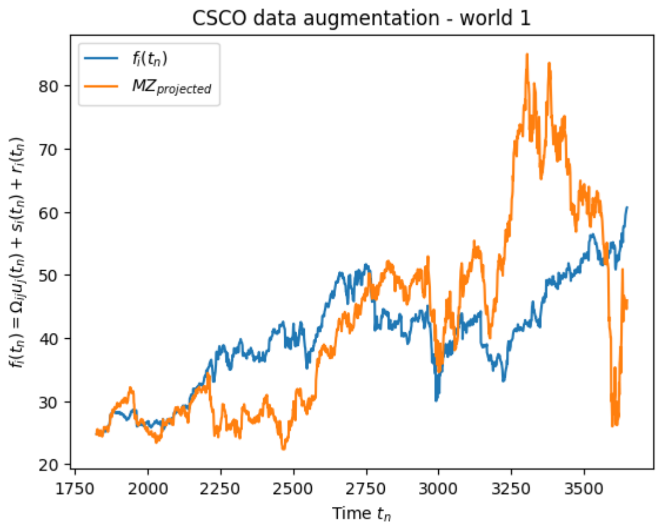


Figure 4.6: Data augmentation for CSCO (Cisco)

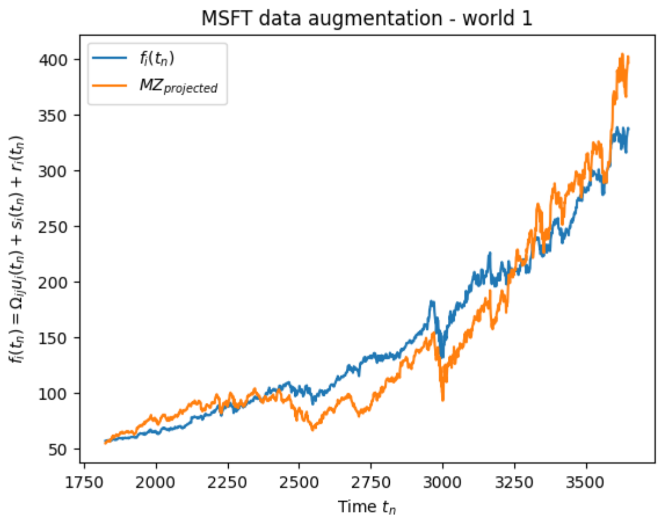


Figure 4.7: Data augmentation for MSFT (Microsoft)

4.2 Asset portfolio allocation

As mentioned previously, certain iterations of training the RL agent led to positive results. For instance, on one iteration we observed the following during the training and test phases:

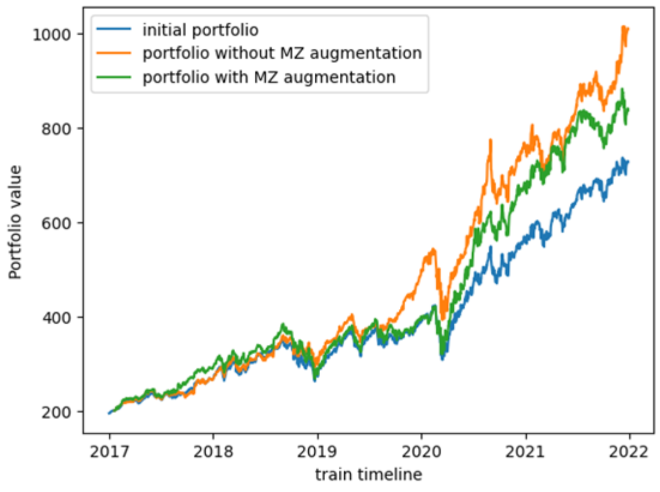


Figure 4.8: Portfolio allocation performance (Training phase)

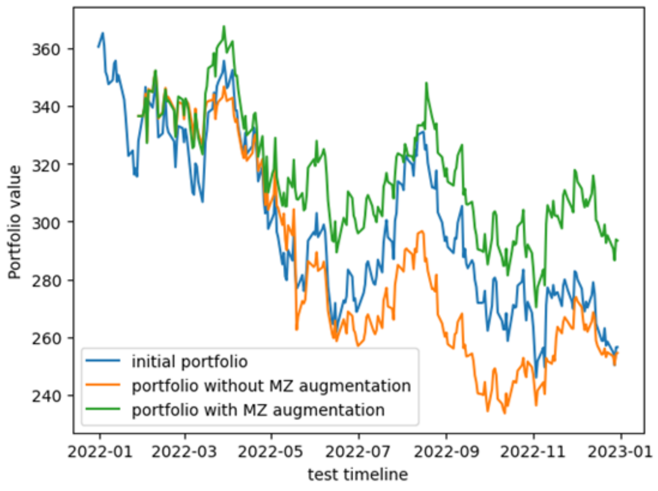


Figure 4.9: Portfolio allocation performance (Test phase)

In 4.8, we see that the agent without the data augmentation (orange) outperforms the agent with data augmentation (green) and both increase its initial value (blue). However, in 4.9, it is actually the agent with the data augmentation that manages to make higher gains than the one without. In fact, the agent without data augmentation actually performs worse than the portfolio with no agent intervention. We can infer from these observations that the agent without augmentation over fitted on the training data, and its generalization capabilities were poor (the year 2022 was a particularly complicated year in terms of stock asset valuation). On the other hand, it would seem that the MZ data augmentation provided more data points from which the agent learned more general realities, more diverse and broader interactions that ultimately allowed it to outperform both the baseline agent and the initial portfolio.

Unfortunately, due to the stochastic nature of the RL agent learning phase itself, it was hard to observe these types of results in a recurring way, and statistically the performance of both type of agents seemed to converge towards a same distribution:

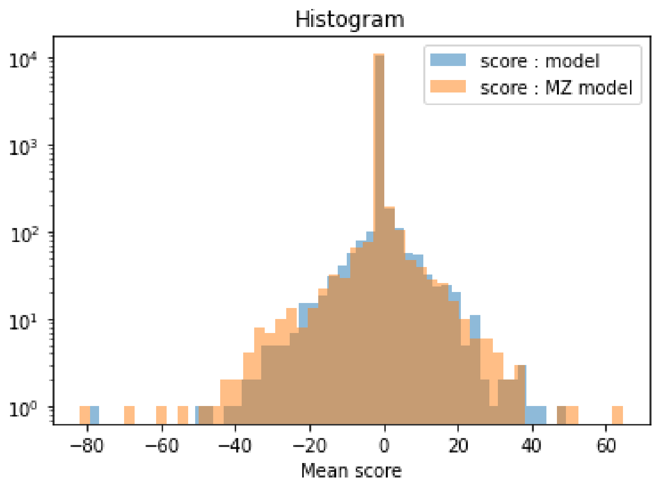


Figure 4.10: Distribution of agent performance (reward value) on 100 iterations

As it can be seen on 4.10 which plots the reward values for each time step in the test data over 100 training iterations, both agent variants (baseline in blue, augmented in orange) show similar performance.

Chapter

5 Discussion

5.1 Data augmentation

An important point to take note of is that the Mori-Zwanzig projection method that allowed us to get the augmented signals cannot be used for actual forecasting and predicting future behavior. In fact, as it was shown in the Experiments section, the projection that was obtained for the response signal **f** during the test period was only possible because we had the information on the explanatory signal **u** during this same period. In other words, if one wanted to forecast a certain signal's future behavior using the MZ projection technique that we showed, one would already need to have information on the future behavior of some other explanatory signal.

As such, the sole purpose for the MZ projection in our case was to augment the data, and create realistic alternate versions of a same signal. The data-driven learning algorithms that were used to extract the different Mori-Zwanzig projectors differ from traditional Machine Learning or Deep Learning methods. As such, even if certain projections seem inaccurate (for example for 4.6), they are representations that capture the relationships and dynamics between the selected explanatory and response variables. The difference with the actual signal could come from an unobserved part of reality (other financial assets, historical events...) that wasn't taken into account for the projection. Yet the projection itself stays loyal to the inductive bias that we introduced which is that its behavior is captured and transcribed in the interactions with other similar class assets.

Our method differs from the data augmentation method presented in [5]. In this work, the authors simply augment data by taking additional data points in minute-candle data, that is instead of only considering the closing price, additional price points (open, high, low) are added to increase the number of data points. Aside from the fact that in this framework, one would need to provide additional effort in the data collection phase, some components of minute-candle data aren't always openly accessible. Our approach on the other hand actually enriches the information by incorporating statistical correlations and dynamics between the observed assets into the training data. Also, it is more generalizable and can be integrated within an automatized framework.

5.2 Asset portfolio allocation

As seen in the results, the distribution of model performance shows that on average, the gains tend to converge towards 0 (the model isn't improving on portfolio allocation). This could stem from a number of potential factors:

- The market data itself is too chaotic and contains too much noise, preventing the agent from learning any statistically significant feature.
- The RL agent parametrization was faulty, a more careful implementation could have led to having a significant difference between unaugmented and augmented data.
- The choice of assets to augment data and the choice of assets for our portfolio allocations were poorly chosen. Other financial assets would have shown more clearly the validity of our approach.

The plurality of the nature of potential factors is a clear indicator that much more additional time is required to carefully investigate the root cause of the problem. However, what is clear is that certain iterations provide promising results, and consequently identifying what allowed for those higher gains could lead to constructing an efficient automated trading agent. Also, changing the strategy of the allocation itself is a core topic that deserves to be studied: for instance, rather than having the agent constantly try to allocate its resources, guardrails that only allow action above a safety threshold could prove to be beneficial.

Chapter

6 Conclusion

Overall, our experiments demonstrated that the Mori projection technique is a promising approach for generating realistic synthetic financial data. Although, no statistically significant occurrences of an improvement on agent performance when using this augmentation was detected in our experiments, several iteration results do show quite promising results. With more time and resources, modifications to the implementation could be brought to show the validity of the approach and confirm or revoke the initial hypothesis, that data generated by this technique can potentially provide valuable information for reinforcement learning agents, allowing them to learn from a broader and more diverse set of data points.

In addition, what was clearly identified during the execution of the different experiments is the great value that resides in the Mori-Zwanzig projection technique. In fact, rather than augmenting data through the creation of alternative signals as we have done in our project, the MZ projection technique and library could be implemented as a missing value imputation method. Considering time series data where certain features would be missing points, the MZ technique could be used to fill in these points by relying on data points of other features with great accuracy. Comparing this imputation technique with already known imputation techniques could be a research topic in itself.

Finally, here we've integrated the Mori-Zwanzig formalism in the context of data augmentation. An alternative method for increasing performance of RL agents could be to integrate certain MZ formalism components in the RL parametrization, in the definition of actions or rewards for example, and therefore serve as powerful neuro-symbolic approach.

Bibliography

[1] Ash Booth, Enrico Gerding, and Frank McGroarty. "Automated trading with performance weighted random forests and seasonality." In: *Expert Systems with Applications* 41.8 (2014), pp. 3651–3661.

[2] Richard Haynes and John S Roberts. "Automated trading in futures markets." In: *CFTC White Paper* (2015).

[3] Boming Huang, Yuxiang Huan, Li Da Xu, Lirong Zheng, and Zhuo Zou. "Automated trading systems statistical and machine learning methods and hardware implementation: a survey." In: *Enterprise Information Systems* 13.1 (2019), pp. 132–144.

[4] Yifan Zhang, Peilin Zhao, Qingyao Wu, Bin Li, Junzhou Huang, and Mingkui Tan. "Cost-sensitive portfolio selection via deep reinforcement learning." In: *IEEE Transactions on Knowledge and Data Engineering* 34.1 (2020), pp. 236–248.

[5] Yuyu Yuan, Wen Wen, and Jincui Yang. "Using data augmentation based reinforcement learning for daily stock trading." In: *Electronics* 9.9 (2020), p. 1384.

[6] Weiqi Chu and Xiantao Li. "The Mori-Zwanzig formalism for the derivation of a fluctuating heat conduction model from molecular dynamics." In: *arXiv preprint arXiv:1709.05928* (2017).

[7] Zhen Li, Hee Sun Lee, Eric Darve, and George Em Karniadakis. "Computing the non-Markovian coarse-grained interactions derived from the Mori–Zwanzig formalism in molecular systems: application to polymer melts." In: *The Journal of chemical physics* 146.1 (2017), p. 014104.

[8] Ralph Abraham and Jerrold E Marsden. *Foundations of mechanics*. 364. American Mathematical Soc., 2008.

[9] Steven H Strogatz. *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[10] Bernard O Koopman. "Hamiltonian systems and transformation in Hilbert space." In: *Proceedings of the National Academy of Sciences* 17.5 (1931), pp. 315–318.

[11] Bernard O Koopman and J v Neumann. "Dynamical systems of continuous spectra." In: *Proceedings of the National Academy of Sciences* 18.3 (1932), pp. 255–263.

[12] Yen Ting Lin, Yifeng Tian, Daniel Livescu, and Marian Anghel. "Data-Driven Learning for the Mori–Zwanzig Formalism: A Generalization of the Koopman Learning Framework." In: *SIAM Journal on Applied Dynamical Systems* 20.4 (2021), pp. 2558–2601.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[14] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning." In: *arXiv preprint arXiv:1509.02971* (2015).

[15] Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods." In: *International conference on machine learning*. PMLR. 2018, pp. 1587–1596.

[16] Shinya Maeyama and Tomo-Hiko Watanabe. "Extracting and modeling the effects of small-scale fluctuations on large-scale fluctuations by Mori–Zwanzig projection operator method." In: *Journal of the Physical Society of Japan* 89.2 (2020), p. 024401.

[17] C Lewis. "International and Business Forecasting Methods Butterworths: London." In: (1982).

[18] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "Openai gym." In: *arXiv preprint arXiv:1606.01540* (2016).

[19] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. *Stable baselines3*. 2019.

[20] Chunli Liu, Carmine Ventre, and Maria Polukarov. "Synthetic Data Augmentation for Deep Reinforcement Learning in Financial Trading." In: *Proceedings of the Third ACM International Conference on AI in Finance*. 2022, pp. 343–351.

[21] Daniel T Schmitt. "Time series analysis of real-world complex systems-climate, finance, proteins, and physiology." PhD thesis. Universität Ulm, 2007.