Exploring Performance of Reinforcement Learning compared to other Heuristics in Low Predictability Environments

Applications in Digital Finance

METTE WEISFELT, University of Twente, The Netherlands

Reinforcement Learning has become increasingly popular in addressing issues in financial environments. A key strength of Reinforcement Learning lies in its sequential decision-making capability, where each choice influences future outcomes. This mirrors real-world financial markets, such as long-term investments. This research paper explores the performance of Reinforcement Learning by playing two games with different complexities. The games provide multiple controlled environments where agents must make decisions that are difficult to predict, simulating the challenges faced in real-world financial contexts. Results show that the complexity and predictability of generators significantly influence model performance. Simpler heuristic algorithms outperform Inferring Models in simple, predictable environments. The ability to ignore noise and focus on fundamental patterns often outweighs attempts to learn nonexistent patterns. Reward signal adjustments and hyperparameter tuning can increase performance of the Reinforcement Learning models.

Additional Key Words and Phrases: Digital Finance, Machine Learning, Reinforcement Learning, Low Predictability, Simple Game Environments

1 INTRODUCTION

Over the last few decades the field of digital finance has evolved, transitioning from basic tools like spreadsheets to complex machine learning-driven investment strategies, taking over decision-making from humans. More specifically, Reinforcement Learning has gained interest as a supporting tool[5][12]. A key strength of Reinforcement Learning lies in its sequential decision-making capability, where each choice influences future outcomes. This mirrors real-world financial challenges, such as long-term investments, where uncertainty plays a big role[4]. To better analyze the practical potential of Reinforcement Learning in these financial contexts, this research will explore its performance in controlled game environments, specifically designed to be abstract models of different financial scenarios. These scenarios simulate market-like dynamics, where multiple agents must make decisions that are difficult to predict, simulating the challenges faced in real-world financial contexts.

2 THEORETICAL BACKGROUND

Reinforcement Learning is a subfield of machine learning that focuses on learning behavior through repeated interaction with an environment[15]. The purpose of Reinforcement Learning is to learn some strategy that maximizes the reward that accumulates. It differs from other kinds of supervised learning in that it does not need any labeled input output pairs[11]. It also does not need suboptimal actions to be corrected immediately. Instead, its 'success' is only defined as the accumulated reward at the end. Therefore, Reinforcement Learning is well-suited to problems that include a long-term versus short-term reward trade-off, such as investing, or problems that do not have a predefined step-by-step solution but where the optimal strategy depends on a wide range of factors. A Reinforcement Learning agent can be modeled as a Markov Decision Process[10]. Formally, a Reinforcement Learning Model can be described as:

- A set of actions (action space), A.
- A set of environment and agent states (state space), S.
- The transition probability,

$$P_a(s, s') = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a),$$

  which describes the probability of transitioning from state s to state s' under action a at time t.
- $R_a(s, s')$, the reward received after transitioning from state s to state s' under action a.

The Reinforcement Learning agent learns by interacting with its environment. It sees how the environment or the final score changes after the interaction, and then decides whether or not that action had any benefit to the goal. More formally, at each interaction the agent looks at its current state S_t (from the state space S), chooses an action A_t (from the action space A), and moves to a new state S_{t+1}. How the environment changes depends on the transition probability, which tells us the chance of moving from one state to another based on the chosen action. After each move, the agent receives a reward, $R_a(s, s')$, which tells it how good or bad that action was. Over time, the agent learns to choose actions that maximize the total reward.

This research will focus on a model-free approach, where the agent learns directly from the consequences of its actions by interacting with the environment and changing its behavior based on the rewards it receives[15]. In finance, there is no complete or precise model of the market. Developing such a model is close to impossible[3]. Instead of trying to model every detail, the Reinforcement Learning heuristic used will learn from the rewards provided by correct or incorrect movements and past outcomes. It will aim to learn both short and long term patterns and develop strategies that will maximize the final score.

There are different types of Reinforcement Learning algorithms[9][2]. For this research, a Double Deep-Q Network will be used over a traditional Q-Learning Table approach. The primary reason for this choice is its ability to generalize well, as not all possible states are seen during training[18]. A Deep-Q Network generalizes well by studying the relationships between the states instead of trying to explore all possible states[7]. It supports the incorporation of Temporal Patterns and other stabilization techniques[10]. It also offers the flexibility to play with reward structures in our environment[6].

In this research, two games will be played. In each of these games, two actors can be defined. The first actor is a mathematical *generator*. A generator will be used to generate a sequence of real discrete numbers, simulating the natural fluctuations of a stock over time. The other actor in the game is the *observer*. The observer is the Reinforcement Learning model or other heuristic that will predict the next number or movement in the sequence of numbers generated by the generator. This setup is an abstraction of one of the most common challenges in finance: forecasting market trends and price movements. The ability to more accurately predict what the next price a stock or index will be, could turn a real profit.

3 RESEARCH QUESTION

In order to examine how effectively Reinforcement Learning can be applied in this sequential decision-making context, the following research question arises.

How does Reinforcement Learning perform when interacting with generators of diverse complexity compared to other heuristic-based decision-making algorithms in simple game environments characterized by low predictability?

The following three sub questions have been identified to assist in addressing the main research question:

1. *How does the complexity or predictability of different generators impact the performance of Reinforcement Learning agents compared to other heuristic-based decision-making algorithms?*
2. *Does the rank ordering of generators by predictability correlate across different observers?*
3. *Does the rank-order of the generator correlate with the outperformance of Reinforcement Learning approaches?*

4 METHOD OF RESEARCH

In order to address the goal of exploring the performance of Reinforcement Learning agents compared to other heuristics, simulations will be performed with different observers and generators in two simple game environments. The first environment will be a simple betting game. The second game environment will be a payoff game. The generators and observers are categorized into different classes, each representing a distinct level of increasing complexity.

4.1 Simple Betting Game

The Simple Betting Game serves as the foundation for this research. In this game environment, a generator produces a time series, and observers place bets predicting whether the next value in the series will be higher or lower compared to the previous value. The choice of higher or lower avoids the challenge of quantifying bad guesses. In a financial context, if a stock has a value of 40 and the observer bets 60 but the price rises to 80, the direction is correct yet the magnitude of the error makes it a poor bet. In applications of digital finance, such a bet could still be profitable. One could buy an option based on the "higher" prediction and still make a profit, even if the exact value differs significantly from the bet. This shows the flexibility in certain financial contexts, where the direction of movement matters more than the magnitude of the bet. This simplicity of only using "higher" or "lower" focuses purely on directional accuracy, without accounting for how far off the prediction might be. The Simple Betting Game can be mathematically formalized as follows. Let

- X_t represent a time series value in time t, where X is generated by a generator G.
- O represent the observer, who makes predictions $P_t \in \{\text{Higher}, \text{Lower}\}$.

At each time step t:

1. **Generator:** X_t is updated by generator G, so that:
$$X_{t+1} = G(X_t).$$
2. **Observer:** The observer makes a prediction P_t on the direction of X_{t+1} relative to X_t:
   - P_t = Higher if the observer predicts $X_{t+1} > X_t$,
   - P_t = Lower if the observer predicts $X_{t+1} < X_t$.
3. **Reward:**
   - If P_t matches the actual direction of X_{t+1}, the observer receives a reward $R_t = +1$.
   - If P_t does not match the actual direction, the observer receives nothing.

The total reward R over T time steps is therefore:
$$R = \sum_{t=1}^{T} r_t.$$

4.2 Payoff Game

The Payoff Game extends the Simple Betting Game by allowing the observer to place bets among different payoff functions, enabling it to manage the exposure to the magnitude of movement of the next number in the time series. This mirrors real-world investing, where the use of derivatives allows for complex pay-off structures that manage exposure to movements. This offers more options to balance risk and reward. To be able to introduce this new dimension of complexity, the observer now has to predict the next number in the sequence correctly, instead of only predicting the direction of movement. After making a prediction, the observer can choose how to allocate its payoff, the possible reward, across different functions. The choice of allocation between the payoff functions by the observer implicitly reflects their view and confidence on the direction, magnitude, or volatility of the change.

4.2.1 Payoff Functions. In these simulations, the observer can choose between three types of payoff functions.

- Linear Payoff: $h(\Delta x) = k\Delta x$, where k is some constant multiplier.
- Non-linear Payoff: $h(\Delta x) = \max(\Delta x - K, p)$, modeling an option-like payoff where the reward is activated only if the change of x exceeds a threshold K
- Custom Structures: $h(\Delta x) = x^2 + K$ or $h(\Delta x) = x^2 - K$, modeling convex and concave payoff functions. This gives the ability to represent high-risk and high-reward scenarios, and diminishing returns where additional risk yields smaller incremental rewards, respectively.

The Payoff Game can be mathematically formalized as follows. Let:

- X_t represents a time series value at time t, where X is generated by a generator G.
- $h_i(\Delta x)$ represents one of the n payoff functions available to the observer, where $i \in \{1, \ldots, n\}$.
- $w_{t,i}$ represents the allocation weight on payoff function $h_i(\Delta x)$ at time t, with $\sum_{i=1}^{n} w_{t,i} = 1$.

At each time step t:

(1) **Generator:** X_t is updated by generator G, so that:

$$X_{t+1} = G(X_t).$$

(2) **Observer:** The observer allocates weights $w_{t,i}$ to each payoff function $h_i(\Delta x)$

(3) **Reward:** The reward r_t at each time step is determined by the sum of weighted payoffs across all payoff functions:

$$r_t = \sum_{i=1}^{n} w_{t,i} \cdot h_i(\Delta x).$$

4.2.2 Extension - Cash Budget Reserve and Dynamic Bet Sizing. To increase the complexity and adaptability of the observer in the simulations and make the game more representative of a stock market, an extension is introduced by incorporating a cash budget reserve. The observer is assigned a reserve which cannot fall below zero, effectively modeling the concept of bankruptcy or debt. This also enables the implementation of dynamic bet sizing, allowing the observer to adjust the size of its bets, influencing its potential reward. This gives an implicit view on the confidence of the bet.

4.3 Observers

The observers are grouped into different classes of complexity. In earlier research, different frameworks for categorizing the complexity of algorithms have been proposed[19]. However, these frameworks often revolve around the efficiency of the algorithm or the polynomial-time approximation[13], instead of the 'learning' involved in trying to accurately predict some next movement. In order to properly address all specific requirements of this research, such as the ability to differentiate between time-windowed heuristics and heuristics that incorporate all historical data, a new framework to categorize has been developed. See Table 1 for an overview of these classes.

There will be 7 different observers in 5 classes of complexity playing these games. The choice of seven observers was to cover and explore as many different types of complexity within the limitations of the study. Only the Reinforcement Learning Observer plays the Payoff Game, as the other observers do not support any type of different reward structure.

4.3.1 Zero Class Observer. A Zero Class Observer makes decisions based solely on the currently available information, without any context or knowledge of past movements. The specific implementation used in the Simple Betting Game always bets "higher".

4.3.2 Fixed Rule Class Observer. A Fixed Rule Class Observer knows about the previous movement. In the simulations for the Simple Betting Game, its strategy will be to repeat the previous correct movement.

Class Name	**Complexity**	**Observer Name**
Zero Class	Does not need any previous movements	Zero Observer
Fixed Rule Model Class	Needs the previous movement	Fixed Rule Observer
Stationary Stochastic Model Class	Access to all historical movements	Bernoulli Observer, Mean Reversion Observer
Dynamic Stochastic Model Class	Access to all historical movements but uses some general time-windowed heuristic	Dynamic Mean Reversion Observer
Inferring Model Class	Infer patterns and rules of the game without explicit definitions	Reinforcement Learning Observer, Classifier Observer

Table 1. Classes of complexity for observers with associated observer names

4.3.3 Stationary Stochastic Model Class Observers. In this context, stationary refers to the observer maintaining consistent rules and using the same data set as the series progresses, only adding to the set of movements as the time series continues. This class consists of two observers with different strategies. The first observer will attempt a 'Bernoulli' approach. The observer estimates the probability of each outcome based on historical movements. For instance, if "higher" has occurred more frequently than "lower" in past bets, the probability of "higher" is considered greater. This observer will analyze the ratio of "higher" to "lower" outcomes and bet the option that has occurred most frequently so far. The second observer employs a mean reversion strategy using the full history of movements. Rather than relying solely on "higher" or "lower" as decision variables, this observer derives its own variable. Specifically, it calculates the mean of the set of data and bases its next bet on a movement toward that mean.

4.3.4 Dynamic Stochastic Model Class Observer. Stationary models, which rely on the entire history, are particularly sensitive to outliers. This makes it interesting to explore how a time-windowed approach performs, as it has the potential to mitigate the impact of outliers. In the simulations, the observer will continue to use the mean reversion strategy but will now base decisions only on a fixed number of recent movements, instead of the entire set of previous movements. For simplicity, this observer will use a fixed window length of the last 50 movements, instead of using a percentage of the total history. This shift in the input data for each bet is considered dynamic.

4.3.5 Inferring Model Class Observers. This class consists of two observers. The first observer is a generic classifier. The classifier used is an SGD Classifier, as this type of classifier is very effective in classifying binary decisions and can support real-time training [16]. The second observer will be a Double Deep-Q Network Reinforcement Learning agent.

4.4 Generators

There will be 5 different generating functions used in the game environments. All but the Stock Data Generator have a starting value of 0. This initial state is the first number that observers get to observe.

4.4.1 Linear Generator. A linear generator will generate a linear sequence of numbers. Formally, the generator can be described as:

$$S_{n+1} = S_n + 1$$

4.4.2 Normal Distribution Generator. A normal distribution generator will generate a time series with values following a normal distribution. In such a distribution, most of the generated values appear around a central mean, with fewer values appearing as they deviate further from the mean. Formally, the generator can be described as:

$$x_n = \mathcal{N}(\mu = 40, \sigma = 15)$$

In the simulations, a mean of 40 and a standard deviation of 15 have been randomly chosen. It does not matter which numbers are chosen as the Simple Betting Game only focuses on the direction of movement rather than the exact value of the next number to predict.

4.4.3 Uniform Distribution Generator. A uniform distribution generator produces a time series in which the values follow a uniform distribution. Here, all values within a specified range have an equal probability of occurring and are not concentrated around a central value like the mean. Formally, the generator can be described as:

$$x_n = \mathcal{U}(0, 40)$$

This uniform distribution uses an interval from 0 to 40. Again, this does not matter as the observer only looks at direction of movement in the Simple Betting Game.

4.4.4 Cellular Automaton Generator. A cellular automaton generator produces a time series based on a cellular automaton model. A cellular automaton consists of a grid of cells, each of which can be in one of a finite number of states. The state of each cell is determined by a set of specific rules that depend on the states of adjacent cells. Over time, the states of the cells change according to these rules, creating complex patterns from simple initial conditions [14]. In these simulations, a two-dimensional grid will be used. Formally, the generator can be described as an initial array of cells:

$$\mathrm{S}_0 = [0, 0, 0, 1, 1, 1, 1, 0]$$

With an update function, which is applied based on the current value of the cell and its neighbours where the indices of the array are cyclic:

$$\text{new\_state}(i) = \text{rule}(s_{i-1} \cdot 4 + s_i \cdot 2 + s_{i+1})$$

Here,

- s_{i-1} represents the state of the cell to the left of cell s_i,
- s_i represents the state of the current cell,
- s_{i+1} represents the state of the cell to the right of cell s_i.

After the update function has been applied, the state array has 'evolved'. After each evolution, a new number is generated from the new state array. Formally, this calculation rule is as follows:

$$\text{Value} = \sum_{i=1}^{n} (-1)^i \cdot s_i$$

where s_i is the latest state array

4.4.5 Real-world Stock Data Generator. The final generator will sample from real-world stock data, focusing on a single stock rather than an index. This is because the composition of stocks within an index changes over time. In the simulations, the opening and closing values of Apple stock have been chosen as the numbers the generator will generate. The first number will be the value of the stock as on the first of January 2000.

4.5 Performance Metrics

The outcomes of the games will be evaluated on four metrics. These metrics will give insights into the final score, but also on how the scores and performance evolve over time.

4.5.1 Accuracy over time. All observers will be judged on average accuracy over time. This will give insights on how the observer adjusts itself and possibly learns over time. The average accuracy over time is calculated by taking the average accuracy for each turn in all simulations combined.

4.5.2 Distribution of final scores across simulations. The distribution of scores across simulations is a common analysis metric in Reinforcement Learning and other simulation-based experiments[17]. It provides insights into the consistency of the performance. This will also detect outlier simulations or other unusual scenarios in which the observer performed significantly better or worse compared to other simulations.

4.5.3 Training and inference time. In the first four classes of complexity, the training time is zero. For the inference models class, both the classifier and the Reinforcement Learning require training time. Inference time is the time it takes the observer to place their bet. In certain applications, such as digital finance, the inference time is of relevance. If it takes too long to place a bet, the optimal bet may no longer be relevant. For instance, in the context of high-frequency trading, where trades are executed in milliseconds, any delay in decision-making can significantly impact profits.

4.5.4 Prediction error over time. Prediction error over time is a metric that will only be used for the Payoff Game. This will track whether the agent is adapting correctly to the environment and thus if it makes more or less mistakes as the sequence progresses. The prediction error is defined as the absolute difference between the correct value and what the observer predicted it would be.

5 SIMPLE BETTING GAME RESULTS

In this section, the results for the Simple Betting Game will be discussed. The results are grouped by generator. Subplots have been generated for better readability if the differences in range of the final scores were too large.

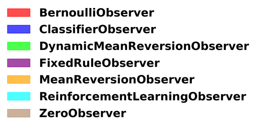

Fig. 1. Global Legend Observers and associated colors

5.1 Linear Generator

The Linear Generator shows an interesting distinction between the different complexities of observers. Some observers perform exceptionally well, such as the Zero Observer, the Bernoulli Observer, the Classifier Observer, and the Fixed Rule Observer, all of which achieve the highest possible score. In contrast, observers in the Dynamic and Stationary Stochastic Model Classes, specifically the Mean and Dynamic Mean Reversion Observer, perform the worst, consistently achieving an accuracy of 0% that does not improve over time. The observers in the Inferring Model Class show a different pattern: they start with a low accuracy in making predictions, but improve as rounds progress, as can be seen in Figure 3, demonstrating their learning over time. The Classifier Observer seems to be learning the best, quickly getting to a 100% accuracy. The more complex Reinforcement Learning Observer learns but seems to be over engineered for this simple use case, as for instance the warm-up phase and any further exploration is unnecessary. Mean reversion strategies fail completely due to the incorrect assumption about the nature of the sequence of numbers generated. This demonstrates how simpler heuristics can outperform more complex models when the underlying pattern matches their assumptions.

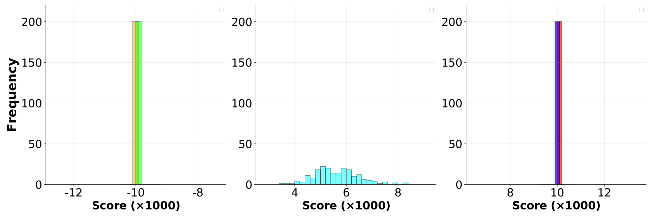

Fig. 2. Histogram of Final Score Distributions for Linear Generator

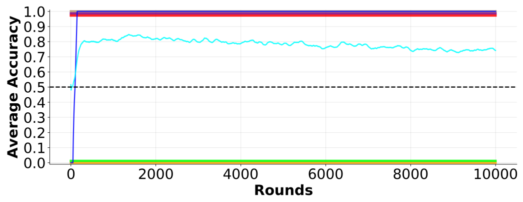

Fig. 3. Accuracy over Time per Observer for Linear Generator

5.2 Uniform Distribution Generator

The Uniform Distribution Generator is a fundamentally different challenge for the heuristics, as the generator now produces 'random' movements rather than deterministic ones. The Fixed Rule

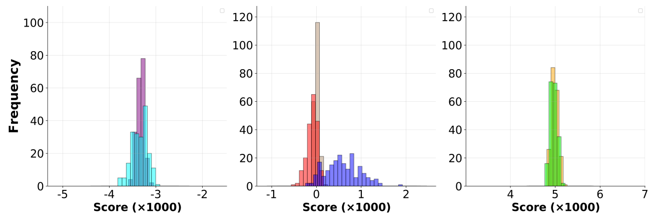

Fig. 4. Histogram of Final Score Distributions per Observer for Uniform Distribution Generator

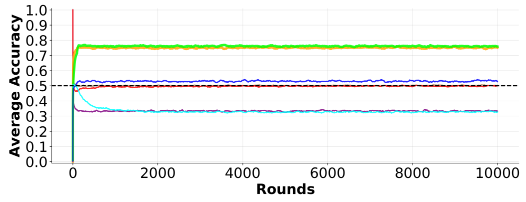

Fig. 5. Accuracy over Time per Observer for Uniform Distribution Generator

Observer and the Reinforcement Learning Observer both performed significantly worse than the others. The Fixed Rule Observer, the Bernoulli Observer, and the Classifier Observer performed slightly better with an average accuracy of about 50%. The Stationary and Dynamic Stochastic Models performed the best. Analysis of the accuracy over time reveals that none of the algorithms showed any sign of learning or improved performance as the rounds progressed. This logically follows from the type of mathematical generator that the Uniform Distribution Generator is. Because the previous movements are not predictive indicators for the next movement, the Inferring Model Class observers are not able to learn patterns. This shows that increased model complexity does not always lead to better performance, especially when the generated data is random.

5.3 Normal Distribution Generator

The Normal Distribution Generator and Uniform Distribution Generator results are very similar. There are no significant differences between the two generators with regards to the grouping of observers. The accuracy over time graph shows roughly the same results as well.

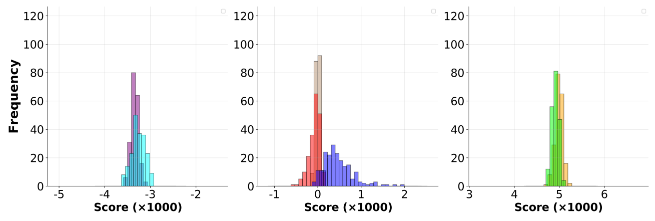

Fig. 6. Histogram of Final Score Distributions for Normal Distribution Generator

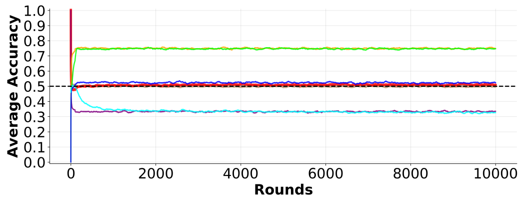

Fig. 7. Accuracy over Time per Observer for Normal Distribution Generator

Again, any previous movements are not predictive indicators for the next number in the generated sequence, which prevents the Inferring Model Class observers from discovering patterns. The difference between the Dynamic Mean Reversion Observer and the Mean Reversion Observer, the window-based approach, does not significantly influence the final score either, as future movements are independent of past movements.

5.4 Cellular Automaton Generator

The Cellular Automaton Generator histogram shows a similar grouping of observers as the other generators discussed so far. The Fixed Rule Observer, the Reinforcement Learning Observer, the Zero Observer, and the Bernoulli Observer performed significantly worse than the Classifier Observer and both Mean Reversion observers. Their better performance suggests the patterns of this specific Cellular Automaton evolution rule oscillate around a central mean. The time-windowed approach compared to the full history was equally effective, with both their final scores averaging 6000 points. The Classifier performed quite well compared to the Reinforcement Learning Observer, suggesting that the more complex Deep-Q Network may have overfit or that the exploration-exploitation balance was not correct for this use-case. Again, simpler statistical approaches outperformed more complex heuristics.

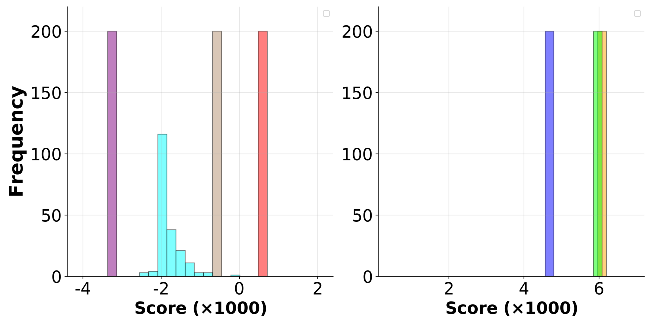

Fig. 8. Histogram of Final Score Distributions for Cellular Automaton Generator

5.5 Stock Data Generator

The Stock Data Generator produces results that differ significantly from the other generators. In the simulations conducted, the Classifier Observer performs the worst, alongside the Mean Reversion Observer from the Stationary Stochastic Model Class. Interestingly, the Dynamic Mean Reversion Observer shows an improvement

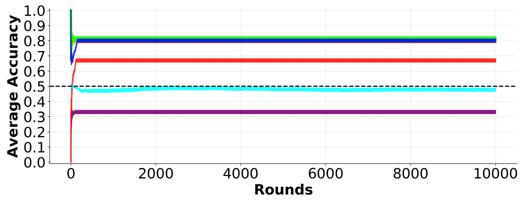

Fig. 9. Accuracy over Time per Observer for Cellular Automaton Generator

in its final score compared to the Mean Reversion Observer. The Simple Reinforcement Learning Observer, the Fixed Rule Observer, and the Zero Observer outperform the others, achieving an average score of around 600. There are no real trends in the accuracy over time, showing that none of the heuristics learn over time. Statistical heuristics that use simple mean values capture market behavior more effectively than complex models. This further confirms that stock prices tend to move around moving averages, which research has shown to roughly be the case[8][1].

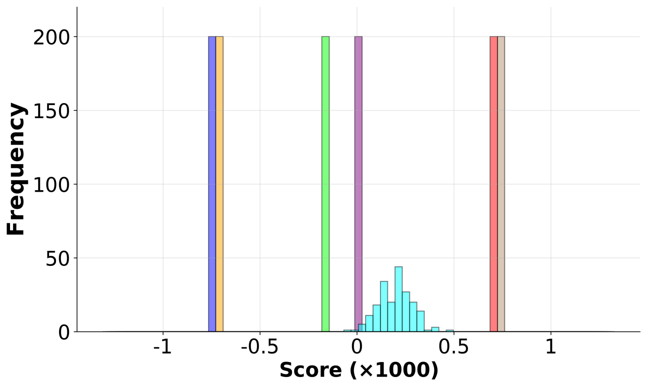

Fig. 10. Histogram of Final Score Distributions for Stock Data Generator

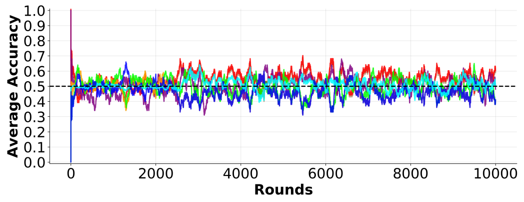

Fig. 11. Accuracy over Time per Observer for Stock Data Generator

6 PAYOFF GAME RESULTS

This section contains the results for the Payoff Game. The simulations below are only run with adjusted Reinforcement Learning observers, as they are the only observers in the classification that can incorporate both the payoff structure and extension in the game. Three extra simulations have taken place. In the first simulation, the only adjustment that was made was the ability to predict real discrete numbers instead of only the direction of movement.

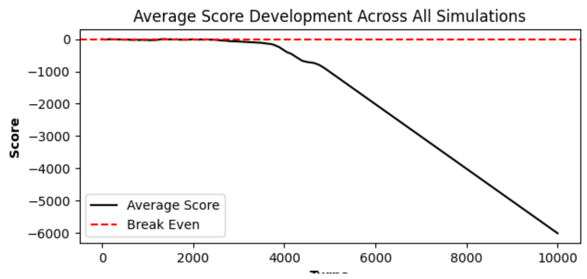

Fig. 12. Average Score over Time

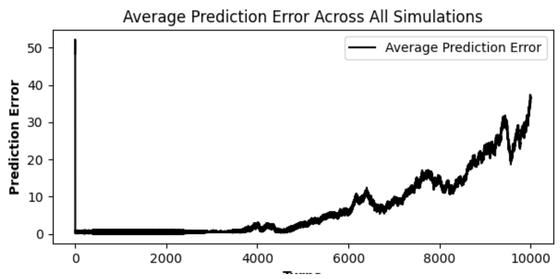

Fig. 13. Average Prediction Error over Time

As can be seen in Figure 12 and 13, the Reinforcement Learning Observer does not perform well. Average score drops down drastically, and the average prediction error increases as the game continues. The observer might be overfitting to historical data patterns. There could also be an accumulation of errors in the learning process. The observer might be trapped in small local optimal strategies, losing sight of the big picture and long-term strategies. This simulation suggests that without any form of external reward signal, the Reinforcement Learning model is unable to accurately predict the price of the stock over time.

The second simulation, a payoff structure was added to control this external reward signal. In this case, the Reinforcement Learning Observer must now not only correctly predict the next number, but allocate some points across the payoff functions mentioned in section 4.2.1 Payoff Functions. Each round, the observer got a fixed number of 10 points to invest across these payoff functions. This model is the first model to actually turn a large profit. It also is the first model where the average prediction error over time trends downwards. This suggests that the external reward signals that the payoff functions provide increase the models ability to predict the next number. The model has learned to favor higher-risk strategies, the Linear and NonLinear Payoff Functions denoted in red and blue respectively, when prediction errors are low, as can be seen in Figure 14. During high error periods there is a more balanced allocation. This is especially clear to see when the model notices the prediction error trending upwards. It uses the Linear and NonLinear Payoff Functions less, preferring the Convex and Concave Payoff Functions, denoted in green and purple, respectively. There is no single strategy dominating. The correlation between prediction error and allocation

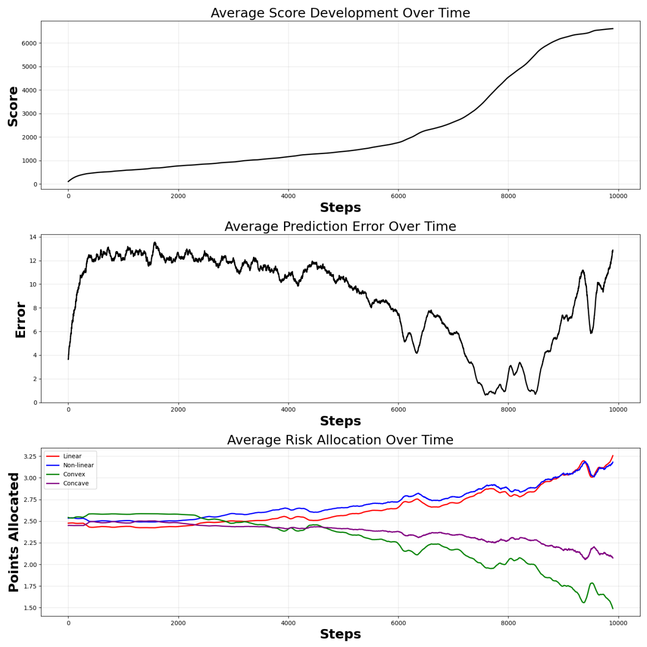

Fig. 14. Results Game with Payoff Structure

strategy suggests the model is learning to correctly balance the risk and reward profiles.

The third and last type of simulation done with the Payoff Game introduced the cash reserve and dynamic bet sizing. The observer has a reserve of previously earned points and is allowed to invest these points in later rounds. As can be seen in Figure 15, the performance

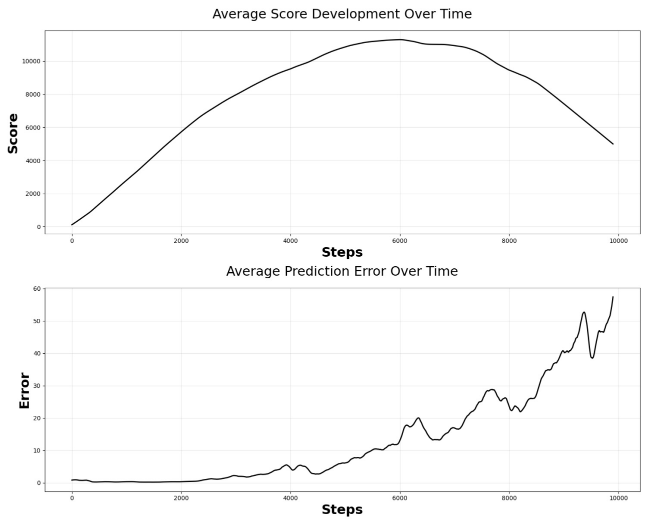

Fig. 15. Results Dynamic Bet Extension

is significantly worse after implementing this extension. The graphs suggest that the observer cannot maintain consistent performance

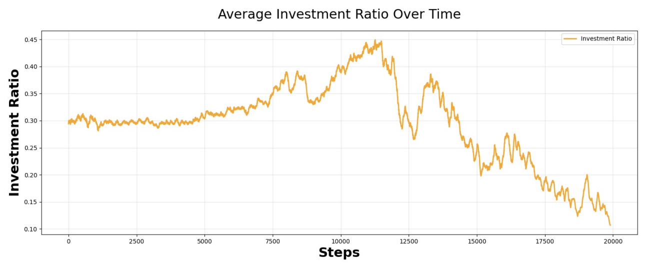

Fig. 16. Results Investment Ratio Dynamic Bet Extension

over time. The investment ratio strategy shows a risk-responsive behaviour, shown in Figure 16. It increases the investment if the score is going up and uses a more defensive strategy when performance deteriorates. This indicates that while the risk management aspects of the observer are functioning as intended, the core prediction capabilities need improvement to maintain performance over time. This dynamic bet sizing does not offer a strong enough reward signal to compensate for wrong predictions.

In all simulations ran, the Reinforcement Learning observers took less than a millisecond to determine its prediction and point allocation.

7 CONCLUSION

This paper explored the performance of Reinforcement Learning in sequential decision-making tasks in low-predictability environments, specifically comparing its performance to other heuristic-based decision-making algorithms across generators of varying complexity. The complexity and predictability of the generators significantly influenced the performance of the Reinforcement Learning models. In simple environments with higher predictability, heuristic algorithms, such as the Fixed Rule or Bernoulli Observer, performed better than algorithms in the Inferring Models Class. One would expect that in environments with greater complexity and less predictability, such as the Stock Data Generator, the Inferring Models would perform better. In the results, it is shown that this is not the case. The counterintuitive results suggest that in the game environments, the ability to ignore noise and focus on fundamental patterns can be more valuable than the ability to try and learn patterns that might not be there. Furthermore, in the Payoff Game, it can be observed that if certain reward signals for correct predictions are adjusted and additional rewards are incorporated, the Inferring Model Class Observers can achieve significantly higher scores and even turn a profit. The rank ordering of generators by predictability generally correlate across observers when the environments are simple and noise-free, as discussed in the results. As complexity and unpredictability increase, however, the correlation starts to differ between different observers as they have varying capabilities to process noise and infer patterns. The rank-order of the generator by predictability correlates with the outperformance of Reinforcement Learning heuristics to some extent, but the relationship is nuanced and depends on the complexity of the environment and the capabilities of the Reinforcement Learning models. In simple environments, the correlation is negative because Reinforcement Learning is often less efficient than heuristics. In more complex environments, the correlation becomes positive as Reinforcement Learning models use their learning capabilities. In very complex environments, the correlation depends on the ability of the Reinforcement Learning approach to adapt to noise and align with the reward structure. To conclude, the complexity and predictability of generators significantly influence model performance. Other heuristic algorithms outperform Inferring Models in simple, predictable environments. Inferring Models might be expected to perform better in more complex environments. However, results show that the ability to ignore noise and focus on fundamental patterns often outweighs attempts to learn nonexistent patterns. Reward signal adjustments, particularly in the Payoff Game, can increase the performance and profitability of the Reinforcement Learning Observer.

8 DISCUSSION

Something that was not taken into account when taking the data from the Stock Generator is the handling of share and stock splits. In financial markets, share and stock splits can significantly change the numerical face value of an asset without changing its economic value. For instance, when a stock has a 2-for-1 split, the price of each share is halved, but the total value of the investment remains the same. The observer, however, only sees a sudden drop in value. Especially observers from the Inferring Models Class might have trouble and learn patterns or scenarios that do not have any real meaning with regards to actual value of the stock. Additionally, the performance of the Reinforcement Learning Observer and Classifier Observer is very sensitive to the choice of hyperparameters. This includes, for example, the learning rate, kernels, temporal pattern sizes, discount factor, and exploration-exploitation balance. While quite some effort was invested in tuning the hyper parameters and reward signals to optimize the performance of both observers, it is important to acknowledge the possibility that the chosen parameters may not be optimal. It can be the case that a completely different set of hyperparameters and reward signals might perform even better.

9 FUTURE WORK

While this research explores the performance of different types of heuristics in low predictability environments, more work can be done to create a better understanding of making predictions based on different generators . This could, for instance, be achieved by adding more context to the model as information to study and see patterns in to support the prediction. One could, for example, create a Natural Language Processing model to scrape the internet about positive or negative messaging, follow political trends, or study yearly financial reports. In addition to adding more contextual information, future work could involve implementing more different types or structures of payoff functions. This would allow the model to explore different strategies to balance risk and reward more effectively. The Reinforcement Learning Model could also be further developed, with its hyperparameters optimized for better performance. Another interesting thing to explore would be training the model on data from different stocks before applying it to predict a new one. This transfer learning approach could reveal how well the model generalizes across different datasets and market conditions.

REFERENCES

[1] Denis Alajbeg, Zoran Bubaš, and Dina Vasić. 2017. Price Distance to Moving Averages and Subsequent Returns. *International Journal of Economics, Commerce and Management* V, 12 (2017), 33–50. https://www.researchgate.net/publication/353120123_Price_distance_to_moving_averages_and_subsequent_returns

[2] Kai Arulkumaran et al. 2022. Reinforcement Learning Textbook. *arXiv preprint arXiv:2201.09746* (2022). https://arxiv.org/abs/2201.09746

[3] William A. Brock and Cars H. Hommes. 1998. Heterogeneous beliefs and routes to chaos in a simple asset pricing model. *Journal of Economic Dynamics and Control* 22, 8-9 (1998), 1235–1274.

[4] Arthur Charpentier, Romuald Elie, and Clément Remlinger. 2023. Reinforcement Learning in Economics and Finance. *Computational Economics* 62, 1 (2023), 425–462.

[5] B. Hambly, R. Xu, and H. Yang. 2023. Recent advances in reinforcement learning in finance. *Mathematical Finance* 33, 3 (2023), 437–503. https://doi.org/10.1111/mafi.12382 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/mafi.12382

[6] Matteo Hessel, Szymon Sidor, Volodymyr Mnih, Jagannath Modayil, Tom Schaul, David Moravák, Martin Wainwright, Demis Hassabis, and David Silver. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. AAAI, 3215–3222.

[7] Aditya Jain, Aditya Mehrotra, Akshansh Rewariya, and Sanjay Kumar. 2022. A Systematic Study of Deep Q-Networks and Its Variations. In *Proceedings of the 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)* (28-29). IEEE, Greater Noida, India. https://doi.org/10.1109/ICACITE53722.2022.9823631 Accessed on: 2024-11-22.

[8] Jacob A. Bikker Laura Spierdijk. 2012. *Mean Reversion in Stock Prices: Implications for Long-Term Investors*. Technical Report. Utrecht University. https://www.uu.nl/sites/default/files/rebo_use_dp_2012_12-07.pdf Accessed: 2025-01-25.

[9] Yuan Li et al. 2022. Reinforcement Learning Algorithms: An Overview and Classification. *arXiv preprint arXiv:2209.14940* (2022). https://arxiv.org/pdf/2209.14940

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Journal of Artificial Intelligence Research* 62 (2015), 529–571.

[11] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2012. *Foundations of Machine Learning*. The MIT Press.

[12] Optiver. [n. d.]. Machine Learning Trading | Optiver. https://optiver.com/working-at-optiver/career-hub/machine-learning-trading/. Accessed: 2024-11-22.

[13] Swapnil Phalke, Yogita Vaidya, and Shilpa Metkar. 2024. Big-O Time Complexity Analysis of Algorithm. In *Proceedings of the IEEE Conference on Algorithm Complexity*. IEEE. https://ieeexplore.ieee.org/document/10007469

[14] Joel L. Schiff. 2011. *Cellular Automata: A Discrete View of the World*. Wiley & Sons, Inc. 40 pages.

[15] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIP Press.

[16] Striim Team. 2024. Training and Calling SGDClassifier with Striim for Financial Fraud Detection. https://www.striim.com/blog/training-and-calling-sgdclassifier-with-striim-for-financial-fraud-detection Accessed: 2025-01-25.

[17] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, and Xin Xu. 2024. Deep Reinforcement Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems* 35, 4 (April 2024), 5064–5078. https://doi.org/10.1109/TNNLS.2022.3207346 Published on 28 September 2022.

[18] Zhikang T. Wang and Masahito Ueda. 2021. Reinforcement Learning: An Overview and Classification. *arXiv* 2106.15419 (2021). https://ar5iv.org/html/2106.15419 Accessed on: 2024-11-22.

[19] Hector Zenil. 2020. A Review of Methods for Estimating Algorithmic Complexity: Options, Challenges, and New Directions. *Entropy* 22, 6 (2020), 612. https://doi.org/10.3390/e22060612 This article belongs to the Special Issue Shannon Information and Kolmogorov Complexity.