Regime aware prediction and RL trading agent

Hunter Gotwalt

May 2025

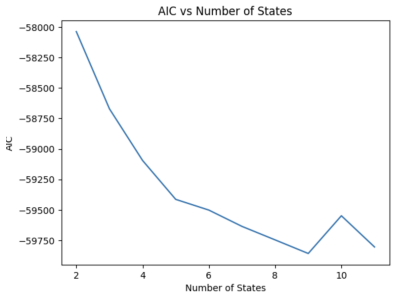
1 Introduction

The goal of this project was to create a data pipeline to allow for better regime aware market predictions and trading. In this project, I combine statistical and deep learning methods in order to predict log returns and simulate trading behavior.

2 Methodology

2.1 Regime Detection

The first thing I did for this project was read in the data of 6 different stocks from the yfinance library. I then calculated the log returns and put the data into a Hidden Markov Model to determine the regime market was in. To determine the optimal amount of states or regimes the model would classify, I had the HMM classify 2-11 regimes, and used the AIC values from the model to determine the optimal amount of regimes for this model. It was determined that 9 regimes would be the best fit



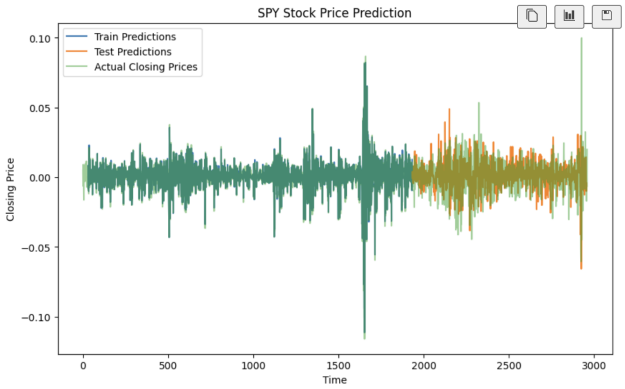
2.2 Monte Carlo Simulation

The original idea was to put the sequence of regimes into the GRU model to predict the next days log return of the SPY, however, when doing this I noticed

it hindered the predictive power of the model, and resulted in less effective generalization on the testing set. So, I decided to run a Monte Carlo simulation to predict the next 30 days, so the model can gain some more insight into what may happen based on this regime. This addition significantly improved the models predictive power.

2.3 GRU prediction

I created a GRU deep learning model in order to predict the next days log return. The data used as input for this model was the previous 30 day log return of the SPY, a normalized version of the previous 30 VIX values, and the 30 Monte Carlo prediction values. The model was 2 GRU layers, each with a hidden size of 32, and then a fully connected layer at the end outputting the predicted log return. The loss function used was the Smooth L1, or Huber loss which allowed for the model to handle the noisy data better. The model yielded a final test loss of .5, a MAE of .0102, and an RMSE of .0140. The prediction output looks like this:

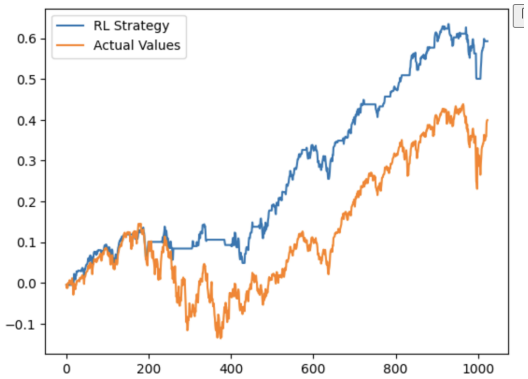


With that, we can see that the model does not capture magnitude as well, but it does to well predicting the direction that the market is going. With these predictions, I was then able to define a threshold of .15, where if the predicted return is greater than that, we buy, and if it's less, we sell, otherwise hold. This strategy slightly outperformed the buy and hold strategy in the SPY as seen below.

However, this threshold-based strategy is rigid and may fail to adapt to changing market conditions or learn more nuanced patterns in the data. It also assumes that a fixed threshold is optimal across all regimes, which is unlikely to be true in a dynamic market. This led me to want to explore the effectiveness of Reinforcement learning.

2.4 Reinforcement Learning

The Reinforcement Learning model I created was a double DQN network that used the same architecture from the GRU model above and same input except for the addition of an in market flag to help the model better determine when to buy hold and sell. The action selection method used was e-greedy with a very slow decay rate. The model outperformed both other strategies slightly.



2.4.1 Problems and Challenges

This reinforcement learning setup encountered several challenges. A key issue was that the agent struggled to recognize whether it was currently in a position,

even with an "in-market" flag and penalties for invalid or redundant trades (e.g., buying when already holding). As a result, the agent often failed to maintain meaningful exposure to the market. To compensate, I adjusted the reward function to encourage more frequent buying in order to keep the agent invested long enough to learn from returns. However, this workaround introduced artificial behavior and biased the strategy toward constant exposure, which undermines its practicality and reliability in real trading scenarios. During testing, it frequently assigned low probabilities to all available actions, resulting in indecisive or erratic behavior. To address this, I implemented a confidence threshold of .5, where if the model's predicted probability for its chosen action was below this, the agent defaulted to a "hold" action. This mechanism helped reduce unnecessary trades and slightly improved overall performance by avoiding low-confidence decisions that often led to poor outcomes.

2.4.2 Key areas for improvement

Since the model struggled with confidence, it likely struggled to distinguish the different states the model was in, so one improvement I would make in the future is create more sophisticated and distinct states by adding more values to the input, or by experimenting with more appropriate data scaling methods. For example, using a RobustScaler instead of a StandardScaler could better handle the presence of outliers and skewed distributions in log returns, leading to a more stable and informative input space for the agent. Another thing that could be improved is the reward function. In it's current form, it heavily emphasizes short term gain, which causes it to make too many trades, and not enough holding. From the models perspective, it made it very susceptible to overfitting, and hindered the performance. From a trading perspective, it opens the company up to more transactions costs, which negates any gain the model may have.

3 Conclusion

This project explored a hybrid approach to financial time series forecasting by combining a Hidden Markov Model for regime detection, a GRU-based neural network for return prediction, and a Double Deep Q-Network for dynamic portfolio decision-making. The pipeline demonstrated moderate success in directional forecasting and showed that reinforcement learning can be applied to develop more adaptive trading strategies. However, the project also revealed critical challenges — including the difficulty of designing an effective reward function, handling model uncertainty, and representing market states with sufficient clarity. Despite these limitations, the project provided valuable insight into the strengths and weaknesses of using deep learning and reinforcement learning in financial environments. It highlighted the importance of balancing prediction accuracy with risk-aware decision-making and demonstrated the need for more interpretable and stable agent behavior. Future work will focus on im-

proving state representation with richer features and volatility-aware scaling, enhancing the reward structure to better reflect realistic trading objectives, and exploring uncertainty-aware methods for more robust decision-making. Overall, this project was a meaningful step toward building intelligent, data-driven trading agents capable of adapting to complex and changing financial regimes.