

A blue parallelogram and a light green parallelogram are positioned on the left side of the slide, overlapping each other and the dark background.

Experimenting with Deep Reinforcement Learning for Self-Training Automated Portfolio Management

Tracy Strickel



Stakeholder: Deutsche Bank

Deutsche Bank has begun cutting jobs in response to the economic environment and their own lackluster profits. Thus far they have fired investment bankers, but are interested in the possibility of liquidating other staff to further reduce costs.

Deutsche Bank has recently learned about reinforcement learning, the third “machine learning paradigm” following supervised and unsupervised learning. In RL learning, a network teaches itself how to perform a task or play a game in an interactive environment through experimentation.

Deutsche Bank has asked bootcamp students to combine a reinforcement learning network with asset management techniques so they can gauge the viability of firing their portfolio managers.



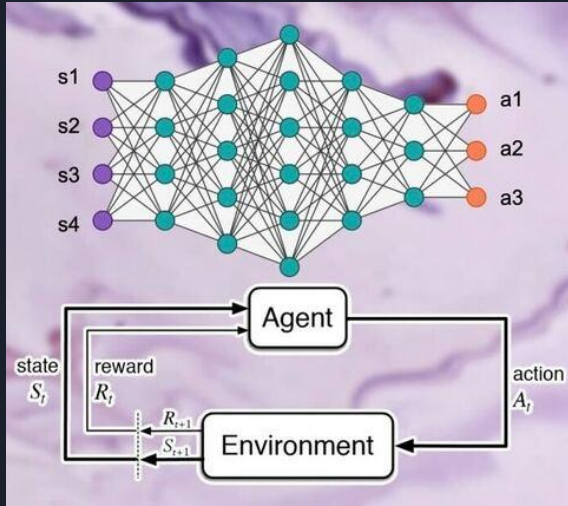
Data Collection

For this project, I created a portfolio of six stocks: Amazon, Costco, IBM, Johnson & Johnson, JP Morgan, and Exxon Mobile. The network will rebalance these stocks once per trading period and attempt to learn the best approach.

Because reinforcement learning requires a massive amount of training data, I used hourly intraday stock prices. I used hourly SP500 futures prices as a proxy for market data. I additionally normalized the daily 1-year treasury rate to give an hourly riskless rate.

Data was purchased from backtestmarket.com. This gave me approximately 28,000 hourly data points from 2007 to this year.


Reinforcement Learning Environment



The first step in reinforcement learning is to create a well-structured 'environment' for the algorithm to experiment in. This is the 'game' the network seeks to optimize. An environment consists of an action space, observation space, and state space.

My observation space, the information the network can see, consists of SPY futures data and standard asset data, ex. volatility, hourly prices, and the previous hour's prices.

My state space, or the information the algorithm can influence, consists of portfolio net worth, asset shares, portfolio weights, and portfolio volatility.



Reinforcement Learning Action and Reward

The action space consists of the possible decisions the network can make at each turn. My model can choose between two standard portfolio balancing techniques - the Sharpe ratio and the maximum diversification technique. These work with similar information, but Sharpe is risk-greedier.

This allows the model to alternate between the riskier method and the less risky method in response to market data. In theory, this should minimize losses.

The network also needs a 'reward', which provides feedback on whether it is making good or bad decisions. My model's reward is the portfolio return per timestep minus the market return for the portfolio assets.



Reinforcement Learning Models

I split my data into train and test sets and fed it through three reinforcement learning algorithms, A2C, PPO, and TRPO.

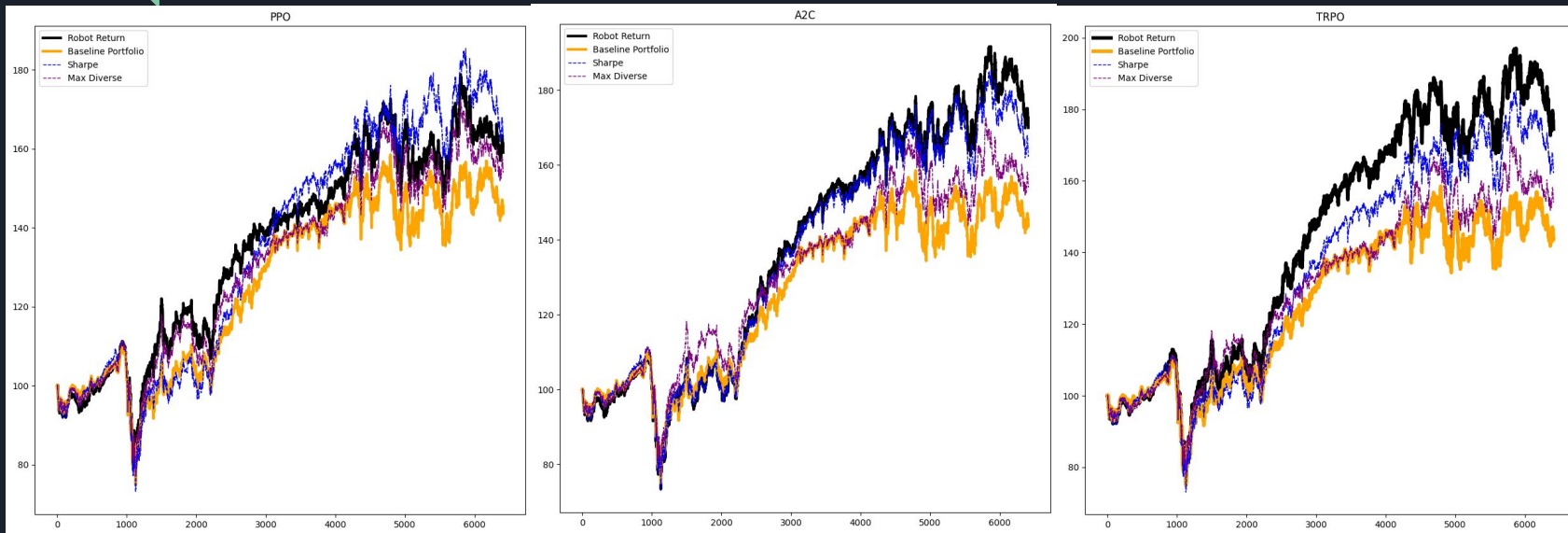
A2C is an actor-critic model in which one network, the actor, experiments with choices while another network, the critic, provides feedback on how good it thinks these choices are.

PPO and TRPO are both policy gradient methods, which means they search the space of possible choices without trying to assign a specific value-map to how good these choices are.

I used a perfectly balanced portfolio as a baseline, with $1/6$ th capital invested in each of the six assets across the entire timespace.

Results

Four years of trading, SPY up +31%



PPO return: 59.1%

A2C return: 70.0%

TRPO return: 74.5%

PPO vs. baseline: +10.6%

A2C vs baseline: +18.2%

TRPO vs. baseline: +21.3%



Next Steps

- More data! Creating a truly robust model would likely require generation of synthetic training data reflecting various market environments.
- Revisit the reward metric - model behavior is highly sensitive to the specific way reward is calculated. Add an error term to encourage exploration - the models tend to cluster towards a favored action.
- Revisit the TRPO model with more hyperparameter tuning, etc. TRPO had the most robust results and seemed to be learning the correct balance between actions.



Thank you!

Contact:

tracystrickel@gmail.com

github.com/hypermoderndragon

linkedin.com/in/tracystrickel