Fall 2018

Deep Reinforcement Learning in Trading Algorithms

Tucker Bennett
Kenyon College, bennett1@kenyon.edu

Delaney Ambrosen
Kenyon College, ambrosend@kenyon.edu

Joe Woody
Kenyon College, woodyj@kenyon.edu

Simon Fruth
Kenyon College, fruth1@kenyon.edu

Deep Reinforcement Learning in Trading Algorithms

Tucker Bennett, Delaney Ambrosen, Joe Woody, and Simon Fruth
Artificial Intelligence for the Humanities
Kenyon College

Abstract

An algorithm that can learn an optimal policy to execute trade profitable is any market participant's dream. In the project, we propose an algorithm that does just that: a Deep Reinforcement Learning trading algorithm. We design our algorithm by tuning the reward function to our specified constraints, taking into account unrealized Profits and Losses (PnL), Sharpe ratio, profits, and transaction costs. Additionally, we use a short 5-month moving average replay memory in order to ensure our algorithm is basing its decision on the most pertinent information. We combine the aforementioned concepts to make a theoretical Deep Reinforcement Learning trading algorithm.

Background Information

The use of Artificial Intelligence and Machine Learning on Wall Street is becoming increasingly predominant. In recent years, AI has become a cornerstone strategy for trading and research in large asset management companies such as Two Sigma and Goldman Sachs. Other quant firms like Acadian Asset Management have more than doubled their assets in the last five years due to strategies implemented by AIs.

Artificial Intelligence is currently taking on several different roles in the financial sector. Companies like Goldman have already implemented it in a lending and banking platform for individuals called "Marcus." Other firms such as JP Morgan Chase have been investing in applications and algorithms that will eventually optimize trade execution. These algorithms and AIs will be considered successes if they reduce market impact, and provide the best trading execution decisions. They will do this by "learning" the best actions based on the market and client preferences.

Currently 45% of Goldman Sachs's revenue is comprised of cash equities trading that executed by trading algorithms. They have found that four traders can be replaced by one computer engineer, with almost a third of the staff already consisting of engineers. This trend of traders and analysts being replaced by programmers and engineers will only grow as managing directors and executives find that AI is less costly than lower-level employees. It is projected that machine learning and Artificial Intelligence will replace over 90,000 jobs in asset management in the United States, and over 300,000 jobs worldwide.

Algorithm Comparison

Three main techniques rising in prominence in algorithmic trading: Genetic Algorithms/Programming (GA), Generative Adversarial Networks(GAN), and Deep Reinforcement Learning(RL).

Genetic Algorithms (GA)
Genetic Algorithms are a set of algorithms that mirror the algorithm of evolution. Across multiple generations, a population of competing algorithms continually mutate over time and hopefully converge on an efficient solution. This approach works well for large problem spaces that are underspecified, like those typically found in finance.

Pros: It can be used on a broader set of problems that would be ill defined for traditional algorithms. Genetic Algorithms generally converge on efficient solutions given enough time and resources.

Cons: The algorithm is not fully transparent, it may take a lot of resources to eventually arrive at an acceptable solution, ultimately it may not converge on an acceptable solution at all.

Generative Adversarial Networks(GAN)
Generative Adversarial Networks use two competing networks to learn a model. A GAN consists of a generative network that creates synthetic data intended to deceive a discriminator network. For example, GAN's can be used to generate realistic photos from random data. In this situation, a discriminator network trained on real photos is fed images created by the generative network. After many iterations, the generative network creates increasingly realistic photos until the discriminator can no longer distinguish between real and synthetic photos. GAN's can learn the latent structure in any data from photos to probability distribution of stock returns.

Pros: Effective method for learning latent structure in complex data.

Cons: It can be difficult to effectively train GAN's.

Deep Reinforcement Learning(RL)
Deep Reinforcement Learning is a technique to learn an optimal policy or sequence of actions to take in an uncertain environment. An RL agent interacts with an environment and receives reward or penalties based on their interaction with the environment. As RL agents become more familiar with their environment, they have a decision between exploiting actions with known rewards versus exploring the environment to learn new actions with potentially higher rewards. As the agent learns an increasingly optimal policy, they are able to maximize its rewards. Reinforcement Learning is behind technology lie alpha go achieving breakthroughs that humans, until recently, thought were impossible.

Pros: Can do effective learning without large labeled data sets.

Cons: It can be difficult to design realistic reward functions and capture all the possible outcomes for a given scenario.

Reinforcement Algorithm Methodology

Along with Genetic Algorithms, Reinforcement Learning and Generative Adversarial Networks have been methods used to implement algorithmic trading in the past, but recently Deep Neural Network (DNN) approaches to Reinforcement Learning (RL) have garnered more attention recently. This approach most naturally replicates the actions of a human trader in that it also constantly deals with the trade off between exploiting known returns versus exploring risky but potentially higher returns by pursuing new actions with unknown rewards (see Figure 1). This also has the advantage of being an end to end unsupervised AI model that results in decisions rather than predictions. The algorithm dynamically solves a sequential decision problem by interacting with its environment to learn the most efficient policy based upon the rewards of individual actions. By continually interacting in a dynamic environment, the algorithm is able to adapt to changing market signals and structures.

The DNN learns the most efficient trading policy based upon the rewards for every action our agent takes, balanced with the objective to both exploit known profitable actions with exploring unknown potentially more profitable actions. The resulting policy informs the RL algorithm on how to make the most efficient trading policy (see Figure 2). At any given point in time, there are three possible actions for the agent to make: Buy, Sell, and Hold. At every point decision point, the algorithm runs multiple simulations for each possible action to estimate the returns for each potential sequence of actions. This gives the agent a recommended sequence of actions to exploit for maximum returns based upon returns from past transactions. The trading policy evolves as the RL algorithm adapts to changing market conditions in real time.

The policy is updated with each action taken by the algorithm through a reward function. The reward function is something that is unique to Reinforcement Learning algorithms and makes it extremely applicable to trading strategies. The feedback at each time step is essential in learning the error at each step and adjusting the policy accordingly. This policy can then be used by the RL algorithm to make real time trades in the market. As the algorithm interacts with the training data, it receives reward signals.

The reward function can be customized to best fit the goals of the trader. For example, the algorithm can be trained to maximize profits and minimize losses, or to maximize the Sharpe ratio, which is a measure of risk-adjusted returns. The algorithm can also be trained to account for unrealized profits and losses (PnL), which is something other strategies do not account for. Unrealized profits and losses come from assets that appreciate, or depreciate in value, but are not yet sold. Therefore, their change in value has not been realized. Including unrealized PnL would allow the algorithm to learn from its inaction and improve its trading policy. Finally, the reward function incorporates transaction costs, which is a vital consideration to make. This ensures that the algorithm does not make frequent frivolous trades.

Figure 1

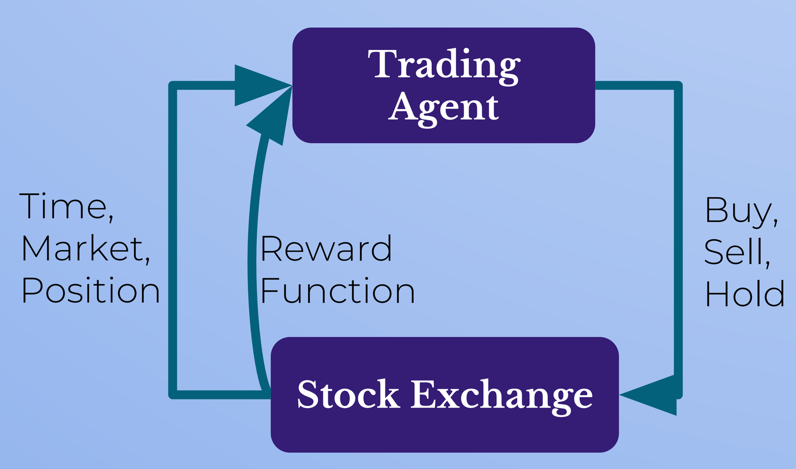

Figure 1 represents how an agent interact with its environment in Reinforcement Learning

The reward function we designed includes minimizing unrealized PnL, maximizing Sharpe ratio, maximizing profits, and especially minimizing transaction costs which is frequently overlooked. Our reward function helps increase the efficiency of the algorithm by decreasing random exploration. Since Deep Reinforcement Learning "learns" through trial and error, it takes many actions for it converge on an optimal policy which we must constrain by accounting for transaction costs. The algorithm finds an optimal policy to fit all the constraints on its own.

Our program would utilize a 5-month moving average replay memory window to influence the algorithm's decision. This shorter time length ensures that the program is not overfitting the data, and is considering only the most relevant information. As the replay memory window increases in time, the incidence of overfitting increases. This way, the most recent data points have the biggest impact on the algorithm's decisions, allowing the algorithm to "forget" old data and not influence its future trading decisions. This time window is supported by research by Tucker Balch at the Georgia Tech College of Computing. An example of how we would code the replay memory window is shown in Figure 3.

After millions of iterations of training on historical data, the policy ultimately converges on an optimal policy. Our algorithm would optimize the portfolio's Sharpe ratio, consider trading costs and unrealized profits and losses, and would utilize a 5-month memory window. We believe that this methodology would produce the most efficient trading strategy, and could be employed trading small-cap stocks. We decided to choose small cap stocks because they are the most overlooked securities and they present the best hidden buying opportunities. The small caps have a higher volatility that benefit models like these that can better detect and profitably act upon changes in real time. Additionally, small caps tend to outperform large cap stocks in the long run due to their increased risk and we feel this algorithm could identify successful buying opportunities given its parameters.

Figure 2

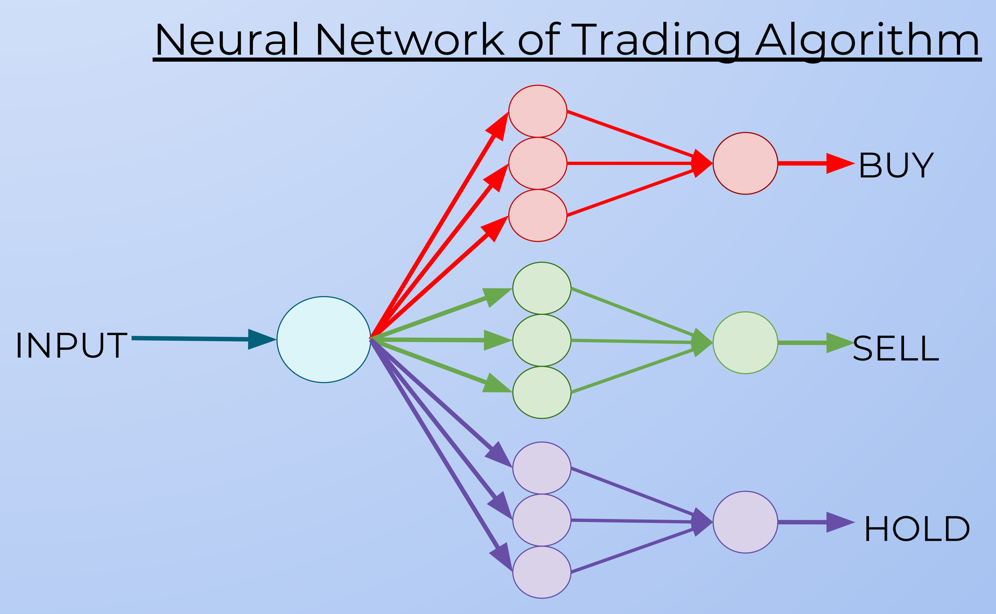

Figure 2 represents how the Deep Neural Network informs the RL algorithm what action to take

Figure 3

```
class ReplayMemory:
    '''Memory buffer for experience replay'''

    def __init__(self, logger, config):
        self.logger = logger

        self._model_dir = join(config[SAVE_DIR], REPLAY_MEMORY)

        self.batch_size = config[BATCH_SIZE]
        self.history_length = config[HISTORY_LENGTH]
        self.memory_size = config[MEMORY_SIZE]
        self.num_channels = config[NUM_CHANNELS]
        self.dims = (self.num_channels,)

        self.actions = np.empty(self.memory_size, dtype = np.uint8)
        self.rewards = np.empty(self.memory_size, dtype = np.float32)
        self.screens = np.empty((self.memory_size, config[NUM_CHANNELS]), dtype = np.float32)
        self.terminals = np.empty(self.memory_size, dtype = np.bool)
        self.trades_rem = np.empty(self.memory_size, dtype = np.float32)

        # pre-allocate prestates and poststates for minibatch
        self.prestates = (np.empty((self.batch_size, self.history_length, self.num_channels),
                                   dtype = np.float32),\
                                   np.empty(self.batch_size, dtype=np.float32))
        self.poststates = (np.empty((self.batch_size, self.history_length, self.num_channels),
                                   dtype = np.float32),\
                                   np.empty(self.batch_size, dtype=np.float32))

        self.count = 0
        self.current = 0

    def add(self, screen, reward, action, terminal, trade_rem):
        if screen.shape != self.dims:
            self.logger.error(INVALID_TIMESTEP)

        else:
            self.actions[self.current] = action
            self.rewards[self.current] = reward
            self.screens[self.current, ...] = screen
            self.terminals[self.current] = terminal
            self.trades_rem[self.current] = trade_rem
            self.count = max(self.count, self.current + 1)
            self.current = (self.current + 1) % self.memory_size
```

Figure 3 is an example of how to encode replay memory into our algorithm.

Conclusion

Artificial Intelligence will inevitably revolutionize the trading models used on Wall Street. As computing power and Machine Learning techniques improve, these strategies are becoming more profitable and accessible not only for large Wall Street firms, but for individual investors. Mahi de Silva, the founder and CEO of Botworx.ai even said, "AI systems that were once only accessible to large hedge funds and megabanks [will serve] a much broader set of customers including day traders that make a living from understanding and reacting to patterns of the financial markets." As these high-power strategies become more accessible to everyday people, financial markets will become more fair and available to the general public. While trading algorithms have shown success on Wall Street, they are still not perfect.

Algorithms can react to changes in price much more rapidly than humans can, exploiting small changes in price for quick profits. However, this can have a negative effect on the market. Some of the recent stock market volatility can be attributed to the sell signals encoded into these algorithms in order to minimize loses. This has caused some algorithms to have to be shut off during volatile trading sessions. Due to issues such as the aforementioned encoded sell signal, the majority of current trading algorithms still require human supervision. This is due impart to humans' lack of trust in machines to make financial decisions. Artificial Intelligence currently operates in somewhat of a black box but tremendous efforts are being made to bring fairness, accuracy, and transparency(FAT) to complex models like Deep Reinforcement Learning. As trading algorithms proliferate in the future, humans must come to terms of understanding sophisticated trading algorithms: the human traders will shift from designing trading algorithms to understanding automated training algorithms.

Resources

- *Bloomberg.com*, Bloomberg, www.bloomberg.com/news/features/2017-12-05/how-ai-will-invade-every-corner-of-wall-street.
- Brackenridge, Gary. "Artificial Intelligence Is Transforming Investment Strategies." *CNBC*, CNBC, 6 June 2017, www.cnbc.com/2017/06/06/machine-learning-transforms-investment-strategies-for-asset-managers.html.
- Britz, Denny. "Introduction to Learning to Trade with Reinforcement Learning." *WildML*, 11 Feb. 2018, www.wildml.com/2018/02/introduction-to-learning-to-trade-with-reinforcement-learning/.
- Byrnes, Nanette. "Traders Are out, Computer Engineers Are in, as Goldman Sachs Goes Digital." *MIT Technology Review*, MIT Technology Review, 19 Apr. 2018, www.technologyreview.com/s/603431/as-goldman-embraces-automation-even-the-masters-of-the-universe-are-threatened/.
- Coles, Terri. "How AI Trading Systems Will Shake Up Wall Street." *IT Pro*, 21 Mar. 2018, www.itprotoday.com/machine-learning/how-ai-trading-systems-will-shake-wall-street.
- Coles, Terri. "How AI Trading Systems Will Shake Up Wall Street." *IT Pro*, 21 Mar. 2018, www.itprotoday.com/machine-learning/how-ai-trading-systems-will-shake-wall-street.
- "Google Colaboratory." *Google*, Google, colab.research.google.com/gist/gjlr2000/15da1e50cebf856cdcea9adfdd655d31/qtrader.i-pynb#scrollTo=0seJSxYgamuh.
- Harwell, Drew. "A Down Day on the Markets? Analysts Say Blame the Machines." *The Washington Post*, WP Company, 6 Feb. 2018, www.washingtonpost.com/news/the-switch/wp/2018/02/06/algorithms-just-made-a-couple-crazy-trading-days-that-much-crazier/?noredirect=on&utm_term=.85980f889e6b.
- Jonathan. "Developing High Performing Trading Strategies with Genetic Programming." *QUANTITATIVE RESEARCH AND TRADING*, 9 Sept. 2018, jonathankinlay.com/2018/09/developing-trading-strategies-with-genetic-programming/.
- Research, Inc. Lucena. "Applying Deep Reinforcement Learning to Trading with Dr. Tucker Balch." *YouTube*, YouTube, 26 Sept. 2018, www.youtube.com/watch?v=Pka0DC_P17k.
- samre12. "samre12/Deep-Trading-Agent." *GitHub*, 14 Apr. 2018, github.com/samre12/deep-trading-agent.
- Zhou, et al. "Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets." *Advances in Decision Sciences*, Hindawi, 15 Apr. 2018, www.hindawi.com/journals/mpe/2018/4907423/.

Special thanks to Professor Elkins and Professor Chun