Algorithmic Stock Trading

Contributors - Ashish Vinodkumar, Charlie Xie, Jasmine Young, Leon Zhang

Abstract

Algorithmic trading has become a popular strategy to trade more successfully with the assistance of computers and machine learning. We evaluate several algorithmic strategies on Apple, Amazon and SPY (S&P 500 Index fund) over a recent time period. We built an agent framework that begins with an initial amount of money, makes buy/sell decisions according to the selected strategy, and reports a return on investment at the end of the test period. We tested our agent with technical indicators (SMA, RSI, MACD, and ADX) and machine learning models (LSTM, ARIMA, Deep Q Learning, and Evolution Strategy). We found that the success of each strategy varies wildly depending on the stock studied and the time period. Our machine learning methods fared better than our technical indicators in most cases.

Introduction

Algorithmic trading executes trade orders based on pre-programmed instructions without human intervention. This form of trading has gained increasing popularity. In 2019, 70% of shares traded on the U.S stock exchanges were placed automatically. (Folger, J. 2020, August 28)

Algorithms have many advantages over human traders. It is a completely passive form of earning money. Once the system is built and deployed, humans no longer need to stay in front of the screens. Moreover, algorithms are detached from emotions, allowing trading to be more consistent. For these reasons, we applied machine learning techniques to this trending topic.

Algorithmic trading takes many forms. In the past, people preferred technical indicators. In recent years, the growing field of machine learning has attracted traders to explore more advanced techniques. In this project, we built trading agents using indicators, supervised learning, and reinforcement learning methods to maximize investment.

Background

We researched and implemented the following machine learning techniques:

Hochreiter and Schmidhuber(1997) initiated Long Short-Term Memory (LSTM) Network, and it has been utilized in language modeling and machine translation. LSTM was leveraged to predict the stock returns of NIFTY 50 with 5 years of historical data.(Roondiwala et al., 2015). Meanwhile, Chen, Zhou and Dai(2015) applied the model to Chinese stock market, and achieved

the improvement in accuracy of predicting stock returns. Furthermore, attention-based LSTM model was utilized for predicting stock price movement.(Cheng et al., 2018)

Box and Jenkins (1976) developed the Autoregressive Integrated Moving Average (ARIMA) model and it has been used in forecasting. Mondal, Shit and Gaswami (2014) studied the effectiveness of the ARIMA model on Indian stocks. In addition, Amry and Siregar(2019) built the model for forecasting Indonesia Composite Index.

Reinforcement Learning(RL) algorithm was applied to make investment policies by Moody and Saffell(2001). It was then to facilitate trade in foreign exchange markets (Forex) (Dempster and Leemans, 2006). Most recently, researchers were trying to apply the deep Q-learning algorithm to transactions in Forex. (Carapuco et al., 2018)

Evolution Strategy(ES) was initiated by Rechenberg(1973) and Schwefel(1977) who were inspired by natural evolution. Korczak, Lipiński and Roger(2002) were trying to leverage this strategy to optimize a stock portfolio taken from the French market. Myszkowski and Bicz(2010) applied the strategy to trade futures contracts on Forex. Most recently, meta-learning algorithms were applied to evolutionary strategy in stock trading, and it achieved less learning time with more trading profits. (Sorensen, et al., 2020)

Data

We pulled stock price data for SPY (S&P 500 index fund), Apple, and Amazon, from Yahoo Finance API. Our motivation for picking these stocks is as follows:

- S&P500: Widely regarded as the best possible measure of large-cap US equities and is a market capitalization weighted index of the 500 largest publicly traded companies in the US.
- Apple: Multinational hardware and software company designing consumer electronics with relative steady increase in stock prices until 2019, and exponential growth after.
- Amazon: Multinational company that specializes in e-commerce, cloud computing, digital streaming, and artificial intelligence with rapid doubling in stock prices, and exponential growth post 2019.

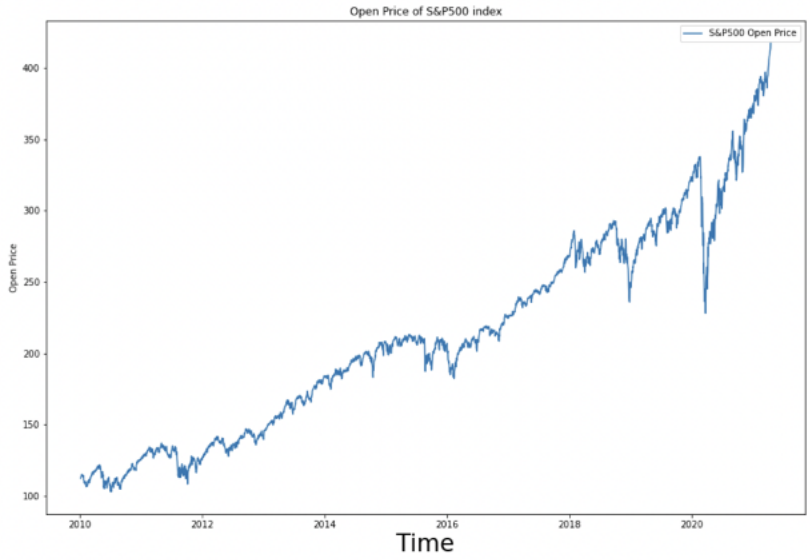


Figure 1. S&P500 stock price shows gradual increase over time, with extreme fluctuation around 2020.

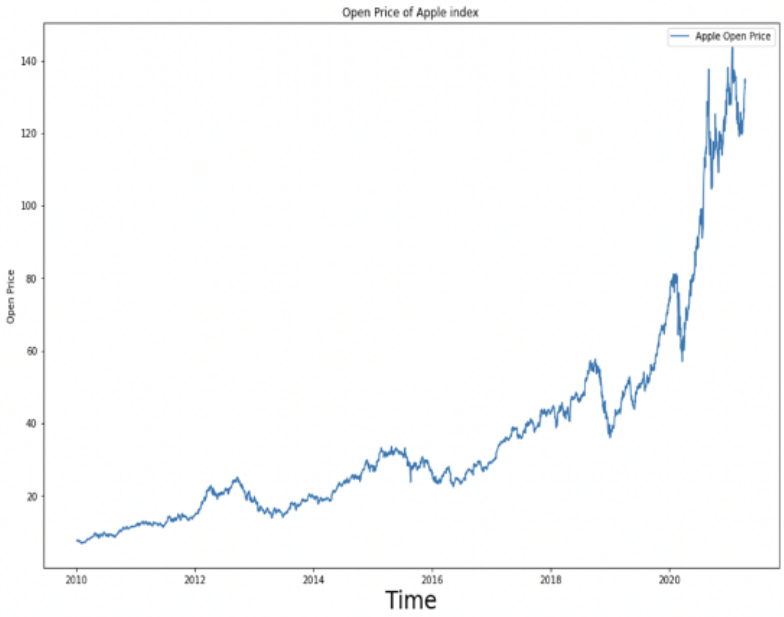


Figure 2. Apple stock price ranges from from $20 -$60 up until mid-2019, and experienced exponential growth post mid-2019

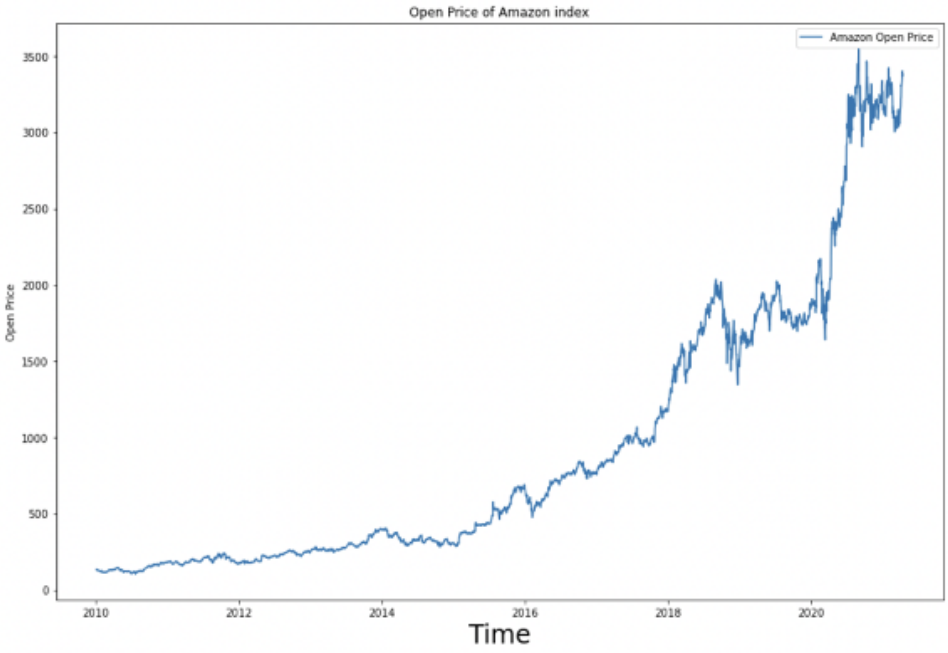


Figure 3. Amazon stock price almost doubling every 3-5 years with a surge in growth post 2019.

As detailed in the plot labels above, both Amazon and Apple experience a surge in stock prices mid/post-2019 and this brings interesting complications to our model training process as the agent will have to learn over time when it will be best to buy, sell, or hold stocks due to the large increases in evaluation. Due to the volatility of the stock market and sometimes external factors such as recessions, creating a reliable automated stock bot is difficult. In order to combat and overcome these difficulties, we have chosen a start date of 2010-01-04. We further wanted to maximize the available training data and as result, chose to perform a roughly 90-10 train-test split. This gave us 90% of the data under each stock for the model to train on, and the remaining 10% data to perform testing. This design yielded the following dates:

Table 1. Training and Testing Data Timeframes

	Dates
Training Dataset (S&P500, Apple, & Amazon)	2010-01-04 - 2020-08-06
Testing Dataset (S&P500, Apple, & Amazon)	2020-08-07 - 2021-04-16

Methods

We first built trading agents that utilize technical indicators as trading strategies. Indicators such as SMA, MACD, RSI, and ADX are commonly used by many quantitative traders to evaluate buy and sell signals. We believe implementing these strategies is important because it provides us a good benchmark to compare with agents that integrate more advanced machine learning techniques later on.

To compare agent performance, we introduce the term - return of investment (ROI):

$$\text{ROI} = \frac{\text{Current Value of Investment} - \text{Cost of Investment}}{\text{Cost of Investment}}$$

Equation 1. Return of Investment

Baseline Agent
Baseline agent uses a buy and hold strategy. It invests all available money on the first day and holds till the last trading day. This agent's ROI is used as a baseline score for us to compare with other agent's performance.

I. Technical Indicators

SMA

SMA, smooth moving average, averages recent prices over a time window. SMA smooths out volatility. A popular strategy is to create a 20 days short-term and a 50 days long-term SMA. If the 20-SMA passes above the 50-SMA, an uptrend is expected, so it's an appropriate buy signal. If the 20- SMA goes below the 50-SMA, a downward trend is expected, so it's an appropriate sell signal. (Raudys, A., & Rabarskatie, Z., 2018)

MACD

Moving Average Convergence Divergence (MACD) is a trend-following momentum indicator. It is calculated by "subtracting the long term Exponential Moving Average (EMA) (26-periods) from the short term EMA(12-periods)." Meanwhile, a signal line, a nine-day EMA, is created for showing buy and sell signals. Investors may buy the stock when the MACD crosses above its signal line and sell the stock when the MACD crosses below the signal line. (Wang and Kim, 2018)

RSI

The Relative Strength Index (RSI) is a technical analysis tool used to evaluate overbought and oversold conditions in the price of a stock. Traditionally, RSI values above 70 mean a stock is overvalued. An RSI below 30 is considered evidence that a stock is undervalued. The RSI formula calculates the average percentage gain or loss over a certain look-back period - typically 14 days, before smoothing (Bhargavi, 2017).

ADX

Average Directional Index (ADX) is a technical indicator used to determine the strength of a trend, comprising of a Negative (-DI), a positive (+DI) and an Average (AD) Directional Index, all of which are used to indicate best predicted moments to buy, sell, or hold a given stock. When ADX is over 25, it indicates a strong trend, and when the ADX is over 20, it indicates a weak trend (Gurrib, 2018).

II. Machine Learning Methods

ARIMA Agent (Detailed explanation in appendix)

ARIMA is another common technical analysis tool that uses time-series data to predict future values of a given stock.

ARIMA models have 3 main parameters:

1. Auto-Regressive (AR): We calculated 'p' - the number of lags to be used as predictors.
2. Integrated (I): We calculated 'd' - the number of differences required for stationarity.
3. Moving Average (MA): We calculated 'q' - the number of lagged forecast errors.

We chose to use Auto-ARIMA to determine the optimal parameters (p, d, and q) for each model. We used a training period from 2010 until the beginning of our test period (8/7/2020) on the ARIMA model fitted with Auto-ARIMA hyperparameters.

After training the ARIMA model, we used it to predict stock prices for the test period of August 7th, 2020 to April 16th, 2021. The agent would look for a >1% increase before buying and a >1% decrease before selling. Otherwise, the agent would hold the stocks for the upcoming day. This strategy resulted in the ROI's that will be discussed in the results section.

LSTM Agent

We utilized Roondiwala et.al (2015) and Chen et.al (2015) to reference prior work in the field to provide a baseline LSTM model. We then tuned this model by performing a greedy hyperparameter search to identify the optimal number of LSTM layers, dense layers, and nodes, with the intention of bringing down the validation root mean squared error (RMSE). Through iterative model tuning, we ended up with the following model configurations:

- S&P500: LSTM(128), LSTM(70), LSTM(50), Dense(25), Dense(1)
- Apple: LSTM(128), LSTM(64), Dense(25), Dense(1)
- Amazon: LSTM(230), LSTM(100), LSTM(70), LSTM(50), Dense(25), Dense(1)

These models were trained on 93.84% of the stock data from 2010-2020, and 6.16% of the most recent trading days were used for testing. The model trained for 5 epochs and took in stock opening prices from past 60 days to predict the following day's stock opening price. Based on these predictions, an agent with the following strategy was implemented:

- If projected gain for tomorrow's price is
    - over 2%, then buy (if can buy).
    - less than -2%, then sell (if can sell).
- Else, hold position.

Deep Q Learning Agent (Detailed explanation in appendix)

Stock trading naturally fits into the reinforcement learning framework. Recent advances of reinforcement learning have shown people its ability to tackle difficult problems. It would be interesting to see how it performs on trading. We used Deep Q-learning to address the infinite state space issue. (Cartea, Á, Jaimungal, S., & Sánchez-Betancourt, L. 2021) (Gao, Xiang 2018)

The agent has 3 actions: buy, sell, and hold. When the agent chooses an action, the environment checks if the action is allowed. If action is confirmed, the environment steps one day forward and returns the reward and the new state's observation (past 60 days of opening price). The flowchart of environment, reward setting, and agent training procedure figures are shown below.

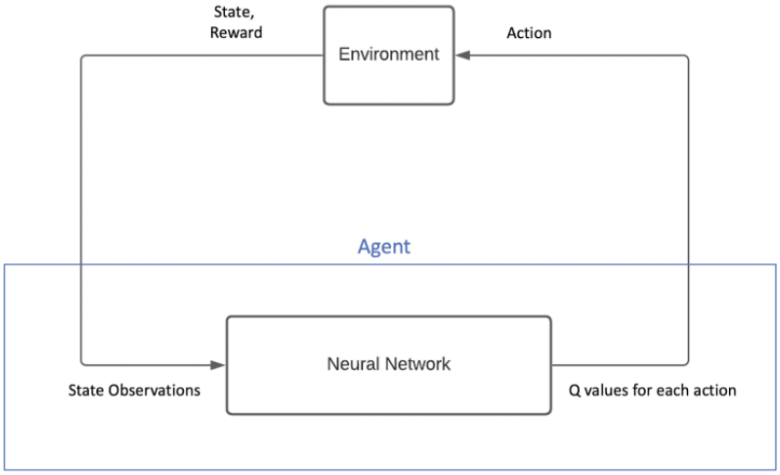


Figure 4. Flow Diagram of Deep Reinforcement Learning

If action is to buy:

$$R(S,A) = 100 \times \frac{Price_{t+1} - Price_t}{Price_t}$$

If action is to sell:

$$R(S,A) = -(100 \times \frac{Price_{t+1} - Price_t}{Price_t})$$

If action is to hold:

$$R(S,A) = 0$$

Equation 2. Reward Assignment for Each Action

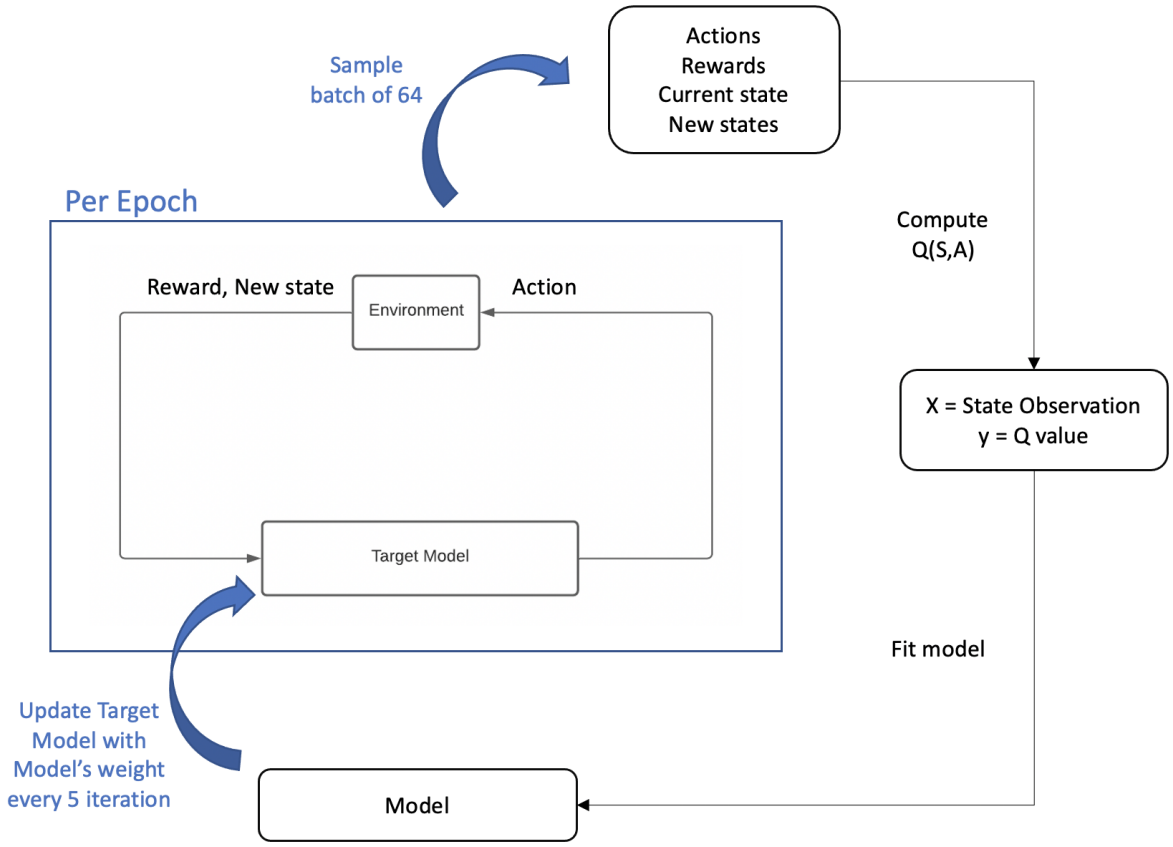


Figure 5. Training Procedure of Q-learning Agent

As can be observed in Figure 5, the training runs for 500 episodes. In each episode, the agent is given $1000 and interacts with the environment until it loses 5% of initial investment. The process records the current state observations, the action taken by the agent, its reward, and the next state observations into a deque of size 3000 and samples a batch of 64 records. Each record is then used to compute Q-value of that state-action pair. The state observations become the input for training and the Q-value becomes the y-label. After processing for 64 samples, they are then fed into the model for fitting. Every 5 episodes, the target model is updated to have the same weights as the fitted model for stability enhancement.

Evolution Strategy Agent (Details in appendix)

The idea of evolution strategy (ES) is to take samples from a population of individuals and let the successful ones guide the distributions of future generations. The figure below shows an ES optimization process. The white dots represent the internal parameters of the current model, and the black dots around are the sampled noise parameters. A return vector is calculated for all the noise parameters, by which the internal parameters can be updated to a better value through iterations.

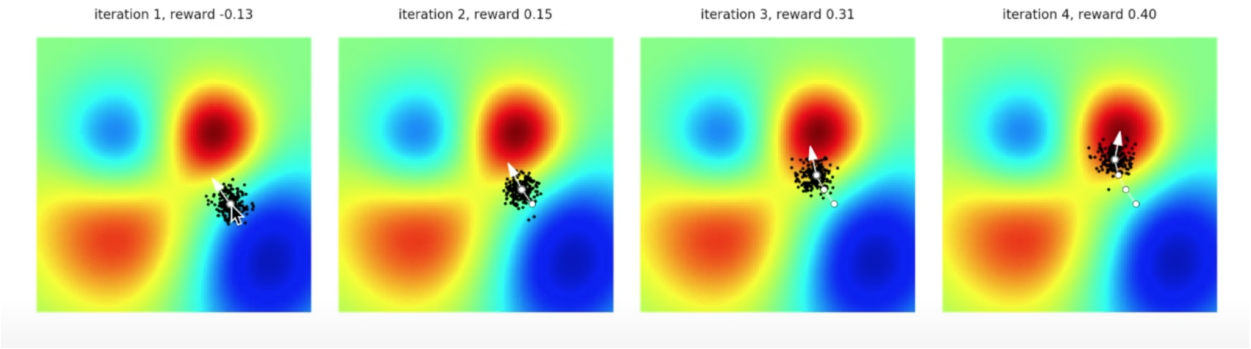


Figure 6. Visualizing an ES Optimization Process
(Source: https://openai.com/blog/evolution-strategies/)

Utilizing this black-box optimization method, we are optimizing a function f(w) with respect to the input parameters w, but we make no assumptions about the structure of f. Therefore, it is a fascinating tool to capture the characteristics of security prices movement which is hard to model. Our model includes three components: an ES algorithm that is employing Gaussian as search distribution, updating parameters iteratively(See table below); a neural network that is to predict buy or sell signals based on the updated parameters; an agent which is to implement the trading decisions and calculate the ROI.

Table 2: Evolution Strategy Algorithm Implementation

Algorithm 1 Evolution Strategies

Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
  Sample $\epsilon_1, \ldots \epsilon_n \sim \mathcal{N}(0, I)$
  Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$ for $i = 1, \ldots, n$
  Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$
end for

(Source: https://arxiv.org/abs/1703.03864)

Results

I. Technical Indicators

The below table summarizes the ROI of each of the indicator agents on SPY, AAPL and AMZN stock during the testing period:

Table 3. Technical Indicator Agents' ROI on Test dataset: 7/8/2020 - 16/4/2021

	SPY	AAPL	AMZN	Average
SMA	+14.2 %	-0.7 %	-8.4 %	+ 2.6 %
MACD	+ 4.1%	+ 1.1%	- 6.2%	- 0.3%
RSI	+5.9%	+9.2%	-4.1%	+3.7 %
ADX	+5.49%	55.52%	24.97%	+28.66%
Baseline	+24.1 %	+18.2 %	+4.6 %	+15.6 %

II. Machine Learning Methods

ARIMA

We evaluate ARIMA's performance on Apple, Amazon, and the S&P 500 over our time period of interest. We also look at the RMSE normalized by initial stock price. The ROI performance is listed in a table below with our other machine learning methods. The RMSE values are shown here:

Table 4. ARIMA's Test RMSE Normalized Comparison across Selected Stocks

	APPL	AMZN	SPY
RMSE	1.05	0.99	1.08

When we normalize by opening stock price we see that the ARIMA model has the lowest RMSE with Amazon, and the highest RMSE with the S&P 500. The ARIMA model also had the lowest ROI with the S&P 500 - which lines up with the high RMSE.

LSTM

For the S&P 500, Apple, and Amazon LSTM models, we evaluated performance based on RMSE over the test period of interest. As it can be observed in the below table, the test RMSE for Apple and S&P500 is comparatively really low at around ~5. However, the test RMSE for Amazon is relatively high at ~128. Even after greedy hyperparameter search, we could not bring the test RMSE further down for Amazon.

Table 5. LSTM's Test RMSE Normalized Comparison across Selected Stocks

	APPL	AMZN	SPY
RMSE	4.21	128.65	5.92

Deep Q-Network (Detailed explanation in appendix)

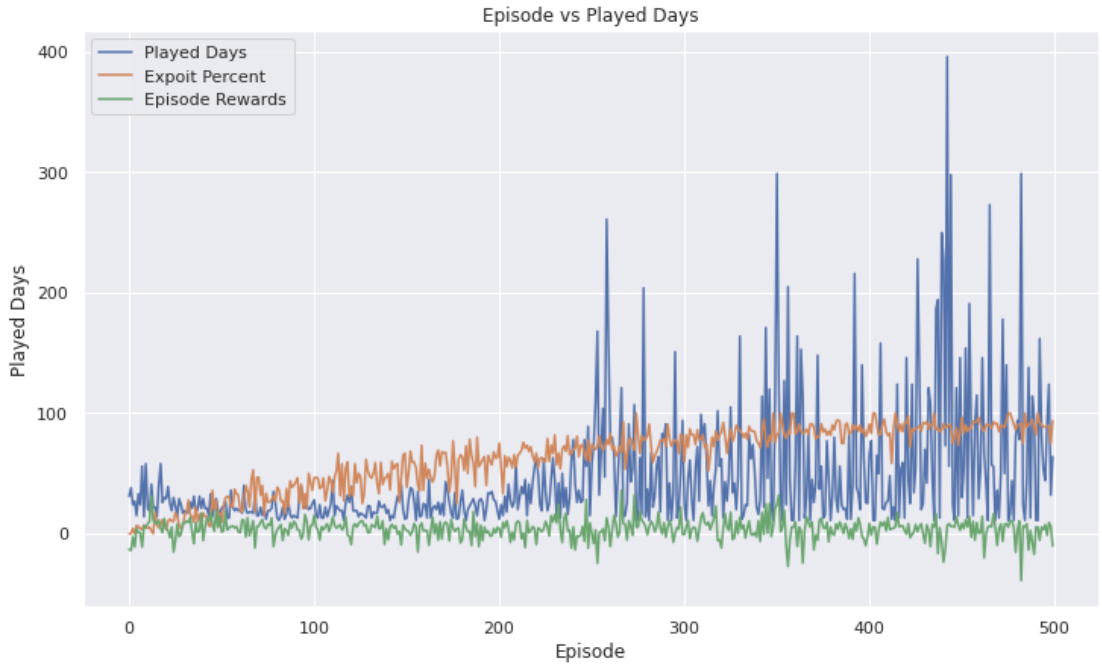


Figure 7. Training result of deep Q-network

The above plot shows the training result of our deep reinforcement learning agent. The blue line indicates the number of days the agent played for each episode. Intuitively, the more days the agent plays, the better the agent performs. The agent is performing better as episode number increases. However it suffers from inconsistency. Stability is a common issue in training deep Q-networks. Even with an additional target model in hope to stabilize the system, the volatility in the stock market still made training difficult.

Evolution Strategy

The major performance evaluation for the ES model is ROI(See following table). Meanwhile, another way to gauge the learning performance of the model across different stocks is to compare the rewards over generation evolution on different stocks. We demonstrate this measurement on the figure below. We see a common trend that the reward across the stocks first experiences an increase until reaching a plateau as the generation evolves from 0 to 500. Furthermore,

combining the following figure and table, we also highlight that the higher the reward reaches at the end, the higher the ROI is.

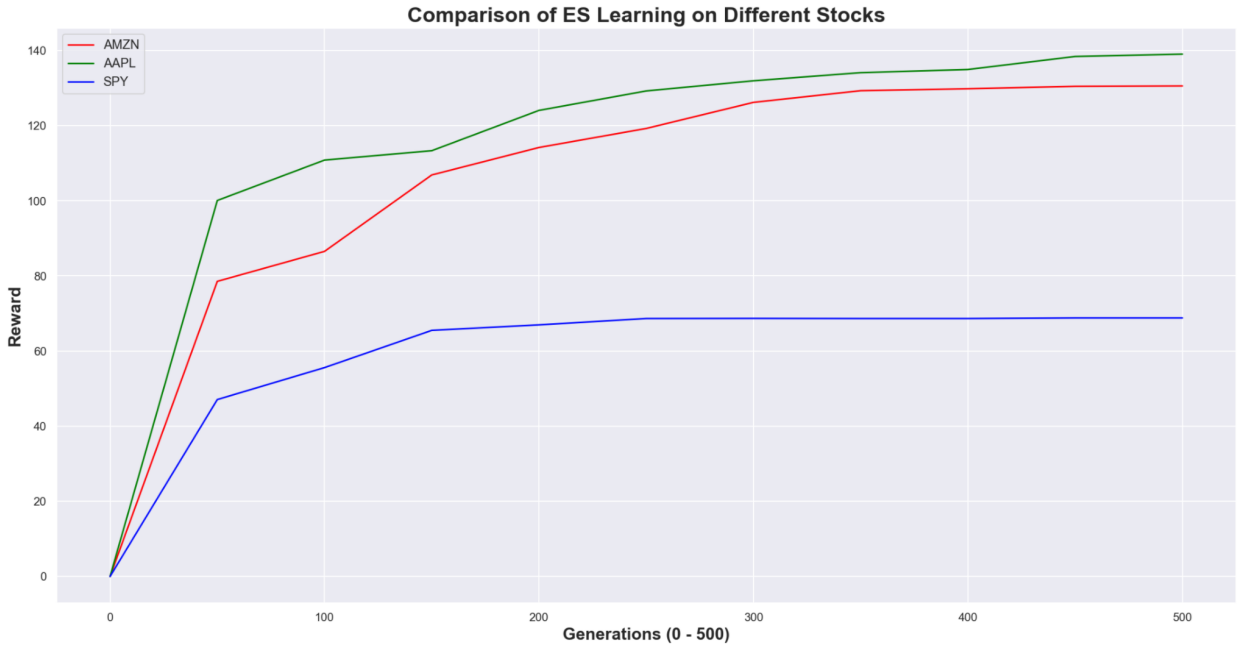


Figure 8. Reward Comparison of ES Learning Performance

The below table summarizes the ROI of each of the ML agents on SPY, AAPL and AMZN stock during the testing period:

Table 6. ROI Performance Comparison across Machine Learning Methods

	SPY	AAPL	AMZN	Average
ARIMA	+7.8 %	+38.0 %	+16.1 %	+20.6 %
LSTM	+16.6 %	+32.7 %	+35.0 %	+28.1 %
Evolution Strategy	+42.9 %	+104.9 %	+74.6 %	+74.1 %
Deep Q Learning	+7.0 %	+13.2 %	+9.6 %	+9.3 %
Baseline	+24.1 %	+18.2 %	+4.6 %	+15.6 %

Test dataset: 7/8/2020 - 16/4/2021

Conclusions

In conclusion, the stock market is very volatile and the best algorithmic trading strategy depends heavily on the stock characteristics and time period. Keeping this in mind, ADX was the best performing technical indicator, on average, across our stocks. ADX had an average return of 28.7%, which beat the baseline average ROI of 15.6%. Our machine learning models had higher average performances overall, but the evolution strategy was our best performing machine learning model with an average performance of 74.1%.

As we look forward to areas of future work - we consider model stacking and including more inputs to be most important. A truly successful trading strategy would make use of the strengths of multiple algorithms. Model stacking would capitalize on each model's strengths to create better algorithms. We also only considered opening prices as our training inputs here, and believe including other inputs such as volume or indicators would be beneficial.

References

Introduction

Folger, J. (2020, August 28). Pros and cons of automated trading systems. Retrieved April 25, 2021, from https://www.investopedia.com/articles/trading/11/automated-trading-systems.asp

SMA

Raudys, A., & Rabarskatie, Z. (2018). Optimizing the smoothness and accuracy of moving average for stock price data. *Technological and Economic Development of Economy, 24*(3), 984-1003. doi:10.3846/20294913.2016.1216906

MACD

Wang, J., Kim, J.. (2018). Predicting Stock Price Trend Using MACD Optimized by Historical Volatility. *Hindawi, Mathematical Problems in Engineering,* Vol.2018, Article ID 9280590, 12 pages, 1-12. doi.org/10.1155/2018/9280590

RSI

Bhargavi, R., et al. "Relative Strength Index for Developing Effective Trading Strategies in Constructing Optimal Portfolio." *School of Computer Science and Software Engineering, VIT University*, International Journal of Applied Engineering Research, 2017, pp. 8926–8936.

Moroșan, Adrian. (2011). The relative strength index revisited. African journal of business management. 5. 5855.

ADX

Gurrib, I. (2018). Performance of the Average Directional Index as a market timing tool for the most actively traded USD based currency pairs. *Banks and Bank Systems*, *13*(3), 58–70. https://doi.org/10.21511/bbs.13(3).2018.06

ARIMA

Amry, Zul, and Budi Halomoan Siregar. "ARIMA Model Selection for Composite Stock Price Index in Indonesia Stock Exchange." *International Journal of Accounting and Finance Studies*, vol. 2, no. 1, 2019, doi:10.22158/ijafs.v2n1p31.

Mondal, Prapanna, et al. "Study of Effectiveness of Time Series Modeling (Arima) in Forecasting Stock Prices." *International Journal of Computer Science, Engineering and Applications*, vol. 4, no. 2, 2014, pp. 13–29., doi:10.5121/ijcsea.2014.4202.

LSTM

Sherstinsky, A. (2021, January 31). *Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network*. arXiv.org. https://arxiv.org/abs/1808.03314.

Chen, K., Zhou, Y., & Dai, F. (n.d.). *A LSTM-based method for stock returns prediction: A case study of China stock market*. IEEE Xplore. https://ieeexplore.ieee.org/document/7364089.

Cheng, L.-C., Huang, Y.-H., & Wu, M.-E. (n.d.). *Applied attention-based LSTM neural networks in stock prediction*. IEEE Xplore. https://ieeexplore.ieee.org/document/8622541.

Roondiwala, M., Patel, H., & Varma, S. (n.d.). *(PDF) Predicting Stock Prices Using LSTM*. ResearchGate. https://www.researchgate.net/publication/327967988_Predicting_Stock_Prices_Using_LSTM#:~:text=Roondiwala%20et%20al.,daily%20percentage%20changes.%20...

Deep Reinforcement Learning

Cartea, Á, Jaimungal, S., & Sánchez-Betancourt, L. (2021). Deep reinforcement learning for algorithmic trading. *SSRN Electronic Journal*. doi:10.2139/ssrn.3812473

Gao, Xiang (2018). Deep reinforcement learning for time series: playing idealized trading games.

Evolution Strategy

Rechenberg, I., Eigen, M..(1973). Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart: Frommann-Holzboog.

Schwefel, H..(1977). Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, volume 26. Basel/Stuttgart: Birkhaeuser.

Korczak, J., Roger, P., Lipiński, P..(2001). "Evolution Strategy in Portfolio Optimization." *Artificial Evolution, 5th International Conference.* Doi: 10.1007/3-540-46033-0_13.

Myszkowski, P., and Bicz, A.. (2010). "Evolutionary algorithm in Forex trade strategy generation." *Computer Science and Information Technology,* pp.81-88, doi:10.1109/IMCSIT.2010.5679921.

Sorensen, E. , Ozzello, R. , Rogan, R. , Baker, E. , Parks, N. and Hu, W. (2020) Meta-Learning of Evolutionary Strategy for Stock Trading. *Journal of Data Analysis and Information Processing*, 8, 86-98. doi: 10.4236/jdaip.2020.82005.

Appendix

ARIMA Methods

Autoregressive Integrated Moving Average (ARIMA) is another common technical analysis tool that uses time-series data to predict future trends/values of a given stock. More specifically, it assesses the strength of a dependent variable such as Close price, to predict future market moves by calculating the difference between values in the series instead of using the raw Close prices. ARIMA models comprise 3 components:

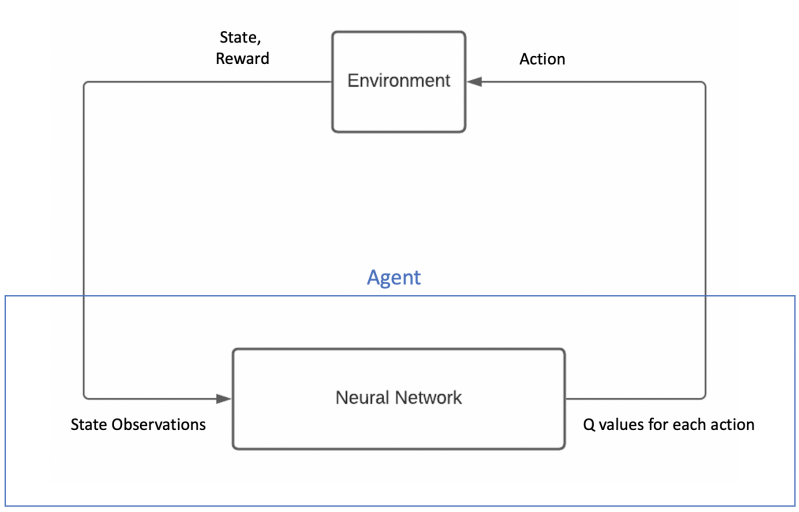
1. Auto-Regressive (AR): Showcases a model where the dependent variable, such as stock Close price, regressed on its previous lagged values. We calculate 'p' which refers to the number of lags to be used as predictors.
2. Integrated (I): Determines the number of differencing iterations on the dependent variable to ensure stationarity is met. We calculate 'd' which refers to the number of differences required to make the time series stationary.
3. Moving Average (MA): Incorporates any residual error between the predicted and observed values and applies it to the lagged values in the model. We calculate 'q' which refers to the number of lagged forecast errors to be incorporated into the ARIMA model.

Deep Q Learning

Stock trading naturally fits into the reinforcement learning framework. The trading process can be described by a partially observable Markov Decision Process. Recent advances of reinforcement learning such as AlphaGo have shown people the power of this framework in tackling difficult problems. We believe it would be interesting to see how it performs on stock

trading. In particular, we picked Deep Q learning, a form of deep reinforcement learning, to address the infinite state space issue. In typical Q-learning, the agent builds and updates a Q-table. However, the stock market has too many states, which makes Q-learning agents impossible to estimate the entire Q table. Deep reinforcement model replaces the Q-table with a deep neural network. The neural network takes in state observations and directly estimates the Q value. This is a powerful method to solve complex problems like stock trading. The below flowchart shows the simplified framework of reinforcement learning applied to stock trading: More details about the design of environment and agent are explained in later paragraphs.

Environment design is a crucial part of reinforcement learning that directly impacts how well an agent can learn. We allowed three actions for an agent to take: buy a single unit of stock, sell a single unit of stock, and hold its position. Everytime the agent makes an action, the environment first checks if the agent has enough money or stock to perform that action. The agent is then forced to pick the best action from the allowed action space. Once action has been confirmed, the environment steps one day forward and returns the reward and the new state's observation, which is the past 60 days of opening price.



Flow diagram of Deep reinforcement learning

To avoid sparse reward assignment, we designed our reward to be the percentage gain/loss from current day's opening price to next day's price:

If action is to buy:

$$R(S,A) = 100 \times \frac{Price_{t+1} - Price_t}{Price_t}$$

If action is to sell:

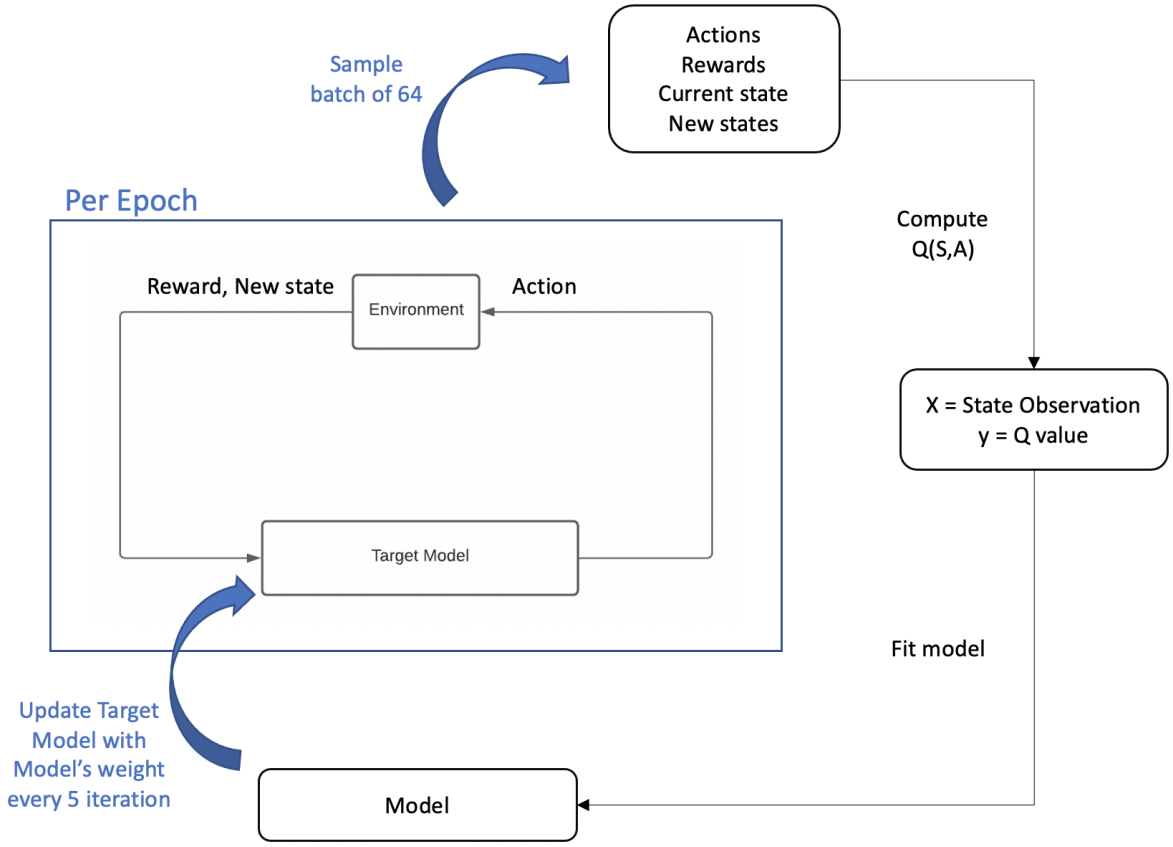
$$R(S,A) = -(100 \times \frac{Price_{t+1} - Price_t}{Price_t})$$

If action is to hold:

$$R(S,A) = 0$$

Reward assignment for each action

This reward setting rewards the agent for buying or selling stocks if tomorrow's price rises or drops respectively. Oppositely, it penalizes the agent for making bad decisions. The hold action provides the agent an alternative option if the agent could not buy (if not enough cash) or could not sell (no holding stocks).



Training procedure of Q-learning agent

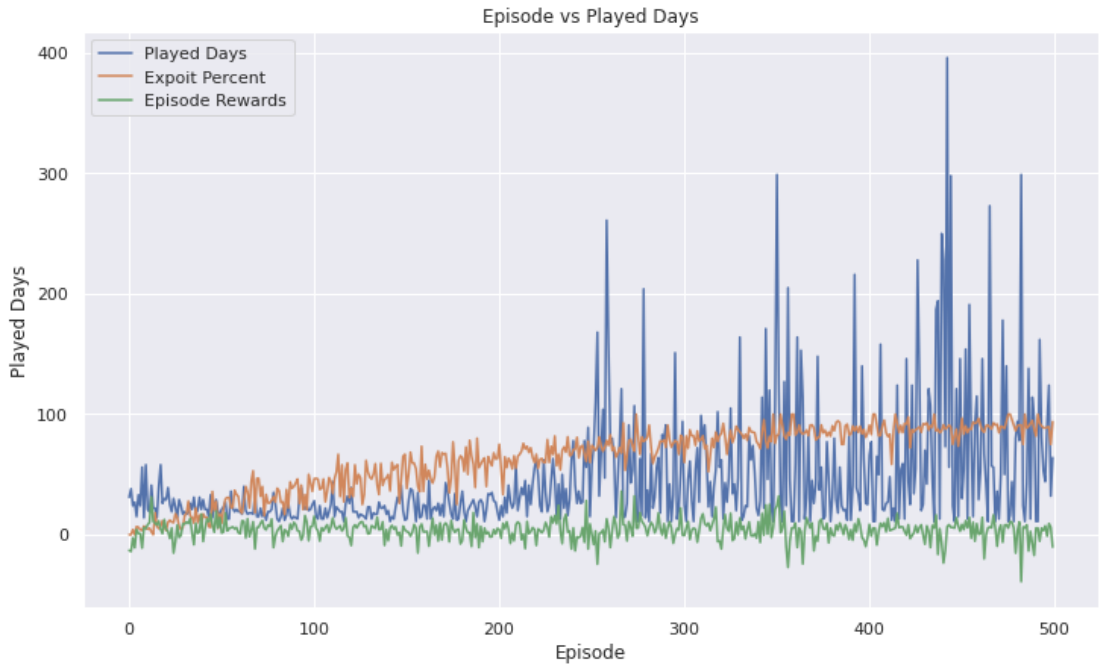
The training process consists of two models: a model and a target model consists of the exact same architecture. We used the exact same model architecture as the LSTM agent, but the output layer was replaced with three nodes corresponding to the Q-values of each action. We used 2

models because training deep reinforcement learning agents is inherently unstable. An additional model called target model is used to stabilize the system. Simply speaking, the target model is responsible for predicting Q-values and making actions at every iteration, while the actual model is getting fitted at every iteration. Every 5 iteration, the target model's weight is updated with the model's weight.

The iteration process starts off by the agent selecting random actions. We defined the parameter epsilon as the exploration constant, where epsilon ≤ 1 and ≥ 0. We also defined an epsilon decay constant of 0.995, so as the training process proceeds, the agent will reduce its exploration and exploit the environment more. The training continues for 500 episodes. For each episode, the agent is given $1000 to invest in the stock market. The episode is forced to end if the agent's investment becomes lower than 95% of its initial investment ($1000) or there is no more stock data to play on (step to the last day of data).

The training process begins by the agent playing one single episode. The process records the current state observations, the action taken by the agent, its reward, and the next state observations into a deque of size 3000 and samples a batch of 64 records. Each record is then used to compute the real Q-value of that state action pair. The state observations become the input for training and the actual Q-value becomes the y label. After processing for 64 X and y samples, they are then fed into the model for fitting. The process continues for 500 episodes.

The below plot shows the training result of our deep reinforcement learning agent. The blue line indicates the number of days the agent played for each episode. Intuitively, the more days the agent plays, the better the agent performs. We can see the agent is performing better as episode number increases. However it suffers from inconsistency. This is a common issue in deep reinforcement learning. In many cases, the target model could solve the stability problem, but because we are using reinforcement learning to trade stocks, the additional volatility in the stock market made training even more difficult.



Due to the slow training process, we only trained our model for 500 episodes, which is far from enough. Unfortunately, due to time limits, we did not find a solution to the stability problem, so we had to stick with this procedure. After the agent is trained, we tested its performance on the 3 stock's test period.

Evolution Strategy

The ES is an alternative to solve the problems where we cannot use the backpropagation to calculate the gradients directly. For example, in reinforcement learning (RL) problems, we can also train a neural network to make decisions to perform a sequence of actions to accomplish some task in an environment. However, it is hard to evaluate the gradient of reward signals given to the agent in the future to an action performed by the agent right now, especially if the reward is realized many time-steps in the future. Even if we are able to calculate accurate gradients, there is also the issue of being stuck in a local optimum, which exists for many RL tasks. On the contrary, ES have competitive properties relative to RL in solving the above: indifference to the distribution of rewards (sparse or dense), no need for backpropagating gradients, and tolerance of potentially arbitrarily long time horizons.