An Application of Deep Imitation Learning to Algorithmic Trading

Christopher Chifor
University of Toronto
christopher.chifor@mail.utoronto.ca

Payam Fakoorziba
University of Toronto
payam.fakoorziba@mail.utoronto.ca

Parssa Kyanzadeh
University of Toronto
parssa.kyanzadeh@mail.utoronto.ca

Abstract

This research paper presents a novel approach to solve the problem of calculating an optimal trading strategy. Recently, there have been efforts in the space of financial AI research, namely machine learning techniques to assist in the quantitative trading field. Prior methods devised had issues with noisy, high-frequency financial data as well as determining a suitable balance between exploration and exploitation of the agent. To address these issues, we present iRTDQN which leverages the challenges of volatility as well balancing exploration and exploitation.

1 Introduction

Because of large amounts of noise and volatile nature, stock market predictions have been a challenging problem in time series prediction in recent times. The process of *accurately* predicting prices in various markets has seen much dispute with the increase of Deep Learning (DL) publications. Quantitative trading as field has become more efficient, allowing for less chance of human judgment mishaps as a decision algorithm permits for more concrete trading strategies. Leveraging newly developed applications of machine learning, the financial technologies industry is continually evolving, but we are aware that no algorithm is *perfect*. Each technique attempts to apply new theoretical ideas so solve a problem presented by a previous trading decision algorithm. In this paper we showcase the novel iRTDQN which attempts to solve the problem of volatility as discussed in *An Application of Deep Reinforcement Learning to Algorithmic Trading* [1].

2 Related Works

2.1 Technical Analysis (Indicators)

Technical analysis is one of the most commonly used tools in the financial markets, where technical indicators are used to obtain trading signals. Technical analysis takes into account historical prices and trading activity to predict future market movements. To achieve this, technical indicators, which are mathematical formulas applied to historical prices, are used model different attributes of the market, such as: market trends, volatility and momentum [2]. Some examples of technical indicators applied this analysis are: Moving average convergence divergence (MACD), Commodity Channel Index (CCI), Stochastic Momentum Index (SMI), Exponential Moving Average (EMA) and Average True Range (ATR).

Preprint. Under review.

2.1.1 Details of Technical Indicators

Table 1: Technical Indicators

Name	Description
MACD	Displays trend following characteristics and momentum characteristics.
CCI	Helps find the start and the end of a trend.
SMI	Shows where the close price is relative to the midpoint of the same range.
EMA20	20 day Exponential Moving Average
ATR	Measures the volatility of price.

2.2 Trading Deep Q-Network (TDQN)

From T. Théate and D. Ernst's *An Application of Deep Reinforcement Learning to Algorithmic Trading* [1], they introduced the TDQN which was inspried from the DQN algorithm presented by Mnih et. al [3]. This is a deep reinforcement learning algorithm capable of learning control policies from high-dimensional sensory inputs. See Figure 1 for the architecture of the model. Effectively it is a reinforcement learning problem where the agent is a Deep Q-Network. In our analysis we will alter this architecture by introducing imitation learning by introducing the discriminator. This is done by implementing the expert actions and a discriminator into the RL environment. The algorithm is trained based on the generation of artificial trajectories from historical daily stock data.

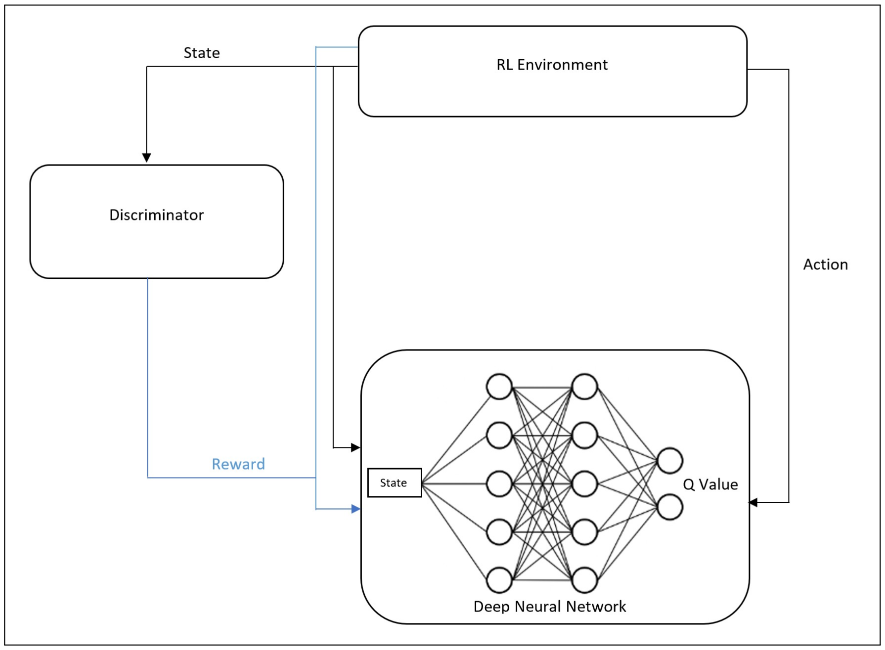


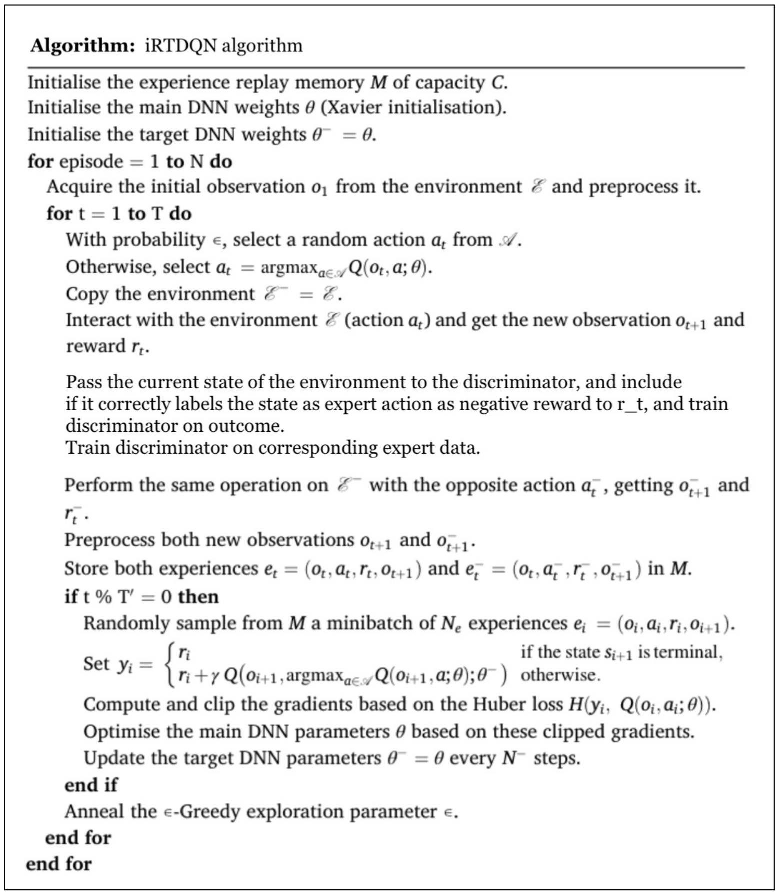
Figure 1: Trading DQN

2.3 Trading Strategy

The trading strategy is a programmed policy which outputs a trading action (buy/sell) given information from the trading agent at a given time step. Notably, it is sequential in nature, meaning that at each trading position, the model makes a decision. First, the model gets updated based on the market information at a time step, then it executes the policy and finally applies the action for that given time step. The data being fed into the model include: open, close, high, low, volume, the five indicators as well as the expert decision. Based on this information the model makes an inferred action (buy/sell). It is important to note, the model can make a decision that is *do nothing*, which is not buy or sell, which is the most common action taken.

3 Algorithm Design

In this section, the modifications of the bass-line algorithm are discussed. Firstly, a deep neural network was used for the RL agent. The only modification for this, is instead of using a CNN, this was replaced by a standard feed-forward deep neural network. Second, the double deep Q-Network was used to assist in the overestimation problem introduced by the DQN. The ADAM optimizer was also implemented to increase the training stability as well as the convergence time of the deep reinforcement learning component. The Huber loss was implements since marginal improvements were noticed in the training phase. Gradient clipping was used to assist with the gradient explosion issues of training a deep neural net. Also, Xavier initialisation was used to improve the convergence time. The expert actions are fed into the reinforcement learning environment to assist in the action process. Finally, general regularisation and normalisation techniques were used to reduce over fitting. The data comes from T. Théate and D. Ernst's *An Application of Deep Reinforcement Learning to Algorithmic Trading* [1].



Algorithm: iRTDQN algorithm

```
Initialise the experience replay memory $M$ of capacity $C$.
Initialise the main DNN weights $\theta$ (Xavier initialisation).
Initialise the target DNN weights $\theta^{-} = \theta$.
for episode = 1 to N do
  Acquire the initial observation $o_1$ from the environment $\mathscr{E}$ and preprocess it.
  for t = 1 to T do
    With probability $\epsilon$, select a random action $a_t$ from $\mathscr{A}$.
    Otherwise, select $a_t = \text{argmax}_{a \in \mathscr{A}} Q(o_t, a; \theta)$.
    Copy the environment $\mathscr{E}^{-} = \mathscr{E}$.
    Interact with the environment $\mathscr{E}$ (action $a_t$) and get the new observation $o_{t+1}$ and
    reward $r_t$.
    Pass the current state of the environment to the discriminator, and include
    if it correctly labels the state as expert action as negative reward to r_t, and train
    discriminator on outcome.
    Train discriminator on corresponding expert data.
    Perform the same operation on $\mathscr{E}^{-}$ with the opposite action $a_t^{-}$, getting $o_{t+1}^{-}$ and
    $r_t^{-}$.
    Preprocess both new observations $o_{t+1}$ and $o_{t+1}^{-}$.
    Store both experiences $e_t = (o_t, a_t, r_t, o_{t+1})$ and $e_t^{-} = (o_t, a_t^{-}, r_t^{-}, o_{t+1}^{-})$ in $M$.
    if t % T' = 0 then
      Randomly sample from $M$ a minibatch of $N_e$ experiences $e_i = (o_i, a_i, r_i, o_{i+1})$.
      Set $y_i = \begin{cases} r_i & \text{if the state } s_{i+1} \text{ is terminal,} \\ r_i + \gamma\, Q(o_{i+1}, \text{argmax}_{a \in \mathscr{A}} Q(o_{i+1}, a; \theta); \theta^{-}) & \text{otherwise.} \end{cases}$
      Compute and clip the gradients based on the Huber loss $H(y_i,\ Q(o_i, a_i; \theta))$.
      Optimise the main DNN parameters $\theta$ based on these clipped gradients.
      Update the target DNN parameters $\theta^{-} = \theta$ every $N^{-}$ steps.
    end if
    Anneal the $\epsilon$-Greedy exploration parameter $\epsilon$.
  end for
end for
```

Figure 2: iRTDQN Algorithm

4 Performance

In this section, performance of the iRTDQN is discussed in relation to other classical strategies, namely: buy hold, short hold, trend following, mean reversion. Where buy hold is when one buys stock and holds for long period of time. Similarly, short hold is where one does the same action but shorts the stock. Trend following is where one follows market trends and buys/sells according to market trends. Finally, mean reversion is where asset price volatility and historical returns will converge to the mean of the price in the long-run. Below is the results of the aforementioned:

Table 2: Performance on Apple Stock

Performance Indicator	Buy&Hold	Short&Hold	Trend Following	Mean Reversion	iRTDQN
Sharpe ratio	1.239	-1.593	1.178	-0.609	2.01
Profit & loss	79823	-80023	68738	-34630	286772.53
Annualised return	28.86	-100.00	25.97	-19.09	48.89
Annualised volatility	26.62	44.39	24.86	28.33	17.64
Profitability ratio	100	0	42.31	56.67	100
Profit & loss ratio	∞	0	3.182	0.492	∞
Sortino ratio	1.558	-2.203	1.802	-0.812	3.09
Max drawdown %	38.51	82.48	14.89	51.12	8.86
Max drawdown (days)	62	250	20	204	24

Comparing to the iRTDQN to the other classical strategies, we can observe that the results are favourable. This will be elaborated upon in the results section.

5 Results

The iRTDQN will be analysed on it's performance on daily apple stock data between the dates: 01/01/2012 - 31/12/2019. Table 2 showcases the performance of the discussed trading strategies. Over 50 epochs the iRTQDN was able to achieve favourable results to the classical strategies in each performance indicator as well as the TQDN proposed by T. Théate and D. Ernst's *An Application of Deep Reinforcement Learning to Algorithmic Trading* [1]. Note that all hyper parameters are the same as in T. Théate and D. Ernst.

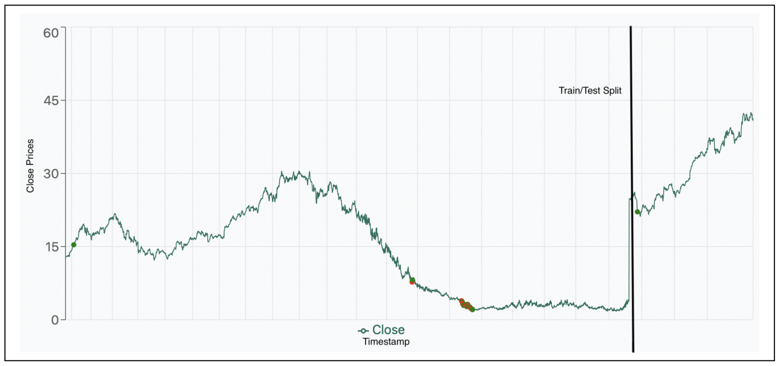


Figure 3: iRTDQN execution for Apple

Firstly, the green dots represent a buy action and red represents a sell action by the model. In the training phase there is a clump of buy and sell actions, however there are still some more spread out actions earlier in the training phase. In the testing phase, the model decided to buy at a low point in the data and hold the stock indefinitely.

6 Conclusion

In this paper, we proposed the iRTDQN architecture to solve the algorithmic trading problem of finding profitable signals in the financial markets. The proposed model combines the TDQN model with imitation learning to improve the model's balance of exploration and exploitation, and also its performance in more volatile markets. The improvement in performance was tested against common trading strategies, and also various performance indicators. The iRTDQN also outperformed T. Théate and D. Ernst's *An Application of Deep Reinforcement Learning to Algorithmic Trading* [1], which shows the potential for this algorithm to be used in the live market.

Link to github project

7 References

[1] Théate, Thibaut, and Damien Ernst. "An application of deep reinforcement learning to algorithmic trading." Expert Systems with Applications 173 (2021): 114632.

[2] Murphy, John J. Technical analysis of the financial markets: A comprehensive guide to trading methods and applications. Penguin, 1999.

[3] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. A. (2013). Playing Atari with deep reinforcement learning. CoRR, abs/1312.5602.