Stock Trading Bot Using Deep Reinforcement Learning

Akhil Raj Azhikodan, Anvitha G. K. Bhat and Mamatha V. Jadhav

Abstract This paper proposes automating swing trading using deep reinforcement learning. The deep deterministic policy gradient-based neural network model trains to choose an action to sell, buy, or hold the stocks to maximize the gain in asset value. The paper also acknowledges the need for a system that predicts the trend in stock value to work along with the reinforcement learning algorithm. We implement a sentiment analysis model using a recurrent convolutional neural network to predict the stock trend from the financial news. The objective of this paper is not to build a better trading bot, but to prove that reinforcement learning is capable of learning the tricks of stock trading.

Keywords Deep learning · Deep reinforcement learning · Deep deterministic policy gradient · Recurrent neural network · Sentiment analysis · Convolutional neural network · Stock markets · Artificial intelligence · Natural language processing

1 Introduction

Trading stocks is a financial instrument developed over years to distribute the risk of a venture and to utilize the stagnant wealth. Distributing the securities, get the company capital for growth which in turn create more jobs, efficient manufacturing, and cheaper goods. Trading of securities makes the economy more flexible while delivering benefits both for the issuer and the holder. Stock trading has gained popularity

A. R. Azhikodan (✉) · A. G. K. Bhat · M. V. Jadhav
Department of Computer Science and Engineering, Ramaiah Institute of Technology, Bangalore 560054, India
e-mail: akhil95raj@gmail.com

A. G. K. Bhat
e-mail: anvitha16@gmail.com

M. V. Jadhav
e-mail: mamsdalvi@msrit.edu

H. S. Saini et al. (eds.), *Innovations in Computer Science and Engineering*, Lecture Notes in Networks and Systems 32,
https://doi.org/10.1007/978-981-10-8201-6_5

as a way of investment, but the complicated environment of trading and the costs of expert traders are hurdles for the common public. The development of adaptive systems that take advantage of the markets while reducing the risk can bring in more stagnant wealth into the market.

We will discuss the concepts we use in the background section. That is followed by the explanation of the design in the architecture section. In the end, we discuss the observations of the trained systems and draw conclusions.

2 Background

The implementation leverages two algorithmic techniques for stock trading. Deep deterministic policy gradient for reinforcement learning and recurrent convolutional neural network for classification of news sentiment.

2.1 Deep Deterministic Policy Gradient

Swing trading is modeled as a Markov decision process (MDP). The states of the environment are denoted as $s_t \epsilon S$, the actions as $a_t \epsilon A$, and rewards as $r_t \epsilon R$ at each time $t \epsilon \{0, 1, 2, 3 \ldots\}$. A standard form of policy gradient technique as defined by Stutton and Barton [1] is followed in this paper. We also define a reward value function denoted by $R(s, a)$. A policy is a set of probabilities of state transitions,

$$P^a_{ss'} = P_r\{s_{t+1} = s' | s_t = s, a_t = a\}$$

The expected rewards are

$$R^a_s = E\{r_{t+1} | s_t = s, a_t = a\}$$

The decisions made by the agent is characterized by the policy π where it can be represented as

$$\pi(s, a, \theta) = P\{a_t = a | s_t = s, \theta\}$$

where θ is the parameter vector, $\forall s \epsilon S$ and $\forall a \epsilon A$. $\pi(s, a, \theta)$ is also represented as $\pi(s, a)$ for ease. The reward is calculated by a reward function of our environment. The reward represents the goodness of each action, but we use discounted reward. The eventual reward would be

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_t + k$$

where γ is called discount factor and has a value between 0 and 1. We standardize these rewards and plug them into back-propagation for each episode. The RL-agent, which is a neural network is trained over multiple episodes for optimization.

Deep deterministic policy gradient (DDPG) is a policy gradient algorithm that uses a stochastic behavior policy for good exploration but estimates a deterministic target policy. DDPG falls under the category of actor-critic reinforcement learning algorithms. The algorithm has two neural networks, actor and critic. The input of the actor is the observation of the environment, and the output is an action. The critic outputs the Q value of the action predicted by the actor and the state of the environment. The actor network is updated using the DDPG algorithm and the critic network is updated using the temporal difference error signal [2].

2.2 Recurrent Convolutional Neural Network

A pure recurrent neural network (RNN) classifier was not chosen for sentiment analysis because it would fail at identifying discriminating phrases occurring in different orders.

The convolutional layer can fairly determine discriminative phrases in a text with a max-pooling layer [3]. Thus, the convolutional neural network (CNN) better captures the semantic of text compared to RNN.

Addressing the issues discussed in *"Recurrent Convolutional Neural Networks for Text Classification"* [3], we choose recurrent convolutional neural network (RCNN) as the network. The RCNN accepts word embeddings which is a result of text pre-processing as the input. The RCNN combination gives benefits of RNN and CNN. Max-pooling of the convolutional layer extracts the best representation of the input. Recurrent nature of the network captures the contextual information to a greater extent while learning word representations.

3 Architecture

We consider the daily stock information, capital, stock assets, and predicted stock trend as the environment the RL-agent interacts with. The agent observes the environment to interact with it using three actions. We restrict the scope of the RL-agent's set of actions to buying, selling, and holding stocks. The fundamental difference between previous architecture [4] and the proposed is the enhancement of stock trend prediction using sentiment analysis of news. The previous RL-based systems are just based on the stock values and the statistics.

The trained RL-agent in Fig. 1 would take the current stock closing price, moving averages, the capital, the number of stocks held, and the prediction of the stock trend as inputs. The stock trend is predicted using a model trained to analyze the sentiment

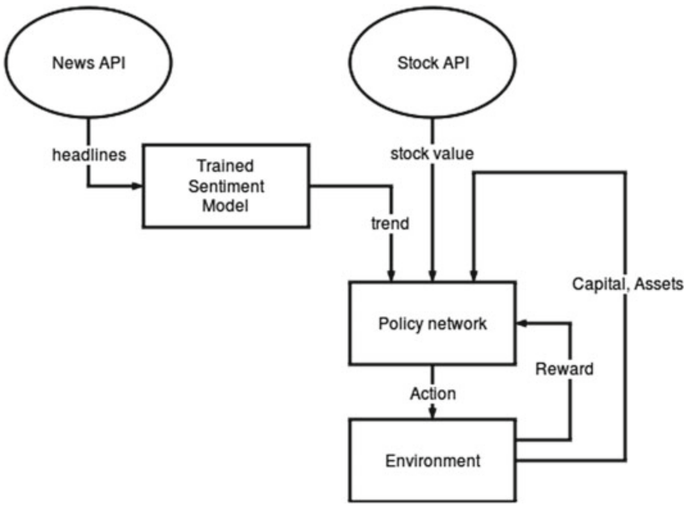

Fig. 1 Overview of stock trading bot

of the news headline. The RL-agent with the given input selects an action. The agent in the paper is restricted to trade a single stock.

3.1 Reinforcement Learning Agent

The reinforcement learning system of the trading bot has two parts, agent and environment. The environment is a class maintaining the status of the investments and capital. It is responsible for accounting stock asset, maintaining capital, providing observation for the RL model, buying stock, selling stock, holding stock, and calculating the reward for action.

The RL-agent uses an object of the environment to interact with it. The agent executes everyday observing the environment to select the action with the policy it learned on training. The agent referred to as the bot from hereafter is responsible for observing the environment, selecting an action with policy, recording rewards, computing the discounted reward, calculating gradient, and updating the policy network with gradients.

3.2 Sentiment Analysis

The financial news along with the change in the stock price is the input for the training sentiment analysis model. The news headlines passed through the sentiment analysis

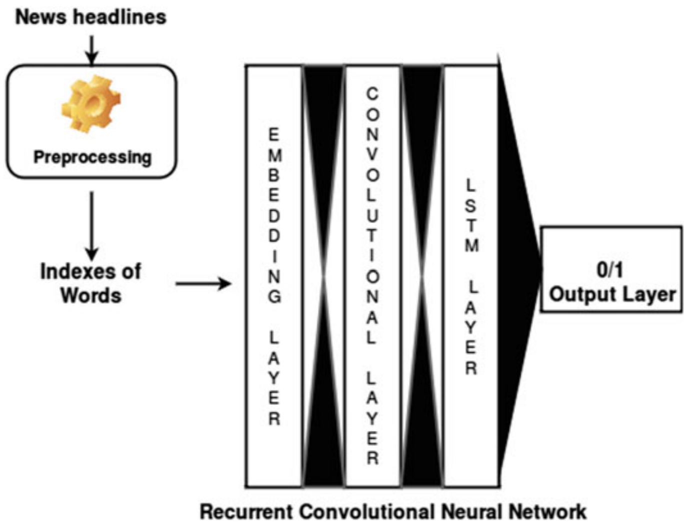

Fig. 2 Recurrent convolutional neural network

model would predict if the stock price will increase or decrease in the next few days. This prediction is fed into the RL-agent as an observation of the environment. The news headlines that are collected are run through a preprocessing which includes—removing HTML markup, tokenizing sentences, removing stop words, stemming, indexing the words from a bag of words. The sentences after cleaning are converted from a list of words to a list of indices [5]. The words are indexed with a bag of words approach. The bag of words is built from a corpus of financial news headlines.

The network has four layers as illustrated in Fig. 2, embedding, convolutional, LSTM (recurrent), and output. The embedding layer converts the positive indices of the words into dense vectors of a fixed size. The embedding layer takes input—a constant size sequence (list of word indices); hence, we pad the shorter sequence to a fixed-sized sequence. The maximum sequence length in the implementation is selected to be a hundred words. The maximum length is selected by analyzing the length of the sequences. The mean length is found to be 56 words in a corpus of 34,000 sequences. The embedding size is 128. The layer is given a dropout rate of 0.25. The significance of dropout in an embedding layer is discussed by Yarin Gal and Zoubin Ghahramani [6]. Dropout is analogous to dropping words at random throughout the input sentence. It can be interpreted as encouraging the model to not depend on single words for its output. The second layer creates a convolutional kernel that is convolved with the input over a single spatial dimension to produce a tensor. The layer is used with one-dimensional max-pooling with a pool length of four. This layer extracts the semantic information from the word embeddings given by the embedding layer. The third layer is a RNN implemented as long short-term memory (LSTM). The LSTM layer introduces a memory element to the network. The layer is efficient in extracting sentence representations enabling our model to analyze long sentences. The final layer is an output layer which predicts the sentiment as a binary number. The positive sentiment is represented using one and the negative by a zero.

4 Results

4.1 Sentiment Analysis

The network Table 1 was trained on 95947 news headlines of 3300 companies and validated on 31581 samples.

The network was modeled with keras [7] running on top of tensorflow [8]. The loss function used was binary cross entropy and the optimizer was Adam. The activation for the other layers was rectified linear units (ReLUs). The training was done with two epochs to avoid overfitting. The test accuracy of multiple tests averaged at 96.88%, while the training accuracy oscillated around 95%. This proves that the stock value change can be predicted to be positive or negative with the news headlines of the company.

Sample result, "ArcBest Corporation: Good Management Battles Poor Economics Seeking Alpha—May 24, 2016 In many ways, the situation that ArcBest Corporation finds itself in today is perfectly captured in Buffett's quotation." returns 0.0024, represents downward trend, whereas " Danaher Completes Acquisition Of Cepheid PR Newswire—Nov 4, 2016 In the merger, each outstanding share of Cepheid common stock was canceled and converted into the right to receive 53.00 per share in cash, without interest." returns 0.99, represents upward trend.

4.2 Reinforcement Learning Agent

The DDPG agent is trained with actor and critic networks modeled in Keras and the training algorithm from keras-rl library [9]. As the training of the RL-agent was done with historical stock data, the news headlines are not available. This problem can be solved by simulating the output of the sentiment analysis with 96% accuracy.

Table 1 RCNN network for sentiment analysis

Layers (type)	Output shape	Parameters
Embedding	(100, 128)	2560128
Convolutional	(96, 64)	41024
LSTM	(70)	37800
Output	(1)	71

The graphs in Figs. 3 and 4 represent the performance of the RL-agent. The "stocks held" graph indicates the number of stocks held on everyday of the experiment. The graphs show that the agent buys and sells continuously. The "comparison of stagnant and RL-bot asset" value graph shows that the agent always maintains a higher value than the stagnant stock value. The system holds the stock for first few days after it

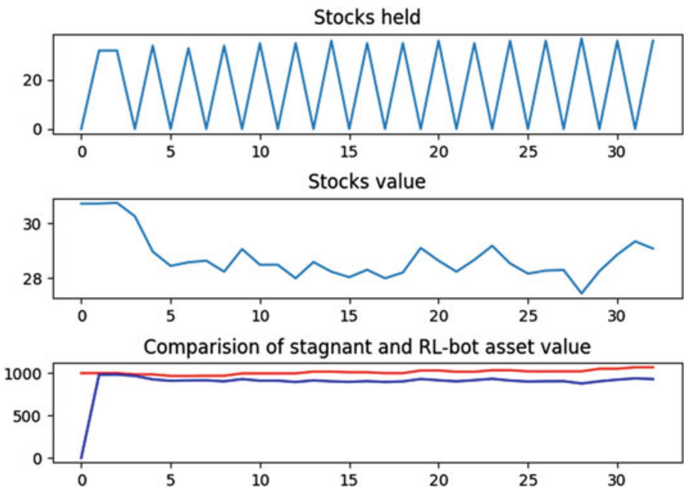

Fig. 3 Training over 30 days with NASDAQ-GE stock. The agent was given $1000 which it tries to maximize. The red line indicates the agent's assets, and the blue line indicates the value of the stagnant stock. The training was done with 50,000 steps which is 1248 episodes of the training data

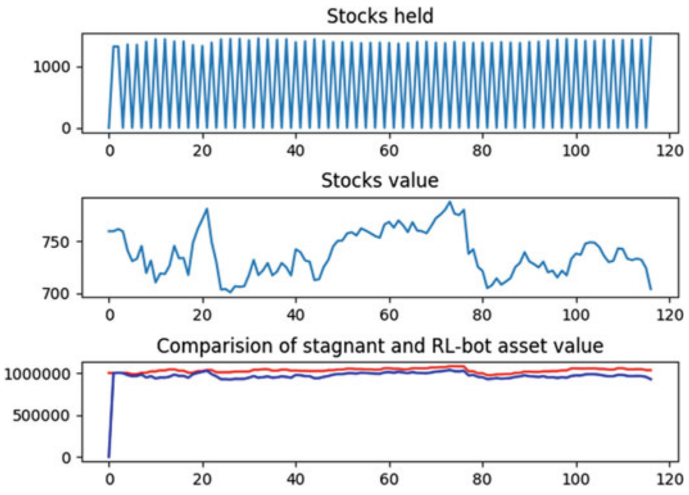

Fig. 4 Training over 5 months with NASDAQ-GOOGL stock. The agent was given $1,000,000 which it tries to maximize. The red line indicates the agent's assets, and the blue line indicates the value of the stagnant stock

makes its initial purchase. This action can be justified by the decrease in the stock prices. The repeated buying action can be seen as an attempt by the system to gain profit. The buying and selling cycles do not always result in profit.

The actor and the critic network (Table 2 and 3) weights are randomly initialized to facilitate exploration. Each episode is also randomly iterated with the first five steps to give the RL-agent a different state to start exploration. To facilitate training of the network, the action predicted by the actor is shuffled 10% of the time.

The three reward systems experimented with were,

1. Difference between RL-agent asset value and stagnant asset value
2. Difference between the cost at which the stocks are sold and the cost at which the stocks were bought
3. A binary reward representing if the action was profitable or not.

The first reward function was tested over multiple stocks, small and big, but it failed to train the neural network. The network gets stuck in the local minima where the agent repeatedly holds the maximum stock. The second reward function also exhibited the same characteristic. The binary reward system performed the best. With a smaller number of episodes, it showed positive results.

Table 2 Actor network for DDPG

Layers (type)	Output shape	Parameters
$\text{Flatten}_1\ (Flatten)$	(5)	0
$\text{Dense}_1\ (Dense)$	(32)	192
$\text{Dense}_2\ (Dense)$	(32)	1056
$\text{Dense}_3\ (Dense)$	(16)	528
$\text{Dense}_4\ (Dense)$	(3)	51

Table 3 Critic network for DDPG

Layers (type)	Output shape	Parameters
Observation_input	(1, 5)	0
Action_input	(3)	0
$\text{Flatten}_2\ (Flatten)$	(5)	0
$\text{Merge}_1\ (Merge)$	(8)	0
$\text{Dense}_5\ (Dense)$	(32)	288
$\text{Dense}_6\ (Dense)$	(32)	1056
$\text{Dense}_7\ (Dense)$	(32)	1056
$\text{Dense}_8\ (Dense)$	(1)	33

5 Conclusion

This work proves the concept that reinforcement learning can be used to trade stocks. Stock trade is not currently best solved with reinforcement learning, but the idea of a computer being able to generate revenue just by trading stocks is encouraging. Such non-deterministic problems can only be solved with neural networks. The system built for this project works with the stocks of one company. This architecture could be scaled to take advantage of the stocks of multiple companies. Scaling this project would need coordination among multiple networks. A master network could be trained to leverage the predictions from individual company networks. The master would consider the actions predicted by the networks and choose among them the best actions it can perform with the resources it has.

References

1. Sutton, R.S., Barto : A.G., Reinforcement Learning: An Introduction in Advances in Neural Information Processing Systems, MIT Press (1998)
2. Patrick Emami (2016) Deep Deterministic Policy Gradients in Tensorow. http://pemami4911.github.io/blog/2016/08/21/ddpg-rl.html. Cited 25 Apr 2017
3. Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao: Recurrent Convolutional Neural Networks for Text Classiffication, Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence
4. M A H dempster and V Leemans: An Automated FX Trading System Using Adaptive Reinforcement Learning, Center of Financial Research Judge Institute of Management University of Cambridge
5. Vasilios Daskalopoulos: Stock Price Prediction from Natural Language Understanding of News Headlines, Rutgers University, Department of Computer Science
6. Yarin Gal and Zoubin Ghahramani: A Theoretically Grounded Application of Dropout in Recurrent Neural Networks, University of Cambridge 2016
7. Franois Chollet: Keras (2017), GitHub repository, https://github.com/fchollet/keras
8. Google. Inc. Tensorow. https://www.tensorflow.org/
9. Matthias Plappert, keras-rl (2016): GitHub repository. https://github.com/matthiasplappert/keras-rl