CS230

Deep Learning Applying on Stock Trading

Bicheng Wang, Xinyi Zhang
{bichengw, xyzh}@stanford.edu

1 Introduction

Profitable trading plays a critical role in investment. Given that the stock market is dynamic and complex, it is challenging to continuously profit on trading. The project proposes to leverage machine learning advantage in data mining, forecasting, automatic trading to explore different approaches to get a profitable portfolio. In our work, to obtain a profitable stock trading portfolio, we design indirectly trading and directly trading approaches–time series forecasting and reinforcement learning– with different Deep Learning models' advantages. Time series forecasting model is used to predict the market price and apply basic trading strategy based on the result, while reinforcement learning model directly learns and outputs with trading action to build portfolio.

2 Related Work

The original idea to use LSTM to predict market stock price is inspired by [1]. [2] summarizes the design experience of using LSTM Recurrent Neural Network to predict stock market. [3] provides an architecture reference to build time series forecasting model. Some previous work ([4-9]) has already discussed the potential approach to apply reinforcement learning in equity market in recent years, but several challenges still exist: 1) Real-world trading data is limited. 2) The reward of reinforcement learning for trading strategy can be defined in multiple ways. Trade-off needs to be fully considered among robust learning rule, the final optimal target, and dataset limitation. 3) Data sparsity [10]. Since training is only based on historical data, some potential pattern may not be captured. It can then cause data sparsity issue because of insufficient historical data.

3 Dataset and Features

We choose 20 stocks with top market capitalization in the S&P500 index from 2000 to 2020 as our dataset. The choosing reason is that the historical data resource is limited and the chosen stocks can account for 90% of the entire market capitalization of S&P 500 Index. Companies which IPO after 2000 are excluded to ensure data completeness within the entire timeframe. The original data is fetched from Yahoo Finance API. Each dataset row is comprised of date, open price, high price, low price, close price, volumn, ticker symbol, # of day in a week. The dataset has 105680 rows in total. We split the dataset into training and test on a 90/10 basis. The training set contains data ranging from 2000 to 2018, while the test set contains data ranging from 2019 to 2020. The sampled original dataset is presented in Table 1.

Feature Engineering is a crucial step for training a high quality machine learning model. We need to check for missing data and do feature engineering in order to convert the data into a model-ready state. Feature engineering is described as follows:

Add technical indicators. In practical trading, various information needs to be taken into account, for example the historical stock prices, current holding shares, technical indicators, etc. In this project, we pick several trend-following technical indicators: MACD, RSI, BOLL, CCI, SMA, DX and EMA.

CS230: Deep Learning, Spring 2021, Stanford University, CA.

date	open	high	low	close	volumne	tic
2000-01-03	0.936384	1.004464	0.907924	0.859423	535796800.0	AAPL
2000-01-03	16.812500	16.875000	16.062500	16.274673	7384400.0	ADBE
2000-01-03	81.500000	89.562500	79.046875	89.375000	16117600.0	AMZN
2000-01-03	25.125000	25.125000	24.000000	13.952057	13705800.0	BAC
2000-01-03	36.500000	36.580002	34.820000	35.299999	875000.0	BRK-B

Table 1: Original Data Sample

3.1 Technical Indicators

The sampled dataset after preprocessing is presented in Table 2.

tic	day	macd	boll_ub	...	cci_10	dx_30	close_120_sma	close_120_ema
AAPL	0	0	0.9256	...	-66.67	100	0.859	0.859
ADBE	0	0	0.9256	...	-66.67	100	16.274	16.274
AMZN	0	0	0.9256	...	-66.67	100	89.375	89.375
BAC	0	0	0.9256	...	-66.67	100	13.952	13.952
BRK-B	0	0	0.9256	...	-66.67	100	35.299	35.299

Table 2: Dataset Features Sample

3.2 Logarithmic Scaling

In practice, if take a look of SP500 in last 100 years, an interesting phenomenon is although growth with recession, the whole stock market growth with exponential rate. And it would introduce no-linear feature in some derivative values. As some features are better to fit in Logarithmic regression, like the basic price, SMA, EMA. We apply log minimum maximum scaling to some specific features to make sure Deep Learning model easier to capture the feature pattern with shadow depth.

4 Modeling

We investigate different approaches to optimize stock trading strategies. Firstly we choose the deep learning architecture, time series forecasting combined with single stock trading strategy, to evaluate stock trading performance. Next we explore reinforcement learning models to optimize the trading performance.

4.1 LSTM Time Series Forecasting model

The LSTM time series forecasting model target is first predict the market then applying with a simple strategy to build portfolio in the market. In this section, we designed a time series stock forecasting and trading model from start to the end.

4.1.1 Loss Function Definition

To training and evaluate model, metrics and loss function would highly influence the final result optimization direction. In our stock prediction model, the loss function choose is different from general regression projects.

In general, we expect the prediction to be as close to the real value as possible, but telling which data point is closer to real value remains a question. For example, imagining AAPL in year 2000 when each share was worth about $1, if you buy with $1, $0.1 increase means you get 10% return. However, in year 2021, the equal event is $200 AAPL share increasing to $220. If we use mean square error to evaluate the y difference, we would underestimate the low price influence, and introduce unbalanced fit. A better choice is to try different loss definition.

We tried Mean Squared Error, Mean Absolute Percentage Error, Mean Squared Logarithmic Error, Huber Loss, Log Cosh loss.

Mean Absolute Percentage Error is a better approach to describe the price and predict price relationship in a long time period as stock price grows or goes down. In our test, the prediction result with MAPE loss model has less skew in prediction time period as stock price goes high.

$$Loss = \frac{100}{n}\sum_{i=1}^{n}|\frac{\hat{price}_i - price_i}{\hat{price}_i}| \tag{1}$$

The gradient derivative of the Loss:

$$\partial Loss = \begin{cases} -\frac{1}{N_{price_i}} & \text{if } \hat{price}_i < price_i \\ undefined & \text{if } \hat{price}_i = price_i \\ \frac{1}{N_{price_i}} & \text{if } \hat{price}_i > price_i \end{cases} \tag{2}$$

To increase the derivative feature for MAPE, we could also define our own loss function with squared MAPE.

4.1.2 Architectures and Hyper-parameters

We explored different architectures including Bi-LSTM, multiple layer LSTM, LSTM with multiple layer Dense with different parameters, and found two layers LSTM with one Dense layer would more easy to get stable predicted model.

Units and dropout selection, based on our current dataset scale, too many parameters would introduce overfit issue, but if introduced regularization methods or dropout to avoid, it would waste calculation resources, and still cannot get better training metrics.

Dataset usage, as we have different stock data, there are multiple combination to use, we investigated training in single stock, and predicting single stock, training in multiple stock and predicting in single stock, and training in multiple stock and retraining in single stock and predicting single stock. The final result shows training in multiple stock is already good enough to predict, but we could still retrain model in specific stock before prediction.

Here are some explored model with metrics comparison table:

Model	Loss	MAE	MAPE	MSE	MAE_{val}	$MAPE_{val}$	MSE_{val}
$baseline_{model}$	1.052367	0.004578	1.052367	0.000038	0.004099	0.972073	0.000032
$Dropout_{model}$	2.371855	0.011330	2.371855	0.000271	0.007442	1.535101	0.000095
MSE_{model}	0.000336	0.006153	1.412530	0.000336	0.004355	1.036787	0.000038
$Huber_{model}$	0.000098	0.005407	1.253396	0.000196	0.004077	0.974119	0.000035
Log $Cosh_{model}$	0.000066	0.005286	1.226172	0.000133	0.003798	0.898953	0.000029
$MAPE_{model}$	0.935923	0.004015	0.935923	0.000031	0.003789	0.888290	0.000029
Single Tic_{model}	2.464035	0.010135	2.464035	0.000172	0.010011	2.582896	0.000145
Less $Unit_{model}$	0.995828	0.004298	0.995828	0.000035	0.003565	0.856291	0.000027
More $Unit_{model}$	1.074033	0.004663	1.074033	0.000039	0.005058	1.143134	0.000044

Table 3: Model Training Metrics Comparison

4.1.3 Trading Strategy

As LSTM model already give a good time series forecasting, we could easily apply a basic trading strategy–Mutant Buy and Hold. In condition that predict price lower than current price, hold the cash, in condition that predict price higher or equal to current price, leverage buy shares and hold. We tried without leverage and with leverage trading, and the results shows in Table 4.

4.2 Deep Reinforcement Learning

The Reinforcement Learning architecture target is to directly generate portfolio trading action end to end according to the market environment.

4.2.1 Model Definition

1) Action: The action space describes the allowed actions that the agent interacts with the environment. Normally, action **a** can have three values:

$$a \in \{\text{-1}, 0, 1\}$$

where -1, 0, 1 represent selling, holding, and buying one stock. Additionally, an action can be carried upon multiple shares. We use an action space {-k, ..., -1, 0, 1, ..., k}, where k denotes the number of shares. For example, "Buy 10 shares of AAPL" or "Sell 10 shares of AAPL" are 10 or -10, respectively.

2) Reward function: r(s, a, s′) is the incentive mechanism for an agent to learn a better action. The change of the portfolio value when action **a** is taken at state **s** and arriving at new state **s**′, i.e., r(s, a, s') = v′ - v, where **v**′ and **v** represents the portfolio value at state **s**′ and **s**, respectively.

3) Environment State: The state space describes the observations that the agent receives from the environment. Just as a human trader needs to analyze various information before executing a trade, so our trading agent observes many different features to better learn in an interactive environment. The state space is represented as ***[b, p, s, macd, boll_ub, boll_lb, rsi_10, rsi_20, cci_10, cci_20, dx_30, close_20_sma, close_60_sma, close_120_sma, close_20_ema, close_60_ema, close_120_ema]***, where b = available balance, p = stock close price, s = shares owned of each stock, followed by technical indicators.

4.2.2 Learning Methods

We apply five kinds of reinforcement learning methods:

1) Proximal Policy Optimization (PPO). The algorithm determines the maximum step size and find the local maximum of the policy within the region like policy gradient method to maximize the gradient. Compared to the TRPO, it directly introduces the KL divergence item as a policy learning penalty to ensure policy learning progress. As a off-policy strategy, it uses importance sampling in the historical trading data, and provide more advantage on historical data shortage.

2) Actor-Critic. In PPO, it would directly optimize the policy. However, it is also important to leverage the value methods — evaluated the expected return, as historical trading data already provide them well — to improve the reinforcement learning. Advantage Actor-Critic (A2C) method would be the potential approach we would use.

3) Deep Deterministic Policy Gradient (DDPG). Deep Deterministic Policy Gradient (DDPG) is a good and non-avoidable model to learn in continuously environment. The algorithm which concurrently learns a Q-function and a policy, uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy. We based on the baseline 3 from OpenAI team to build our DDPG model and evaluate the performance in the stock market.

4) Twin Delayed DDPG (TD3). In our practice, DDPG can achieve great performance sometimes, but it is frequently brittle with respect to hyperparameters and other tuning, and cause stable issue. The improved model–Twin Delayed DDPG (TD3)–trying to addresses these issues. TD3 model learns two Q-functions instead of one, and uses the smaller of the two Q-values to form the targets in the Bellman error loss functions, has “Delayed” policy updates optimization, and added noise to the target action to smooth the Q-function errors.

5) Soft Actor Critic (SAC). Soft Actor Critic (SAC) is another algorithm that optimizes the stochastic policy in an off-policy way, forming a bridge between stochastic policy optimization and DDPG-style approaches. One advantage of SAC is entropy regularization. The policy is trained to maximize a trade-off between expected return and entropy, and it also try to explore increased entropy fields to prevent the policy from prematurely converging to a bad local optimum.

4.2.3 Rolling Train

We adopt rolling model training compare with non rolling model training. In rolling model training, we splits the test data into different time slices, on each slices, fully used all historical data to retrain model and predict on current time slice.

5 Results Analysis

In table 4, model performance is evaluated by annual return, cumulative return, sharpe ratio, max drawdown, Alpha and Beta. The benchmark index is S&P500.

Model	Return_{annual}	$\text{Return}_{cumulative}$	Sharpe Ratio	Drawdown_{max}	Alpha	Beta
LSTM	16.985%	36.854%	1.19	-9.568%	0.09	0.31
$\text{LSTM}_{leverage}$	34.245%	80.216%	1.19	-19.135%	0.20	0.61
A2C	20.465%	45.226%	0.82	-31.112%	-0.03	1.02
$\text{A2C}_{rolling}$	29.829%	68.731%	0.99	-33.518%	0.03	1.14
PPO	27.885%	63.706%	1.09	-25.325%	0.05	0.91
$\text{PPO}_{rolling}$	32.155%	74.842%	1.30	-22.839%	0.10	0.84
DDPG	55.512%	142.263%	1.38	-33.516%	0.22	1.24
$\text{DDPG}_{rolling}$	29.823%	68.715%	1.09	-33.518%	0.04	1.02
TD3	42.003%	101.928%	1.37	-25.235%	0.18	0.89
$\text{TD3}_{rolling}$	41.889%	101.605%	1.27	-28.681%	0.13	1.11
SAC	38.949%	93.321%	1.33	-24.19%	0.15	0.90
$\text{SAC}_{rolling}$	40.443%	97.507%	1.28	-23.196%	0.19	0.83

Table 4: Model Performance on Test Set

The LSTM portforlio result is at a middle performance in the return and sharpe ratio part. However, combined LSTM time series forecasting with the leveraged trading strategy, the Alpha is dramatically high as well as Beta value also outstanding. Benefit from the LSTM prediction result easy to explain to human, it has high potential to combine with other trading strategies together.

Compared on the RL model with rolling predicted RL model, A2C, PPO TD3 and SAC models has stabled improvement on annual return, except DDPG. In our practice, DDPG sometimes may not easy to tune on a suitable state for all different rolling stage, so that may cause some differentiate between tuned not rolling DDPG with rolling DDPG.

Compared with DDPG and TD3 results, and in our model training practice, TD3 indeed give a highly improvement on model training stabilization, to ensure the result not highly depend on the hyper-parameters or tuning, which is a huge advantage in the hard predictable stock market.

6 Conclusion

This project uses time series forcasting LSTM model and reinforcement learning model to learn a profitable trading mechanism. For LSTM, we explored the performance under different hyperparameters to pick the best model. For reinforcement learning, we tried five kinds of reinforcement learning models, and two training strategies.

7 Contributions

Bicheng Wang responded for the full LSTM and half of RL model, and Xinyi Zhang responded for the half of RL model. Our project is under the guidance of Ayush Kanodia. LSTM and RL comparison, rolling prediction improvement is advised by Ayush Kanodia. The original stock trading model design discussed and inspired by Huizi Mao. The LSTM model hyper-parameters, loss functions, derivative, metrics selection, RL models applying inspired by Stanford CS230 Professor Andrew Ng and Kian Katanforoosh.

References

[1] Stock Market Analysis + Prediction using LSTM. https://www.kaggle.com/faressayah/stock-market-analysis-prediction-using-lstm

[2] Adil Moghar, Mhamed Hamiche. Stock Market Prediction Using LSTM Recurrent Neural Network. (2020) https://www.sciencedirect.com/science/article/pii/S1877050920304865

[3] Time series forecasting. (2020) https://www.tensorflow.org/tutorials/structured_data/time_series

[4] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, Anwar Walid. Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy (2020). SSRN: https://ssrn.com/abstract=3690996 or http://dx.doi.org/10.2139/ssrn.3690996

[5] Thomas G. Fischer. Reinforcement learning in financial markets - a survey - (2018) https://www.econstor.eu/handle/10419/183139

[6] Jingyuan Wang, Yang Zhang, Ke Tang, Junjie Wu, Zhang Xiong. AlphaStock: A Buying-Winners-and-Selling-Losers Investment Strategy using Interpretable Deep Reinforcement Attention Networks. (2019) https://arxiv.org/abs/1908.02646

[7] Yuqin Dai, Chris Wang, Iris Wang, Yilun Xu. Reinforcement Learning for FX trading. (2019) http://stanford.edu/class/msande448/2019/Final_reports/gr2.pdf

[8] Zhuoran Xiong, Xiao-Yang Liu, Shan ZXhong, Hongyang Yang, and Anwar Walid. Practical Deep Reinforcement Learning Approach for Stock Trading. (2018) arXiv preprint arXiv:1811.07522 https://arxiv.org/abs/1811.07522

[9] Thibaut Théate, Damien Ernst. An application of deep reinforcement learning to algorithmic trading. (2021) ISSN:0957-4174 https://doi.org/10.1016/j.eswa.2021.114632.

[10] Mahdi Nasiri, Behrouz Minaei, Zeinab Sharifi. Adjusting data sparsity problem using linear algebra and machine learning algorithm. ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2017.05.042.

[11] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In The 31st International Conference on Machine Learning. (2015) arXiv:1502.05477 https://arxiv.org/abs/1502.05477

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. Proximal Policy Optimization Algorithms. (2017) arXiv preprint arXiv:1707.06347 https://arxiv.org/abs/1707.06347

[13] Vijay Konda , John Tsitsiklis. Actor-critic algorithms, Society for Industrial and Applied Mathematics, vol. 42, 04. (2001) https://papers.nips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[14] Sharpe Ratio. https://www.investopedia.com/terms/s/sharperatio.asp

[15] Cumulative Return. https://www.investopedia.com/terms/c/cumulativereturn.asp

[16] Maximum Drawdown (MDD). https://www.investopedia.com/terms/m/maximum-drawdown-mdd.asp

[17] David Pfau, Oriol Vinyals. Connecting Generative Adversarial Networks and Actor-Critic Methods. (2016) arXiv preprint arXiv:1610.01945 https://arxiv.org/abs/1610.01945