Deep Learning in Asset Pricing

Chang Deng, Haoyu Guan, Yanbin Liu, Yuxuan Liu
May 2, 2020

Abstract

We followed the way in "Deep Learning in Asset Pricing" by Luyang Chen, Markus Pelger and Jason Zhu, which introduce how to corporate deep learning technique with fundamental theorem of asset pricing. In the process, a Recursive Neural Network with Long-Short-Term-Memory units was used to deal with macroeconomic variables and a Forward Feedback Network was used to expression nonlinear weights function and conditional function. Finally, a Generative Adversarial Network was employed to solve asset pricing problem. Here we perform this approach in Chinese Market.

1 Introduction

1.1 Fundamental Theorem of Asset Pricing

In finance, by the First Fundamental Theorem of Asset Pricing, if we have the no-arbitrage condition satisfied, then there exist a stochastic discount factor(SDF), such that:

$$\mathbb{E}_t \left[M_{t+1} R^e_{t+1,i} \right] = 0 \tag{1}$$

where the M_{t+1} is the SDF at time $t+1$, and the $R^e_{t+1,i}$ is the excess return of asset i at time $t+1$.

The stochastic discount factor(SDF) is an affine transformation of the tangency portfolio, we then have the following expression of stochastic discount factor(SDF):

$$M_{t+1} = 1 - \sum_{i=1}^{N} \omega_{t,i} R^e_{t+1,i} = 1 - \omega_t^\top R^e_{t+1} \tag{2}$$

Plug Equation (2) back into Equation (1), given no-arbitrage condition satisfied, we then have:

$$\omega_t = \mathbb{E}_t \left[R^e_{t+1} {R^e_{t+1}}^T \right]^{-1} \mathbb{E}_t \left[R^e_{t+1} \right] \tag{3}$$

which are the portfolio weights of conditional mean-variance efficient portfolio.

In this research, we are trying to generate the function of these weights:

$$\omega_{t,i}(I_t, I_{t,i}) \tag{4}$$

in which I_t represent the information set of macroeconomic variable at time t, while the $I_{t,i}$ represent the specific microeconomic variable of company i at time i

In one way, since these weights form a conditional mean variance efficient portfolio, we can directly trade this portfolio in the market. In th other way, these weights also shows an optimal strategy in every time t, and the function helps us to update easily.

1.2 Generative Adversarial Method of Moment

In order to find the function for the SDF weights, we need to solve a moment of methods problem. The no-arbitrage condition implies that we need to satisfy:

$$\mathbb{E} \left[M_{t+1} R^e_{t+1,i} g\left(I_t, I_{t,i}\right) \right] = 0 \tag{5}$$

for infinitely many function $g : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^d$. The unconditional moment condition can be thought as the pricing error for a choice of portfolios and times determined by $g(.)$. The challenge lies in finding the relevant moment conditions to identify the SDF.

In the paper, they come up with a min-max problem that characterizes all the above observation:

$$\min_{\omega} \max_{g} \frac{1}{N} \sum_{j=1}^{N} \left\| \mathbb{E}\left[\left(1 - \sum_{i=1}^{N} \omega\left(I_t, I_{t,i}\right) R^e_{t+1,i} \right) R^e_{t+1,j} g\left(I_t, I_{t,j}\right) \right] \right\|^2 \tag{6}$$

in this min-max optimization problem, we are actually minimizing the violation of no-arbitrage condition from Fundamental Theorem of Asset Pricing, which is to find a function ω that can perform the best under the "worst" case: a optimal function g that can maximize the violation.

1.3 Deep Learning Algorithms

In order to solve this min-max optimization problem, the paper introduce the following procedure:

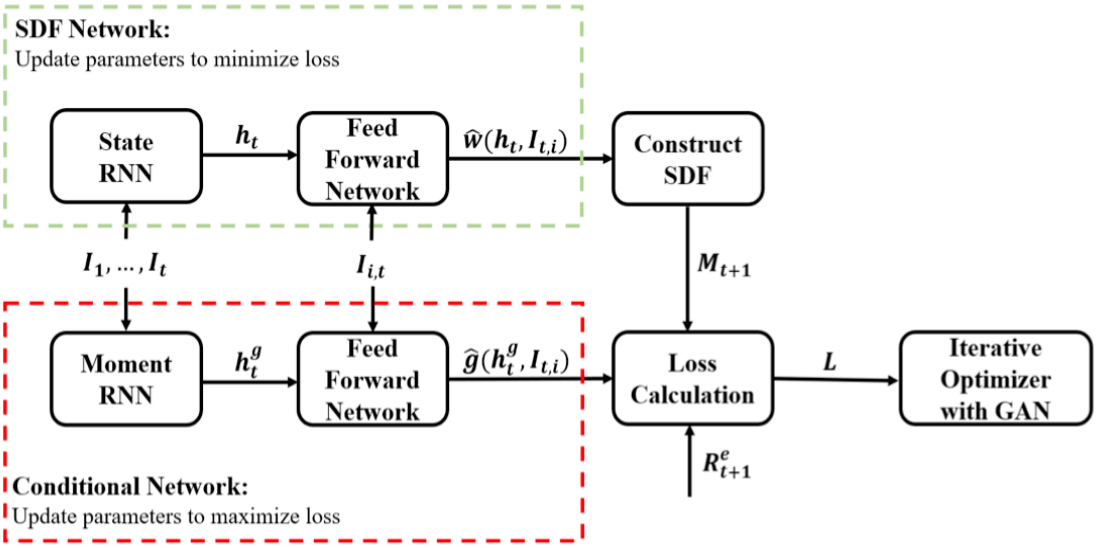

Figure 1: Model Architecture

It contains mainly two part, one of which is the SDF Network, the other of which is the Conditional Network. Finally these two parts are joined to calculate the loss and be optimized by a GAN optimizer.

Then we take a look at each component in this architecture.

1.3.1 Forward Feedback Network

A forward feedback network is a dense neural network. It contains three components. The input layer takes in the each sample with multiple features at a time and the pass them into the next layer. The hidden layers perform the nonlinear calculate. Each hidden layer takes the output of previous layer as input, then take a linear transform, and impose a nonlinear activation function in the end, then finally pass it to the next layer. The output layer takes in the output of hidden layers, and then generate the required output. The typical image of the FFN is shown in Figure 2.

In the architecture, FFN is used in both the SDF Network and Conditional Network. In SDF Network, FFN is used to express weights function ω. In Conditional Network, FFN is used to show the multiple moment conditions $\hat{g}$.

1.3.2 Recursive Neural Network

When people are reading a sentence, we have to input a word but remember the last word at the same time. However, forward feedback network does not have memory. It just deal with the input word, which is not enough.Recursive

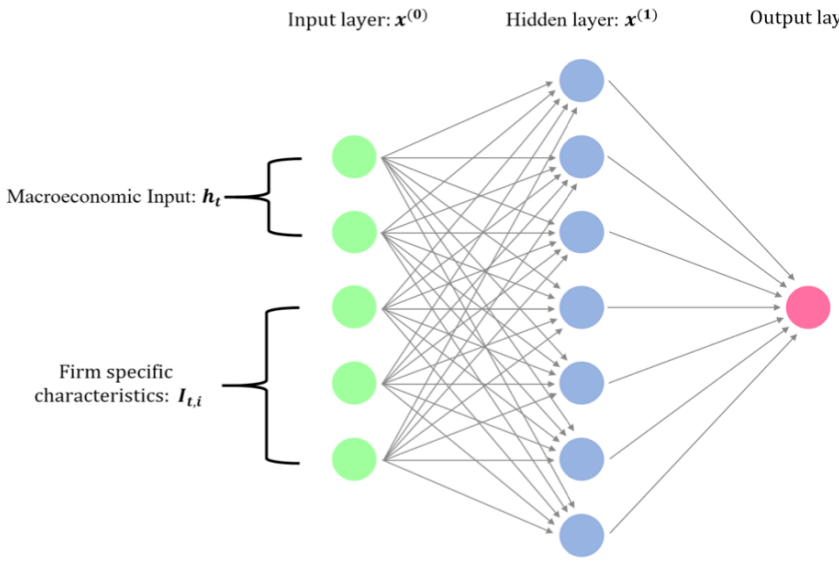

Figure 2: Model Architecture

Neural Network is a special network for compute an element in sequence. The data before would be saved as a state to control the output of the whole network.

Long Short-term Memory(LSTM) a complex network which could include not only the last data but also the data far before.The RNN with LSTM will find the appropriate stationary transformation of the variables such that their dynamics explain asset prices. A conventional Recurrent Neural Network can take into account the time series behavior but can encounter problems with exploding and vanishing gradients when considering longer time lags. This is why we use Long-Short-Term-Memory cells.

There are three Gates in LSTM network to control the output. We want to use LSTM to deal with both the large dimensionality of the system and a very general functional form of the states while allowing for long-term dependencies. Therefore, we use LSTM to learn the macroeconomic variables I_t and output h_t into the Forward Feedback Network.

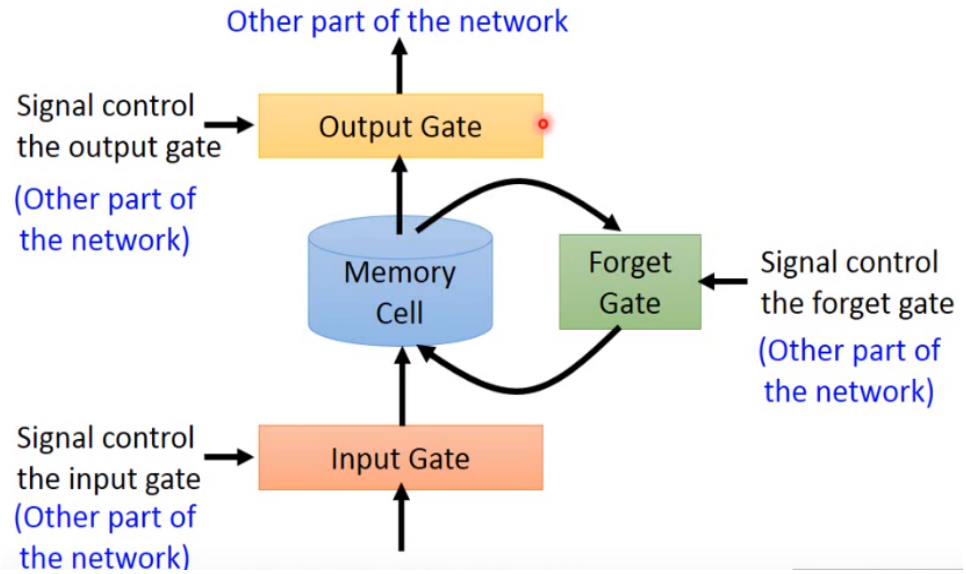

Figure 3: LSTM network

1.3.3 Generative Adversarial Network

Notice here we are actually facing a min-max problem, where the SDF Network is aiming to minimize the loss, while the Conditional Network is purposed at maximizing the loss. We implement the approach in the following ways.

- First, consider unconditional loss function, that is without $\hat{g}$, and solve the minimization problem and get a initial value for the parameters in the SDF network.
- Second, fix the parameters in the SDF network. Solve the maximization problem in the Conditional Network and get the optimal value for the parameters in the Conditional Network.
- Third, fix the parameters in the Conditional network. Solve the minimization problem in the SDF Network and get the optimal value for the parameters in the SDF Network.
- implement the above repeated until the loss function becomes stable.

1.3.4 Loss Function

The loss function in our approach is:

$$L\left(\omega|\hat{g}, I_t, I_{t,i}\right) = \frac{1}{N}\sum_{i=1}^{N}\left\|\frac{1}{T}\sum_{t\in T} M_{t+1}R^e_{t+1,i}\hat{g}\left(I_t, I_{t,i}\right)\right\|^2 \tag{7}$$

This is exactly the same expression as the object function to minimize the violation of no-arbitrage conditions.

2 Empirical Results

2.1 Data Source

From Wind database, we collect data of stocks traded in China A-share market since 01/01/2007 up to now. Relevant data involves daily close price (for calculation of daily return), macroeconomic data publicized quarterly by Chinese official institution and firms specific data from quarter financial statements.

2.1.1 Stocks

We select 50 stocks from the component of SSE Composite Index who has been on the market before 01/01/2017 so we can acquire relatively complete data. The code of these 50 stocks are shown in the following chart.

600053.SS	600056.SS	600136.SS	600162.SS	600167.SS	600180.SS	600185.SS	600197.SS	600230.SS
600299.SS	600612.SS	600641.SS	600673.SS	600675.SS	600681.SS	600682.SS	600742.SS	600755.SS
600779.SS	600782.SS	600801.SS	600826.SS	600828.SS	600835.SS	600850.SS	600585.SS	600690.SS
600837.SS	600887.SS	601088.SS	601111.SS	601166.SS	601186.SS	601318.SS	601328.SS	601390.SS
601601.SS	601628.SS	601668.SS	601688.SS	601766.SS	601857.SS	601888.SS	601939.SS	601988.SS
600236.SS	600763.SS	600703.SS	601398.SS	601989.SS				

2.1.2 Macroeconomic Data

For various macroeconomic data stated by authority, we divide them into eight categories and choose fifteen indicators representing these categories.

Category	Indicator
GDP	Quarter GDP Amount
Industry	CPI, PPI, RPI
Import and Export	Quarter Import and Export Amount
Consumption	Quarter Consumption Amount
Investment	Quarter Investment Amount
Rate and Monetary	M_0, Exchange rate, Loan rate
Fiscal	Fiscal Revenue, Fiscal Expenditure
Employment and Salary	Employment rate, Average Salary Level

2.1.3 Microeconomic Data

For data revealed in firms' financial statements, we also divide them into eight categories and select thirty indicators to characterize firms' specific traits from the eight categories.

Category	Indicator
Valuation	P/S, P/E, Dividend Yield, Market Value, Earning to Market Value Ratio Book-to-Market, BM to Industry average
Risk	Beta
Profitability	ROA, ROE, EPS, Operating profit / Total operating income, Net Profit Margin on Sales/Net profit margin, Net Profit / Total operating income, Maintenance / Total operating income, Extraordinary Item, Assets Devaluation,
Revenue Quality	Operating profit/Total profit, Operating income, Investment Income
Capital Structure	Debt Asset ratio, Equity Multiplier, Equity ratio
Solvency	Operating profit/Liability
Operating Capacity	Total Assets Turnover
Cash Flow	FCFF, CFPS Net cash flow from operating activities Depreciation and Amortization

For all price data, macroeconomic data and microeconomic data above, we resample data into daily frequency, go through Schmidt orthogonalization to erase collinearity, standardization to reduce effect of different dimensions and even lag data to include impact of time series.

2.2 Results

Here we divide the data into three disjoint sets. The first is used for training purpose, the second is used for tuning hyperparameters. And the last is used to serve as test set.

By tuning in the validation set, we get the following optimal hyperparameters as shown in the chart 1.

Notation	Hyperparameters	Candidates	Optimal
HL	Number of layers in SDF Network	2, 3 or 4	2
HU	Number of hidden units in SDF Network	64	64
SMV	Number of hidden states in SDF Network	4 or 8	4
CSMV	Number of hidden states in Conditional Network	16 or 32	32
CHL	Number of layers in Conditional Network	0 or 1	0
CHU	Number of hidden units in Conditional Network	4, 8, 16 or 32	8
LR	Initial learning rate	0.001, 0.0005, 0.0002 or 0.0001	0.001
DR	Dropout	0.95	0.95

Then we use this set of parameter to the test set, and get the following result. Here we use the equally weighted portfolio and mean-variance portfolio as benchmark.

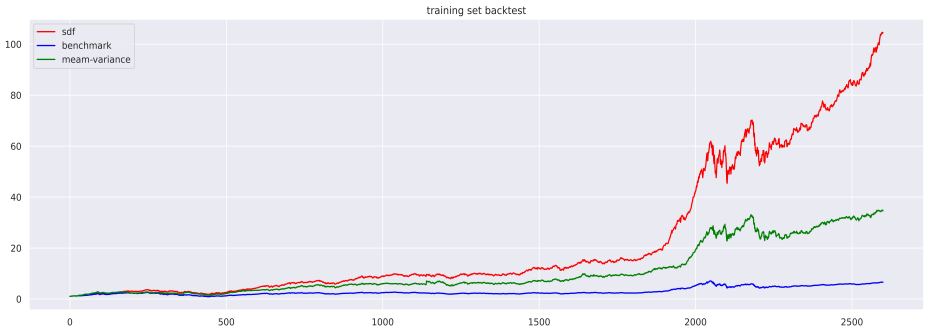

Figure 4: Training set

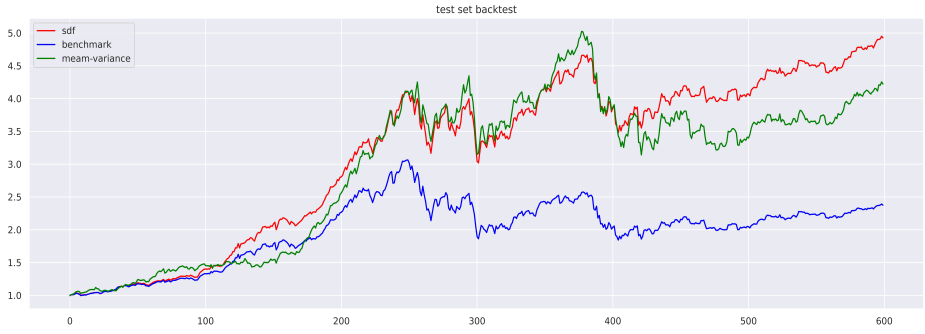

Figure 5: Validation set

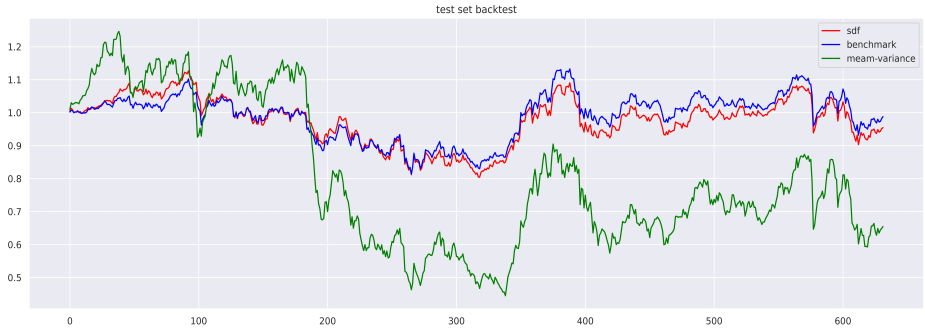

Figure 6: Testing set

We can see that the performance of the back testing in training set works very well while in validation set and testing set looks closer.

	Train	Va1	Testing
Benchmark	0.801714	1.354198	0.080040
MV	1.336177	1.929056	−0.093803
GAN	1.801015	2.458797	0.015558

This are the Sharp Ratio of those three portfolio. We can see our method's performance are all better than other two methods. Therefore, we can conclude a good performance of our model.

3 Conclusion

Our constructed non-linear pricing model performs better than others as we take into account the interrelationship of the vast amount of conditioning characteristics, while considering factors for time-variation. In order to generate this optimal pricing model, we implement three different deep neural network structures together which are Forward Feedback Network, Recursive Neural Network, and Generative Adversarial Network. Our innovation is on the basis of no-arbitrage theory, then to estimate the stochastic discount factor that explains different asset returns. Our SDF is a portfolio of all traded assets on Chinese stock market with time-varying portfolio weights which are general functions of the observable firm-specific , macroeconomic, and microeconomic variables. Our pricing model enables us to identify mis-pricing of sticks, understands the main factors that drive asset prices, and generates the optimal mean-variance portfolio.

Our pricing model is beneficial to investors and portfolio managers. Since our model can outperform the benchmark and be tested on explaining our SDF factor portfolio with respect to the risk exposure. We tested that our model performs better than the liner models as we incorporate the information of all characteristics and macroeconomics factors into a group number of firms. Also we implemented macroeconomic time-series as hidden states that captures the relevant macroeconomic information for asset pricing. Finally, our model generated the risk measure β and the optimal portfolio weight ω as a function of characteristics and macroeconomic variables. The user of our model will be able to determine the risk measure and allocate portfolio weights in a more comprehensive way.

References

[1] Luyang Chen, Markus Pelger, and Jason Zhu. Deep Learning in Asset Pricing. Papers 1904.00745, arXiv.org, March 2019.