Exploration of Risk Factor Neural Networks Based on Daily Stock Data

Tian Tian
Tsinghua University
Computer Science and Finance Dual Degree Program
tiantianunique@gmail.com

Abstract

Two key methods to achieve good investment returns are predicting portfolio risk through the covariance matrix of returns and selecting stocks based on historical data. In terms of risk prediction, traditional multi-factor models often rely on experience and prior knowledge, but they may face adaptability issues when market conditions change. This study develops two neural network-based prediction models: one approach first uses historical stock factor exposures and price-volume data to train the network to predict the future covariance matrix of individual stock returns, and then selects stocks based on maximizing the Sharpe ratio using the covariance matrix; the other approach directly trains the network with the goal of maximizing the portfolio Sharpe ratio, skipping the covariance prediction step and directly selecting stocks. These methods aim to overcome the limitations of traditional models under dynamic market conditions by improving risk assessment and portfolio optimization, thereby enhancing investment decision-making effectiveness.

1 Background

In recent years, research on stock market portfolios has increasingly highlighted the importance of risk management. The core of risk management and asset allocation decisions lies in the accurate prediction of future risks. Good risk predictions enable investors to construct portfolios and adjust strategies flexibly based on estimates of returns and uncertainties. The most common portfolio selection method is based on Markowitz's mean-variance theory, which aims to maximize the difference between the expected return and the risk cost of a portfolio [9]. The correctness of this selection method hinges on the accurate prediction of portfolio risk, which is the core objective of risk models: accurately predicting the future risk of certain optimal portfolios.

In practical applications, the covariance matrix of stocks often has a high dimension, leading to matrix ill-conditioning or rank deficiency, which can cause significant errors and affect the stability and accuracy of the results. One solution is matrix shrinkage, where off-diagonal elements are shrunk towards zero [7]. However, the assumption of matrix sparsity underlying this method may not be applicable in the context of stock risk prediction, as stock returns are generally correlated.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

Another approach is to decompose the high-dimensional covariance matrix of stock returns into a low-dimensional covariance matrix of risk factor returns. This method is based on the assumption that stocks with similar market behaviors are likely to exhibit similar market behavior, implying that high-dimensional stock returns driven by the same factors are influenced by low-dimensional risk factors. This approach is also known as the multi-factor model, whose fundamental idea is that stock returns are driven by a few common factors. The portion of returns not explained by these factors is termed "idiosyncratic returns," and the idiosyncratic returns of each stock are uncorrelated with one another [10].

Despite the practical feasibility of multi-factor models, their estimated results still have some limitations:

Estimation Bias: The estimation based on historical data assumes that financial market data is independently and identically distributed (i.i.d.) over different time steps. However, this assumption does not hold true in real financial markets, as financial data often exhibit distributional shifts, meaning that the distribution of future data can differ from that of historical data. This leads to difficulties in accurately predicting future covariances using historical statistics. In particular, for minimum variance portfolios, stocks that were highly correlated in the past may show decreased correlation in the future, resulting in an underestimation of actual risk and reduced hedging capabilities.

Estimation Variance: Accurate estimation of the covariance matrix requires a sufficiently large number of samples. However, the number of available samples in reality is often limited, which increases the variance of the estimation error. Traditional risk estimation methods often mitigate estimation bias by using a small number of recent samples, but this leads to an increase in estimation variance due to the insufficient number of samples, creating a trade-off between estimation bias and variance.

To address these issues, this study proposes a novel neural network risk model. This model utilizes neural network technology to simulate the drift in sample distribution over time, thereby minimizing estimation bias as much as possible. Simultaneously, by incorporating a factor model-based structure, the model reduces the dependence on a large number of samples, effectively lowering estimation variance and achieving more accurate risk modeling. This approach provides a more reliable and data-driven solution for risk management, aiming to enhance the accuracy and practicality of portfolio risk prediction.

Additionally, this study proposes another model based on neural network optimization of investments, which skips factor prediction. Instead, it directly performs mean-variance optimization based on historical data to maximize the Sharpe ratio, using three models: Gated Recurrent Unit (GRU), Transformer, and Temporal Convolutional Network (TCN). This paper will compare the two prediction approaches to explore whether predicting the factor covariance matrix using neural networks is a necessary step for stock selection.

2 Related Research

The inspiration for this study directly comes from the first reference in the bibliography. This reference, based on the multi-factor model, proposes the idea of obtaining risk factors through neural networks. The multi-factor model posits that stock returns are driven by common factors and can be represented as a multivariate linear model. By regressing stock returns on given factor expo-

sures, one can obtain factor returns and idiosyncratic returns, allowing each stock's return r_i to be described by the following linear relationship:

$$r_i = \sum_{k=1}^{K} X_{ik} f_k + u_i$$

where X_{ik} represents the exposure of stock i to factor k, f_k is the return of factor k, and u_i is the idiosyncratic return of stock i. The return R_p of a portfolio can be expressed as the weighted sum of its constituent stock returns:

$$R_p = \sum_{i=1}^{N} w_i r_i$$

where w_i is the weight of stock i in the portfolio. The portfolio return can be further expressed as a weighted form of individual factor returns:

$$R_p = \sum_{i=1}^{N} w_i r_i = \sum_{i=1}^{N} w_i \left(\sum_{k=1}^{K} X_{ik} f_k + u_i \right) = \sum_{k=1}^{K} \left(\sum_{i=1}^{N} w_i X_{ik} \right) f_k + \sum_{i=1}^{N} w_i u_i$$

Considering the exposure of the entire portfolio to risk factor k:

$$X_p^k = \sum_{i=1}^{N} w_i X_{ik}$$

We can further simplify the expression for portfolio return:

$$R_p = \sum_{k=1}^{K} X_k^P f_k + \sum_{i=1}^{N} w_i u_i$$

Additionally, since the returns of individual factors are uncorrelated with idiosyncratic returns and the idiosyncratic returns of different stocks are also uncorrelated, the expected return and variance of the portfolio can be calculated based on the statistical properties of factor exposures and idiosyncratic risks:

$$\mathrm{var}(R_p) = \sum_{k,l} X_p^k F_{kl} X_p^l + \sum_{i=1}^{N} w_i^2 \,\mathrm{var}(u_i)$$

where F_{kl} is the factor covariance matrix between factors k and l. In summary, the multi-factor model transforms the return analysis of N stocks into the return analysis of K factors, thereby decomposing the stock covariance matrix into the sum of a low-rank matrix and a diagonal matrix. In practice, the number of stocks N is much greater than the number of common factors K, thus the model achieves dimensionality reduction, reducing analysis complexity while improving prediction accuracy.

Furthermore, the multi-factor model adjusts the estimates of factor covariance and idiosyncratic variance in various ways to mitigate estimation bias, achieving accurate risk prediction for optimal portfolios. The reasonable selection of factors is crucial for constructing a multi-factor analysis.

Mainstream factor classifications include value factors, growth factors, profitability factors, size factors, momentum factors, etc. Basic risk factors are initially proposed due to their interpretability and effectiveness, being defined as systematic risk factors in the Capital Asset Pricing Model (CAPM) [11]. Subsequently, several factors such as size, value, and momentum have been proposed [3, 5]. These well-known factors are designed by experts based on extensive historical data and academic research, requiring considerable effort. Nevertheless, these factors still exhibit limitations in predicting market performance.

Given the limitations of manually designed factors, a new method of statistical risk factor generation has been proposed. This often involves applying principal component analysis (PCA) or factor analysis to historical stock returns [1, 6]. Unlike manual factors, this method is more efficient, though its results tend to perform well on the training set but poorly on the test set due to the overfitting of historical returns. As current methods are not sufficiently effective, we plan to design risk factors through a more effective approach.

This paper selects three models for training in mean-variance optimization: GRU, Transformer, and TCN.

The GRU is based on recurrent neural networks and includes reset and update gates. Unlike long short-term memory (LSTM) networks, GRU uses hidden states instead of cell states to transmit information, reducing parameters while increasing efficiency [4]. The reset gate of GRU calculates the parameters $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$ and hidden state $\hat{h}_t = \sigma(W_o \cdot [r_t \circ h_{t-1}, x_t] + b_o)$, which helps capture short-term dependencies in time series data. The update gate calculates the parameters $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$ and hidden state $h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \hat{h}_t$, which helps capture long-term dependencies.

The core of the Transformer model is the attention mechanism, which involves interactions between queries (Q), keys (K), and values (V): $Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V$. Its high computational efficiency allows for parallel processing through matrix operations. This mechanism is particularly suitable for handling long-range dependencies, as it can directly compute dependencies between any two positions in the sequence [12].

TCN is tailored for time series data, aligning well with the research background of this study. TCN handles sequence dependencies through convolutional layers instead of recurrent layers [2]. For an input sequence $x_0, x_1, \ldots, x_T$, we aim to predict the corresponding output: $y_0, y_1, \ldots, y_T = f(x_0, x_1, \ldots, x_T)$, where the causal constraint only uses inputs from the current or earlier time. The goal of sequence modeling is to find a network model f that minimizes the loss between the label outputs and predictions:

$$L(y_0, y_1, \ldots, y_T, f(X_0, x_1, \ldots, x_T))$$

Additionally, TCN has the characteristic of dilated convolutions, allowing the network to capture long-range sequence dependencies without increasing parameter count or computational complexity.

3 Solution and Technical Implementation

3.1 Factor Covariance Matrix Prediction

Considering that methods used to adjust estimates in multi-factor models are manually designed and dependent on prior knowledge, they may not be suitable under changing market conditions. Therefore, we plan to develop a neural network-based risk model that predicts factor covariance

and idiosyncratic risks in a data-driven manner, eliminating the need for expert experience while achieving risk modeling and prediction.

For predicting factor covariance or idiosyncratic variance based on neural networks, the key is to design a data-driven approach that optimizes the predicted factor covariance and idiosyncratic variance in the right direction. A straightforward idea is to calculate the difference between predicted values and actual values and use this as the loss function for optimization. However, this method faces two challenges.

First, estimating the true future covariance matrix requires future sample data. Although the number of elements to estimate for factor covariance and idiosyncratic variance is significantly reduced compared to individual stock covariances, it still faces the problem of sample size selection. Assuming the estimation interval is $[T, T + \delta t]$, if the sample time interval δt is large, the estimation error for factor covariance and idiosyncratic variance is small, but these values are time-variant. Therefore, the estimates only represent the true values for the interval $[T, T + \delta t]$, not the desired prediction for time $T + 1$, leading to a large time dimension bias. If a smaller δt interval is chosen, the estimation variance increases. Thus, this approach is contradictory and infeasible.

Second, assuming we can obtain the true factor covariance and idiosyncratic variance, traditional loss functions, such as mean squared error between predicted and actual matrix elements, may not be suitable for risk models. Therefore, we need to design a loss function that better aligns with risk models (see Appendix A for specific model principles and steps).

3.2 Mean-Variance Optimization

Given the complexity of factor covariance prediction, we aim to directly predict stock selection strategies to compare the effectiveness of the two methods and determine whether predicting the factor covariance matrix is necessary.

In terms of implementation details, we compared the architectures of GRU, Transformer, and TCN models, ultimately selecting GRU for its superior performance in time series prediction. Additionally, the model includes a fully connected layer for the final output, using a loss function that minimizes the negative Sharpe ratio, ensuring consistency and directness of the optimization objective. The model employs the SAM optimizer to enhance generalization during training, reducing the risk of overfitting. During model training, we can set upper and lower limits on weights to meet various constraints in real-world investment scenarios, thus better applying it to risk management and asset allocation.

4 Experiments and Results Analysis

4.1 Factor Covariance Matrix Prediction for A-Shares

In this study, we plan to use daily data from the Chinese A-share market over the past decade (2012-2022) to train our model, which predicts stock returns for the next 21 days based on historical data from the past 63 days. To obtain historical data, we selected qlib as our data source. Considering some limitations of qlib data, such as inconsistencies in adjustment methods, we decided to exclude all stocks containing adjusted data. Additionally, we standardized the time span and cleaned some invalid data, ultimately obtaining valid data for 639 stocks. For missing values in the data, we used the previous day's data to fill in, maintaining data continuity.

In the model implementation, we built a RiskFactorExtraction model comprising multiple GRU layers and a fully connected layer, aiming to extract risk factors from historical data. The input data for

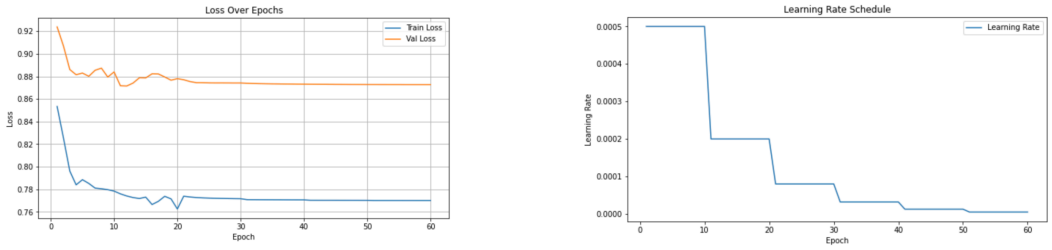
the model consists of two five-dimensional tensors, corresponding to historical stock data and future returns prediction. Specifically, the tensor structure for historical stock data is torch.Size([1035, 2, 63, 639, 9]), interpreted as follows: 1035 batches; each batch contains 2 elements representing a day in the historical data; 63 represents the 63 days of data prior to that day; 639 indicates the number of stocks analyzed; 9 represents nine features for each stock, including high price, low price, dividend, PE ratio, close price, open price, trading volume, adjusted close price, and trading amount. The tensor structure for future stock returns is torch.Size([1035, 2, 1, 639, 2]), with similar dimensional meanings to the historical data, where the third dimension represents the predicted returns for a future day (21 days later in this study), and the fifth dimension indicates that we stored two sets of 21-day-ahead predicted returns to focus on two specific risk factors during model training.

The trained model outputs a three-dimensional tensor predicting the risk factor values for each stock on a future day. Using the factor covariance matrix derived from the paper's mathematical formulas, we transform it mainly through regularization methods to achieve dimensionality reduction and obtain the stock covariance matrix. Then, based on these predicted future covariance matrices and the current returns of all stocks, we apply Markowitz's model in finance to construct a portfolio aimed at maximizing the Sharpe ratio. Additionally, we implemented a dynamic learning rate adjustment strategy, conducting a total of 60 training epochs and splitting the dataset into training and validation sets in an 8:2 ratio. Considering the continuity characteristics of time series data, we chose not to shuffle the data to gain a more intuitive understanding during visualization and analysis. This approach not only maintains the temporal relationship in the data but also helps to more accurately assess the model's ability to capture time dependencies.

The final hyperparameters used are as follows, where gamma represents the weight value for variance and expectation in the loss function:

$$num\_layers = 4, dropout = 0.5, lr = 0.0005, gamma = 0.01$$

The loss values during training are shown in Figure 1, indicating that the model stabilized and converged after approximately 25 training epochs.



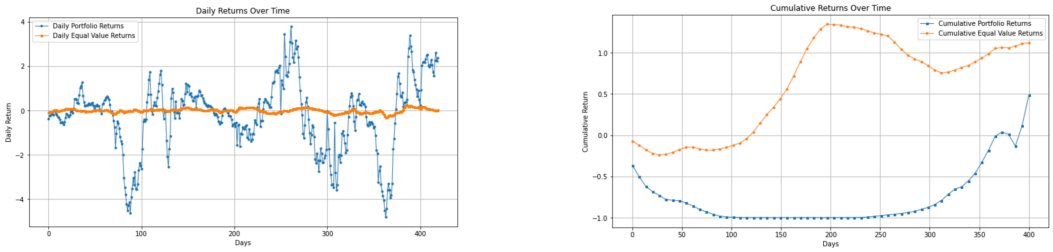
(a) Loss Curve (b) Learning Rate Curve

图 1: Training Results of Factor Covariance Matrix Prediction Model

On the validation set, we used the trained model to predict the covariance matrix and construct the portfolio. The daily returns and cumulative returns of the portfolio are shown in Figure 2.

From the experimental results, it can be observed that our strategy's return performance is not ideal compared to an equal-weight portfolio, accompanied by greater volatility and drawdowns. We believe this outcome is due to two main factors.

First, our approach involves using historical stock data to predict future risk factors, further estimating the future covariance matrix with these factors, and constructing investment strategies based on these covariance matrices. Although this method is theoretically feasible, directly using historical data to adjust portfolio weights may be more straightforward and effective in practice.



(a) Daily Returns (%) (b) Cumulative Returns

图 2: Results of the Factor Covariance Matrix Prediction Model

Second, we used daily data obtained through qlib to extract risk information, whereas the original paper used Barra-style factors for risk information extraction. Daily data tends to be noisier, and the quality of qlib data is lower. The original paper's approach of using neural networks to further extract information based on experience factors might be more effective. Additionally, our risk factor prediction network structure largely borrowed from the original paper's network. Despite adjusting the parameters, the inconsistency in data sources likely led to suboptimal performance.

4.2 Model Switching

Therefore, we decided to make adjustments to address these two issues. In terms of model structure, we plan to introduce a new model that directly uses historical stock data to implement a mean-variance stock selection strategy, selecting the portfolio with the highest predicted Sharpe ratio. We will compare the results of this new model with those of the model implemented according to the paper. Regarding data sources, we decided to switch to more stable US stock data, specifically the top 50 performing stocks in the S&P 500 index, covering daily data from 2001 to 2020, obtained through yfinance. The risk-free rate selected for calculating the Sharpe ratio is 0.02.

Based on these adjustments, we added two new sets of experiments. The first set uses the newly selected US stock data to train the previous factor prediction covariance model, aiming to test the model's performance with different data categories and serve as a control group. The second set, as the experimental group, will use our newly constructed model that directly predicts the Sharpe ratio, trained and tested on US stock data, and compared with the first set of experiments. Due to our lack of experience using RNN models in such predictions, we conducted parameter tuning and training for GRU, TCN, and Transformer models, which are suitable for handling sequence data. After comparing the performance of each model, we selected the best-performing GRU model for the final results.

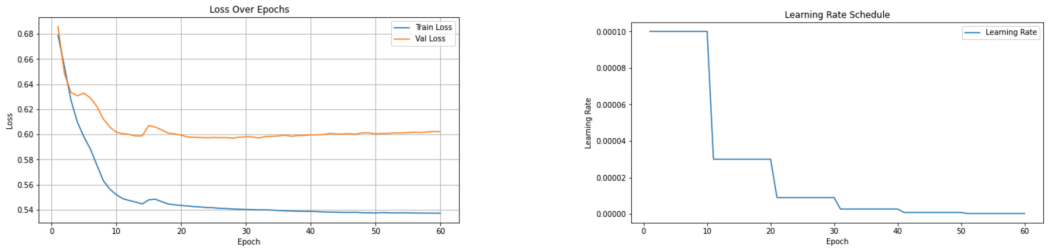
The newly acquired US stock historical data includes six features: open price, close price, high price, low price, adjusted close price, and trading volume.

4.3 Factor Covariance Matrix Prediction for A-Shares

For the original model, we did not change the training framework and data format. The historical data input to the model is formatted as torch.Size([2547, 2, 63, 50, 6]), and the future return data input to the model is formatted as torch.Size([2547, 2, 1, 50, 2]), with the same meanings as the previous formats, which will not be repeated here. Considering the longer time span of the data, we increased the model's complexity, with the final hyperparameters as follows:

$$hidden\_size = 64, num\_layers = 6, dropout = 0.5, lr = 0.0001, gamma = 0.01$$

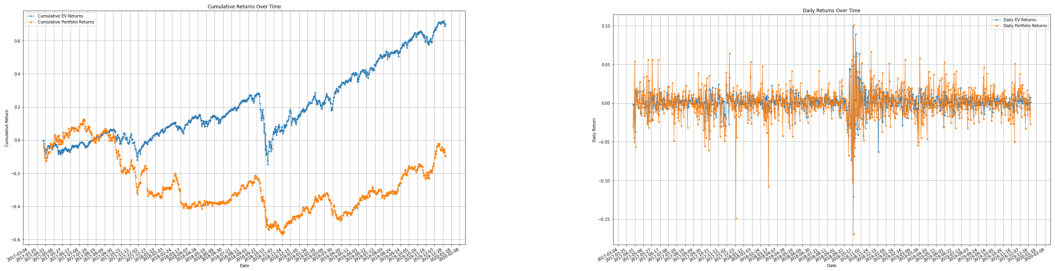
We trained for a total of 60 epochs. As shown in Figure 3, the loss curve indicates that the validation set loss value no longer improved after about 35 epochs.



(a) Loss Curve (b) Learning Rate Curve

图 3: Training Results of Factor Covariance Matrix Prediction Model (US Stocks)

We used data from April 11, 2017, to February 8, 2020, as the validation set. The performance of the strategy constructed using the predicted covariance matrix from the model on the validation set is shown in Figure 4.



(a) Cumulative Return Curve (b) Daily Return Curve

图 4: Factor Covariance Matrix Prediction Model Portfolio Results

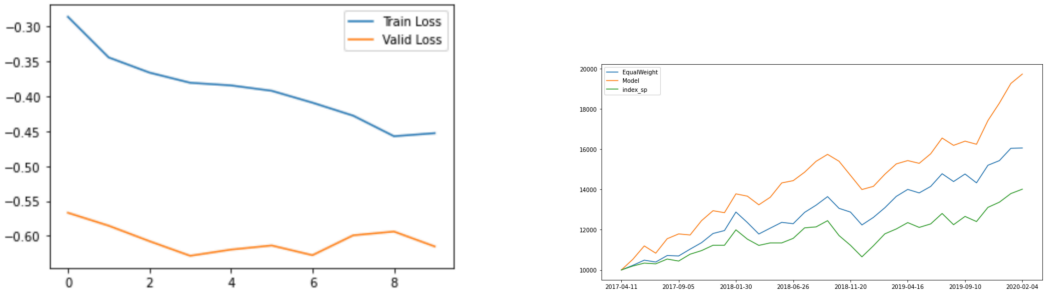
The blue line represents an equally weighted portfolio of these 50 stocks, and the orange line represents the dynamically adjusted portfolio based on our daily predicted covariance matrix. It can be observed that the constructed portfolio's performance is not satisfactory, even failing to exceed the overall index and experiencing larger drawdowns. The daily return curve also shows that our strategy has significantly more volatility compared to the index, indicating higher overall risk.

Even after changing the data, the performance of the factor prediction covariance matrix model remains poor, suggesting that the issue is not with the data but rather that the model's approach is not direct enough.

4.4 GRU-Based Stock Selection Model with Sharpe Ratio Maximization

Therefore, we adopt a strategy of directly predicting the portfolio to maximize the Sharpe ratio. To achieve this, we use the negative Sharpe ratio as the loss value during training, aiming to find the portfolio weights with the highest Sharpe ratio. After consulting with colleagues experienced in deep learning, we chose to use the SAM optimizer for model training.

First, we performed parameter tuning for GRU, TCN, and Transformer models and closely observed their performance to select the best-performing model. The detailed record of this parameter tuning process is included in the appendix. After comparison, we found that the GRU model performed the best in terms of the Sharpe ratio, so we selected it as our final model.



(a) Loss Curve of GRU Stock Selection Model (b) Return Curve of GRU Stock Selection Model

图 5: Results of GRU Stock Selection Model

In the final training, we used a dataloader to batch load the data, ensuring that the time series order was preserved by not shuffling the data, and directly deleted data that did not fit into a complete batch. The final hyperparameters used are as follows:

EPOCHS=10, EARLY_STOP=20, LR=0.025, MOMENTUM=0.95, N_LAYER=3, HIDDEN_DIM=128, DROPOUT=0.3

We trained for a total of 10 epochs. As shown in Figure 5, the loss curve indicates that the validation performance stabilized after about 5 epochs, with a fast convergence rate. We tested using the same mean-variance strategy, employing an industry-standard simulated trading backtesting framework with an initial asset value of 10000 points to measure strategy performance. The validation set spans from April 11, 2017, to February 8, 2020, with portfolio adjustments every 21 days. The strategy performance is shown in Figure 6.

It can be seen that the strategy performs relatively well, outperforming the equally weighted portfolio and the S&P 500 index (as the 50 stocks selected were among the better-performing ones in the S&P 500). Additionally, there were no significant drawdowns. The return curve of the strategy, the equally weighted portfolio, and the S&P 500 index show similar overall trends, indicating that it is difficult to completely avoid market influence regardless of portfolio adjustments. Moreover, we added a rebalancing constraint, stipulating that the change in any single stock' s weight cannot exceed 10% of the total capital pool, making the overall strategy more conservative. As shown in Figure 6, the changes in stock positions within the portfolio during this period are as follows, indicating that the overall changes in proportions were not drastic:

Compared to the control group experiment, we used the same strategy and data but achieved a significant performance improvement by adjusting the prediction method. This indicates that predicting

the covariance matrix using historical data is somewhat indirect and greatly affects the model's predictive ability.

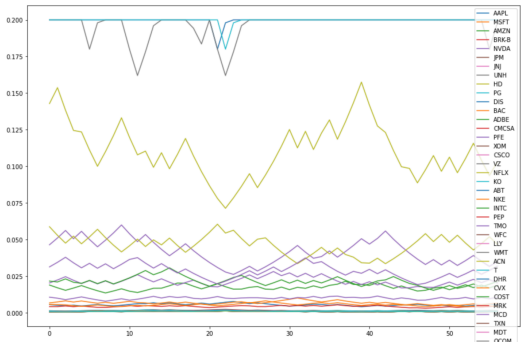


图 6: Changes in Stock Positions for GRU Stock Selection Model

5 Conclusion

Our exploration shows that the strategy of predicting the covariance matrix based on daily stock data is not particularly attractive. In our tests, the GRU neural network strategy, which directly aims to maximize the Sharpe ratio, outperformed the risk factor prediction strategy. It seems that in this problem, reducing the number of intermediate variables and directly focusing on the final variable is more effective.

Of course, it is possible that our implementation of the risk factor strategy was not perfect. We can further optimize from both data processing and model design perspectives.

From the data processing perspective, we aim to obtain and refine higher-quality data from the A-share market. We believe that the poor performance of the model on A-shares is largely due to the low quality of the data used. This was evident during our distillation of the original qlib dataset, where information such as stock splits and adjustments were not properly handled. Additionally, we made several manual selections in the final stock set, retaining only those with specific periods and features, which may not be reasonable. For instance, we can use the discarded split information to train a model on the stock data we did not use and compare the performance of the two models to see if the addition of split features improves the model's effectiveness. Furthermore, we can try changing the time period to observe if our model performs better under specific market conditions. In summary, after obtaining or selecting higher-quality data, we can train the two models on the new dataset to see if the training outcomes change.

From the model design perspective, there is room for improvement in the risk factor model. We can refer to some cutting-edge techniques proposed in academia, such as using AutoEncoder, VAE, etc., to try to enhance the overall quality of factors and improve their prediction accuracy and stability through the introduction of new technologies.

参考文献

[1] Lin, H., Zhou, D., Liu, W., & Bian, J. (2021, November). Deep risk model: a deep learning solution for mining latent risk factors to improve covariance matrix estimation. In *Proceedings of the Second ACM International Conference on AI in Finance* (pp. 1-8).

[2] Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

[3] Carhart, M. M. (1997). On Persistence in Mutual Fund Performance. The Journal of Finance, 52(1), 57–82. https://doi.org/10.2307/2329556

[4] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[5] Fama, E. F., & French, K. R. (1992). The Cross-Section of Expected Stock Returns. *The Journal of Finance*, *47*(2), 427–465. https://doi.org/10.2307/2329112

[6] Harman, H. H. (1976).ă*Modern factor analysis*. University of Chicago press.

[7] Ledoit, O., & Wolf, M. (2017). Nonlinear Shrinkage of the Covariance Matrix for Portfolio Selection: Markowitz Meets Goldilocks. *The Review of Financial Studies*, *30*(12), 4349–4388. https://www.jstor.org/stable/48616725

[8] Avellaneda, M., & Lee, J. H. (2010). Statistical arbitrage in the US equities market. *Quantitative Finance*, 10(7), 761-782.

[9] Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance*, *7*(1), 77–91. https://doi.org/10.2307/2975974

[10] Rosenberg, B. (1974). Extra-Market Components of Covariance in Security Returns.ă*Journal of Financial and Quantitative Analysis*,ă*9*(2), 263–274. doi:10.2307/2330104

[11] Sharpe, W. F. (1964). Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk. *The Journal of Finance*, *19*(3), 425–442. https://doi.org/10.2307/2977928

[12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

A Mathematical Principles of the Covariance Matrix-Based Prediction Model Framework

The linear multi-factor risk model can be expressed as:

$$V = XFX^T + \Delta$$

where V represents the individual stock covariance, F represents the factor covariance, and Δ represents the idiosyncratic variance. We decompose the prediction of V into predictions for F and Δ as follows:

1. Data Preparation: Input historical prior factor exposures X_{prior}, price-volume data S, and future stock returns r.

2. Implicit Factors: Map prior factors to implicit factors X using a multilayer perceptron. This step serves to reduce dimensionality by mapping prior factors and requires the learned implicit factors to be orthogonal to each other.

3. Linear Regression: Perform linear regression on stock returns based on factor exposures:

$$r = Xf + u$$

Considering the heteroscedasticity of the data, a market-cap weighted least squares method is used for regression:

$$f = (X^T W X)^{-1} X^T W r$$

to obtain factor combination weights W_f, pure factor returns f, and idiosyncratic returns u.

4. Feature Extraction: Use methods such as recurrent neural networks to extract stock features e_s from price-volume sequence data.

5. Factor Covariance Risk Prediction:

a. Aggregate stock features using pure factor combination weights to obtain pure factor features:

$$e_f = W_f \cdot e$$

b. Compute the inner product of pure factor features to construct the factor covariance matrix (using the Gram matrix method to ensure the semi-positive definiteness of the covariance matrix):

$$F_{ij} = e_{f_i}^T \cdot e_{f_j}$$

c. Perform eigenvalue decomposition on the factor covariance matrix (using backpropagation):

$$F = UU^T$$

d. Calculate the whitening matrix based on the obtained eigenvalues and eigenvectors:

$$P = U^{-\frac{1}{2}} U^T$$

then apply ZCA regularization to the pure factor returns:

$$f^{'} = U^{-\frac{1}{2}} U^T f$$

ZCA regularization aims to transform pure factor returns into regularized factor returns, whose covariance should be a unit diagonal matrix. PCA regularization cannot be used here as the basis obtained would vary with the factor covariance matrix, making it impossible to directly compute the covariance over a batch when calculating the loss value.

e. Compute the difference between the covariance matrix of regularized factor returns $F^{'} = f^{'} \cdot f^{'T}$ and the unit diagonal matrix, using it as the loss value for factor covariance risk prediction. The design of the loss function should ensure that the diagonal elements of $F^{'}$ are close to 1 and the off-diagonal elements approach 0.
One design is to ensure that the samples in a batch are temporally continuous, thus obtaining consecutive multi-day feature combination returns $f^{'}$ and using these to compute $F^{'}$; then separately compute the loss values for diagonal and off-diagonal elements. Use $Q = f_i'^2 - \ln(f_i'^2)$ as the loss function for diagonal elements to impose a greater penalty for underestimating risk, and use $\frac{1}{k(k-1)} \sum_{i \neq j} |f_i' f_j'|$ as the loss function for off-diagonal elements to drive them towards 0.

f. Further improvements can be made to ensure that the true off-diagonal elements of the factor covariance matrix are 0:

i. In step 4, Feature Extraction, incorporate the attention mechanism of the Deep Risk Model (DRM) to reduce covariance between different stocks, thus identifying differences among similar stocks. For example, within an industry, some

stocks may be mainly influenced by market cap factors, while others may be influenced by debt ratio.

ii. When computing the factor covariance matrix, use

$$F_{ij} = k(e_i, e_j) = \exp\left(\|e_i - e_j\|_2\right)$$

to obtain a nonlinear relationship, thereby reducing covariance arising from nonlinear relationships. This method may cause the gradient of the exponential function to decay rapidly, leading to low efficiency and slow optimization in backpropagation.

6. Prediction of Idiosyncratic Risk:

a. Compute the residuals between stock features and factor features, using these as idiosyncratic features:

$$e_u = e_s - W_f^T \cdot e_f$$

b. Calculate the idiosyncratic variance $Delta_{ii}$ from idiosyncratic features using methods such as inner products and multilayer perceptrons.

c. Compute the difference between the covariance matrix of idiosyncratic returns u and the predicted idiosyncratic variance Δ, using it as the loss value for idiosyncratic risk prediction. Since Δ has a high dimension, it may be difficult to optimize, so only the Q statistic of the diagonal elements is used as the loss value.

7. Individual Stock Covariance Prediction:

$$V = XFX^T + \Delta$$

Here, the sum of two semi-positive definite matrices is still a semi-positive definite matrix, thus satisfying the requirement of the semi-positive definite property of the covariance.

B References

Referenced the following third-party code: https://github.com/Mapledok/Deep-Risk-Model.