LSTM Mean-Variance Portfolio Optimization

MATH GR5430 Spring 2023 group H

Aime Bierdel	Dhyey Mavani	Daniel Schmidle	Boris Ter-Avanesov
ab4884	ddm2149	dps2150	bt2522

Abstract

Portfolio optimization is a crucial task in investment management, where asset managers attempt to maximize the returns while simultaneously minimizing the risk of their investments. In this paper, we introduce different adaptations of Long Short-Term Memory (LSTM) models to predict the expected returns and volatility preceding portfolio optimization with Mean-Variance Optimization (MVO). We compare the performance of the different LSTM-MVO models using historical data on selected assets and evaluate their effectiveness on portfolio optimization. Our results show that LSTM models can predict the expected returns and volatility better than historical performance alone, which suggests ML models have a potential ancillary place in portfolio management.

Introduction

Mean-Variance Optimization (MVO) is an industry standard approach for portfolio optimization that prioritizes diversification and balancing the risk and reward, when selecting and allocating assets. In MVO, optimization involves identifying the portfolio's expected returns and volatility, or standard deviation, and then allocating different assets in the portfolio in such a way that the expected returns are maximized for a given level of risk, or the risk is minimized for a given level of expected return. MVO has been a popular and effective technique for portfolio optimization for several decades. One of the main reasons why MVO has done well over the years is because it is based on rigorous statistical principles and provides a systematic approach to portfolio construction. Although MVO has proven to be a valuable framework for portfolio optimization, it is not immune to limitations and criticisms. One of the key criticisms of MVO is its reliance on historical data to estimate expected returns and risk. MVO's overly reliant use of historical data has been proven to be problematic as it may not accurately reflect future performance. Thus, this project proposes an ancillary process to MVO where a predictive LSTM model is utilized in forecasting the expected returns and volatility of the assets for MVO allocations. In this research, several variations of LSTM models are explored as feeder components to an MVO portfolio.

Data

The data used in this study was collected from Yahoo Finance through the yfinance API for the 10year period of 5/6/2013 to 5/2/2023. Weekly data was chosen for the ability to

predict longer term forward returns than daily data, which would result in less required portfolio rebalances. The chosen universe consists of ['GOOGL', 'AAPL', 'MSFT', 'DIS', 'WBD', 'V', 'JNJ', 'PG', 'AMZN', 'UNH', 'HD', 'CVX']. This set of assets was chosen as a subset to the S&P 500 and represents a diversified set of companies from various sectors and industries. Many of the stocks in this portfolio are from high-growth industries such as technology and e-commerce, which may continue to show strong growth potential in the future. Additionally, several of the stocks are considered to be blue-chip stocks with a long history of stable earnings. The data was preprocessed, checking for any missing values, but none were found. We split the data into train, validation, and test sets by 80/10/10 split, with dates of train: 5/6/13-4/26/2021, validation: 5/3/2021-5/2/2022, and test: 5/9/22-5/1/23. The training and validation sets were utilized for the base historical MVO model and to fit the LSTM models and tune hyper parameters. The test set was utilized to measure out of sample results for comparing model and strategy efficacy.

Methodology

To test against the base MVO model, this project considers 4 versions of LSTM models: A base LSTM | LSTM+GRU | LSTM+GRU+ATTENTION | LSTM+CNN and a LSTM-based transformer (DTML). We utilize the LSTM models to predict the expected returns and volatility of each asset in the universe over the test period 5/9/22 - 5/1/23, to compare against the classical MVO approach of using historical data to estimate the expected returns and volatility. To better illustrate the methodology, we will break down each component of this model in logical order of conceptual structure.

Mean-Variance Optimization

In Mean-Variance Optimization (MVO), the optimal portfolio is the one that lies on the efficient frontier, which is the set of portfolios that offer the highest expected return for a given level of risk. The MVO portfolio is defined by calculating,

$$\mathbb{E}(R_p) = \sum_i w_i \mathbb{E}(R_i)$$

Where $R_p$is the return on the portfolio, R_i is the return on asset i, and w_i is the weight component of asset i, And

$$\sigma_p^2 = \sum_i w_i^2 \sigma_i^2 + \sum_i \sum_{j \neq i} w_i w_j \sigma_i \sigma_j \rho_{ij}$$

where σ_p is the standard deviation of the periodic returns on the asset, and ρ_{ij} is the correlation coefficient between the returns on assets i and j. The output weights of a Mean-Variance Portfolio optimization algorithm represent the optimal allocation of investment

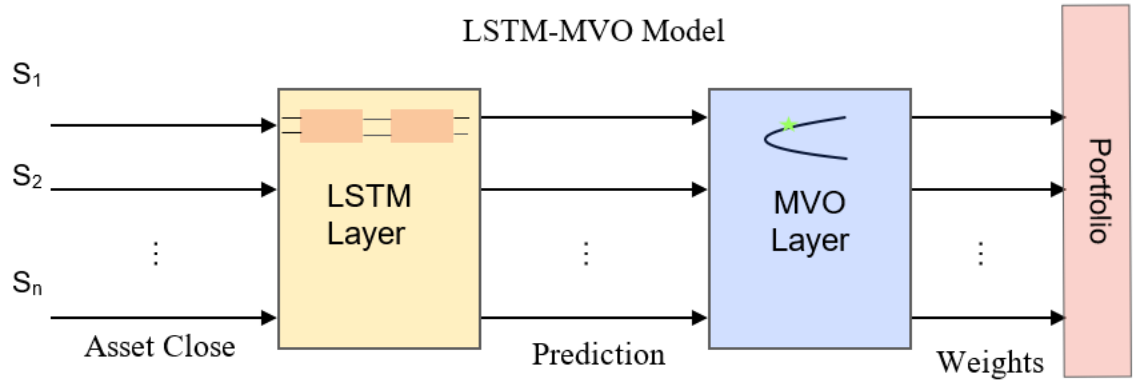
among the assets in the investment universe to achieve the investor's risk and return objectives. MVO utilizes the historical data of the asset to calculate these values however, the use of historical prices for mean-variance optimization can be problematic as it assumes that the past will repeat itself, neglects the possibility of changes in the covariance matrix of asset returns, and may not accurately reflect future expected returns and risk. As a result, alternative approaches, such as the use of machine learning models, may be more appropriate for estimating expected returns and risk in portfolio optimization.

Sharpe ratio

All base and LSTM methods of MVO will utilize the Sharpe ratio to calculate the optimal weights of each portfolio. The Sharpe ratio represents the amount of excess return that the investor is receiving for each unit of risk taken on by the portfolio. The Sharpe ratio was chosen for this project as the higher the Sharpe ratio, the better the tradeoff between risk and return, and that it provides an industry/research standard on which to compare methods.

LSTM

By replacing the historical expected returns with the ML predicted values, this project seeks to enhance the optimization process with portfolio weights being calculated on anticipated data rather than historical. The LSTM model was chosen for this project as they have the ability to capture the complex relationships and patterns present in historical data, giving them the potential to make accurate predictions about future prices. In the proposed LSTM feeding MVO process, historical financial data is used to train an LSTM model to forecast the expected returns and volatility of the assets in the portfolio. The LSTM model is designed to capture the temporal dependencies in the data and can learn to make predictions based on the patterns and trends observed in the historical data. Once the LSTM model is trained, the forecasted returns and volatility are used as inputs to the MVO process to determine the optimal portfolio weights. The MVO process is then optimized to maximize the Sharpe ratio of the portfolio, considering the forecasted returns and volatility from the LSTM model. The output of the LSTM feeding MVO process is an optimized portfolio that is expected to deliver the highest risk-adjusted returns based on the forecasted returns and volatility.



LSTM Versions

For all models TensorFlow / Karas were utilized for build and training. To reduce time complexity, early stopping was applied, measuring the validation loss. Data for the LSTM models were normalized to ensure consistent range, avoiding saturation, and improved convergence so that the LSTM models were able to effectively learn patterns in the data. Manual tuning on the hyperparameters was conducted with optimal chosen for final production.

LSTM Base Model

The base LSTM is built as a Keras Sequential model designed to train an LSTM neural network for handling time series data. It consists of three layers: an LSTM layer with 64 units and L1/L2 regularization, a SpatialDropout1D layer with 0.15 dropout rate, and a Dense layer with 1 unit and linear activation. The LSTM layer uses tanh activation and SELU recurrent activation, Glorot uniform kernel initializer, and orthogonal recurrent initializer. The model is intended to learn patterns in time series data and predict a single output value based on a sequence of input values, and the regularization techniques can help prevent overfitting and improve generalization performance.

LSTM + GRU

A stacked Recurrent Neural Network (RNN) model consisting of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers, along with a Spatial Dropout layer and a Dense output layer. The use of LSTM and GRU layers allows the model to capture temporal dependencies in the input data, while the Spatial Dropout layer helps prevent overfitting during training. Finally, the Dense output layer is used to make the final prediction of the time series value. The use of Bidirectional layers can also help improve the performance of the model by allowing the model to access both past and future time steps when making a prediction.

LSTM + GRU + ATTENTION

This model uses a combination of LSTM and GRU layers along with an attention mechanism. Specifically, it's a stacked RNN model consisting of LSTM and GRU layers with Layer Normalization and Dropout layers for regularization. The attention mechanism is implemented using a Multi-Head Attention layer which allows the model to selectively focus on important time steps while making a prediction. Finally, the model is trained to make a single point forecast, represented by the output of a Dense layer with a single unit. The use of Bidirectional layer with GRU layer helps to capture past and future context of the input sequence.

LSTM + CNN

The architecture consists of four layers that are stacked sequentially to extract and analyze the temporal patterns in the data. The first layer is a 1D convolutional layer that uses the rectified linear unit (ReLU) activation function to extract local patterns in the data. The second layer is a Long Short-Term Memory (LSTM) layer that uses the hyperbolic tangent

(tanh) activation function and a scaled exponential linear unit (SELU) as the recurrent activation function to capture long-term temporal dependencies in the data. The third layer is a Bidirectional Gated Recurrent Unit (GRU) layer that processes the output sequences from the LSTM layer in both forward and backward directions to capture dependencies in both past and future time steps. The fourth layer is a SpatialDropout1D layer that helps to prevent overfitting by randomly dropping out input features during training, followed by a TimeDistributed Dense layer that generates a scalar output for each time step in the input data.

DTML - Transformer with Multi-Level Contexts (omitted from results)

The DTML model we use is a replication and adaptation of "Accurate Multivariate Stock Movement Prediction via Data-Axis" (Yoo et al. 2021). It adopts a transformer structure where time series are encoded using an attention bi-directional LSTM on every lag of every stock. The obtained hidden states are then stacked into a context matrix that represents the encoded panel of stocks throughout the period. The decoder is a multi-head attention layer that learns the optimal weights to put on the context matrix to predict any given stock. It learns correlations between stocks in a flexible way. As the authors do not make their code available, we replicated the full model using Pytorch and adapted it to forecast stock returns. The implementation was very challenging as we encountered all the training problems that come with transformer models. In particular, convergence, vanishing gradients and gradient imbalance issues are still present in the enclosed version of the code. We tried to deal with them by implementing residual connections, layer normalizations and learning-rate warm-up techniques but the issues persist in the decoder part of the model. Therefore, we kept the code separate from the other models as it produces essentially flat forecasts as a result of these issues. A thorough description and discussion of our implementation is present in the enclosed "Transformer.ipynb" notebook that runs the transformer model.

Results

Model performance is measured by the portfolio returns given model weights over the test period 5/9/22 – 5/1/23. The portfolios were balanced once given the information before the test period. We begin by comparing a true Naïve base, where all stocks are given equal weights, to the performance of the MVO model. This allows us to establish the efficacy of the MVO model. The results show MVO outperformed the Naïve base and thus sets an industry-standard high watermark to compare the LSTM models against.

In the Final results we see that the LSTM, +GRU, and +CNN models all performed similarly to the MVO model. The exception being the highest volatility +GRU+ATTN model that performed substantially worse later in the test period but substantially better during early parts of the test period. The base LSTM model performed slightly worse than the MVO model at all points and illustrates the limitations of a simpler LSTM and the necessity to add complexity to the model given the complex data. The +GRU and +CNN models showed the most promise for outperformance of the MVO model as they both alternated for lead against MVO. Comparing max drawdown of all models shows similar results of all models,

except for the high variance +GRU+ATT model. This suggests the non+GRU+ATT models have similar risk appetite profile and should be considered in the same basket. Given the results, there is no decisive winner here but a lot of potential for further research.

Discussion

The research has demonstrated the potential of using LSTM and MVO for financial forecasting and portfolio optimization. However, it is important to acknowledge the limitations of this research and approaches.

Computational limitations

Given the scope of the project and how every section of the project feeds into another, we utilized Google Colab to work in parallel. The use of Colab resulted in some additional hurdles for the project as runtimes were long and Colab ends runtimes preemptively. This limited the scope of rebalances that were able to be run without Colab disconnecting in between sections. The results may have been better for all LSTM models if periodic retraining and rebalances (quarterly/monthly) were able to be performed. Thus, training on a local machine would be recommended for future research.

Model limitations

Another limitation of using LSTM is the potential for overfitting. To train the LSTM model, historical data is used to make predictions about future data. However, if the model is too complex or the training data is not representative of future data, the model may perform poorly. Additionally, LSTM models may struggle with capturing long-term dependencies within the signal to noise ratio of historical data, which may limit their ability to accurately predict trends and patterns. Other potential feeder models should be considered for future research.

Despite these limitations, we believe that MVO with predictive ML models holds promise for financial forecasting and portfolio optimization. Future research may focus on addressing these limitations and developing more robust and accurate models for financial analysis. Ultimately, the success of these approaches will depend on the ability to adapt to changing market conditions and incorporate new data sources and techniques as they become available.

Individual contributions:

Aime Bierdel contributed to the LSTM models with a focus on the DTML implementation.
Dhyey Mavani contributed to the MVO portion.
Daniel Schmidle connected the program elements, contributed to the Results, and wrote the paper.
Boris Ter-Avanesov contributed to the LSTM models with a focus on the non-DTML models.

All team members contributed to the planning and development of the project.