# Portfolio Optimisation using LSTM

Minna Zhang

May 31, 2024

## 1 Abstract

Firstly, the study selects 30 stocks from the SP500 index based on certain constraints. Secondly, the study trains a LSTM neural network using the previous 70 days of stock price data to predict the next day's stock prices. Meanwhile, the study utilizes the shrinkage method to estimate the covariance of the assets. Thirdly, the study uses the mean-variance optimization method to calculate the optimal portfolio weights for each day.

## 2 Motivation

- **RNN** RNN processes inputs one at a time and then return output in the output layer. In the meantime, the output of intermediate layers of the previous step will be delivered to the next layer as part of the input. This ongoing recursive process continues for every element of a sequence until we obtain a prediction of the final output.
- **LSTM** Long Short-Term Memory networks usually called LSTMs are a special kind of RNN capable of learning long-term dependencies. LSTMs have a chain-like structure, but the repeating module has a slightly different structure. There are multiple layers which interact in a very special way. A common LSTM architecture is composed of a memory cell, an input gate, an output gate and a forget gate. Three of the gates can be thought of as a conventional artificial neuron, as in a multi-layer or feedforward neural network that computes using an activation function of a weighted sum.
- **Why LSTM** In deep learning, there are two factors that affect the magnitude of gradients - the weights and the activation functions, especially their derivatives, which the gradient passes through. If either of these factors is smaller than 1, then the gradients may vanish in time; if larger than 1, then exploding might happen. In the recurrency of the LSTM the activation function is an identity function with a derivative of 1.0. So, the backpropagated gradient neither vanishes nor explodes when passing through, but remains constant. The effective weight of the recurrency is equal to the forget gate activation. So, if the forget gate is on (activation close to 1.0), then the gradient does not vanish. Since the forget gate activation is never greater than 1.0, the gradient cant explode either. Thats why LSTMs are so good for learning long range dependencies and are well-suited to classify process and predict time series given time lags of unknown size and duration between important events. This is how LSTMs were developed to deal with the exploding and vanishing gradient problem when training traditional RNNs.

### 2.1 RNN Model Design

Choose an RNN architecture, typically LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Units), because of their ability to capture temporal dependencies and avoid the vanishing gradient problem common in standard RNNs. (We choose LSTM for our coursework)

- Input Layer: Feeds the sequence data into the model.
- Hidden Layers: Multiple LSTM layers can be utilized to learn the data's high-level features.
- Output Layer: A dense layer with one neuron to output the predicted P&L for the next day.

The RNN is trained using a backpropagation algorithm over the historical data, adjusting the model weights based on prediction errors, optimizing perhaps through Mean Squared Error (MSE) or another loss function that quantifies the difference between predicted P&L and actual P&L.

Once trained, the RNN model can predict the daily P&L for the stock portfolio. By running the trained RNN with the most recent data, it can generate P&L forecasts that help financial analysts and portfolio managers in decision-making processes.

# 3 LSTM Network Components

An LSTM (Long Short-Term Memory) network is a type of RNN (Recurrent Neural Network) suited for handling sequences and time-series data. It has a unique structure that helps it remember long-term dependencies and avoid issues like the vanishing gradient problem.
The LSTM-cell consists of a forget gate, an input gate, and an output gate. The main purpose of the LSTM-cell is to remember information over time intervals while the gates control the information that flows in and out of the cell. Essentially, the gates compute which information that is to be remembered and which information that is to be forgotten. The information that is remembered in the LSTM-cell is stored in its cell state $C_t$ and is calculated in accordance with the input $x_t$, the computation in the forget gate $f_t$ as well as the previous output $h_{t-1}$.

### Forget Gate:

- The forget gate decides what information should be discarded from the cell state. It looks at the previous hidden state ($h_{t-1}$) and the current input ($x_t$), and outputs a number between 0 and 1 for each number in the cell state ($C_{t-1}$).
- **Formula:** $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

### Input Gate:

- The input gate decides which new information is going to be stored in the cell state. It creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state.
- **Formula:** $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
- $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$

### Cell State Update:

- The cell state, $C_t$, is updated by forgetting the things decided to forget earlier and then adding new candidate values scaled by how much we decided to update each state value.
- **Formula:** $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

### Output Gate:

- The output gate decides what the next hidden state, $h_t$, should be. The hidden state contains information about previous inputs. The contents of the cell state are usually filtered by the output gate to decide what should be passed to the output.
- **Formula:** $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
- $h_t = o_t * \tanh(C_t)$

# 4 Essay structure: Introduction

This guide replicates the methodology used in the study "LSTM-based Portfolio Optimization Strategy for SP500," which combines Long Short-Term Memory (LSTM) neural networks with the Ledoit-Wolf shrinkage method and mean-variance optimization for portfolio management.

# 5 Data Collection and Preprocessing

## 5.1 Data Source

The daily stock data is acquired using the Python package `yfinance`, which sources data from Yahoo Finance. The list of SP500 component stocks is obtained from Wikipedia.

## 5.2 Data Period

TBD

## 5.3 Stock Selection Criteria

**Below is the criteria from the essay, we can change the length of the training period and the number of stocks.**

1. Rank stocks based on their average daily returns during the 70-day training period.
2. Select the top 30 performing stocks ensuring sectoral diversification:
    - Select a minimum of two and a maximum of five stocks from each GICS sector.
3. Exclude stocks with high volatility (stocks that experienced a consecutive three-day decline in the past seven days).

## 5.4 (TBD: Data Preprocessing - Normalization)

Before feeding the data into the RNN, it undergoes several preprocessing steps:

- **Normalization** Scaling the data to a common scale without distorting differences in the ranges of values to help the neural network converge more quickly. (e.g. Adjust numbers to make them comparable, scaling prices between 0 and 1)
- **Sequencing** Arrange data into sequences that the LSTM can learn from, similar to organizing events for a narrative. Creating sequences of past stock data (e.g., closing prices for the past 60 days) to predict the next day's P&L.
- **Split to Train-Test Data**

## 5.5 (TBD: Data Preprocessing - Standardizing Data)

To ensure uniformity, the stock price data is standardized:

$$X_{std} = \frac{X - \mu}{\sigma} \tag{1}$$

where $X$ is the original data, $\mu$ is the mean, and $\sigma$ is the standard deviation.

## 5.6 Training Data Preparation

Use the adjusted closing price as the daily price and compute daily returns:

$$R_{it} = \frac{V_{t+1}^i}{V_t^i} - 1 \tag{2}$$

where $V_t^i$ is the adjusted closing price of stock $i$ at time $t$.

# 6 Training the LSTM

This step focuses on optimizing the LSTM network by adjusting its weights through a process known as Backpropagation Through Time (BPTT). This involves:

- **Backpropagation Through Time (BPTT):** Update network weights by calculating gradients of the loss function.
- **Loss Function:** Typically Mean Squared Error (MSE) for regression tasks.
- **Formula:** $MSE = \frac{1}{n}\sum(y_{\text{actual}} - y_{\text{predicted}})^2$
- **Optimizer:** Methods like Adam or SGD to adjust weights based on gradients.

## 6.1 LSTM Model Structure

The LSTM model is constructed using TensorFlow's Sequential API:

- Two LSTM layers, each with 100 units.
- Three dense layers with 100, 50, and 1 unit respectively.
- Train the model for 10 epochs using mean squared error (MSE) as the evaluation metric.

# 7 Making Predictions

Once the LSTM is trained, this step uses the model to make forward-looking predictions based on new data sequences. This is the direct application of the LSTM for forecasting, using the learned patterns to predict future values of, for example, stock prices or market volatility.

- **Feedforward Through LSTM:** Process new data sequences through the LSTM to generate predictions.
- Inputs: A vector of 7-dimension, $x_t = (x_{t-6}, \ldots, x_t)$, where $x_i$ denotes the adjusted closing price for day $t$.
- Outputs: Prediction for the next day, $\hat{x}_{t+1}$.
- Convert predicted prices to daily returns for portfolio optimization.

# 8 Preventing Overfitting in LSTM Models

**ONLY DO THIS PART IF WE HAVE TIME**
Overfitting is a critical issue where a model learns the details and noise in the training data to an extent that it negatively impacts the performance of the model on new data. This step involves techniques such as:

- **Early Stopping:** To avoid overfitting, implement the early-stopping method, which halts the training process when the loss on the validation set no longer shows improvement, thus preserving the model's ability to generalize.
- **Dropout:** Utilize dropout techniques that randomly ignore selected neurons during the training process, effectively thinning the network temporarily and reducing the risk of overfitting. This randomness helps prevent co-adaptations on training data.

# 9 Covariance Estimation using Ledoit-Wolf Shrinkage

## 9.1 Sample Covariance Matrix

Calculate the sample covariance matrix for asset returns using historical data.

### 9.2 Shrinkage Method

Apply the Ledoit-Wolf shrinkage method to form a new covariance matrix:

$$\Sigma_{LW} = \delta F + (1 - \delta)S \tag{3}$$

where $\delta$ is the shrinkage constant, $F$ is the structured estimator, and $S$ is the sample covariance matrix.

## 10 Mean-Variance Optimization

### 10.1 Objective Function

Derive the optimal portfolio weights using the mean-variance model:

$$\min_{w} \; w^\top \Sigma w - qR^\top w \tag{4}$$

where $w$ is the vector of asset weights, $\Sigma$ is the covariance matrix, and $q$ measures investor's risk tolerance.

### 10.2 Daily Portfolio Update

Update portfolio weights daily based on LSTM predictions. If predicted returns are lower than the risk-free rate, set all asset weights to zero.

## 11 Evaluation and Benchmarking

### 11.1 Performance Metrics

Compare the optimized portfolio against benchmarks (SP500, 1/N portfolio, and other mean-variance variants) using metrics such as cumulative returns, volatility, Sharpe ratio, and max drawdown.

### 11.2 Graphical Representation

Plot cumulative returns of each portfolio over the testing period. Generate annualized tear sheets to summarize performance metrics.

## 12 Conclusion and Future Research

### 12.1 Summary of Findings

Highlight the performance of the proposed model in terms of returns, risk control, and volatility invariance.

### 12.2 Future Work

Discuss limitations like the exclusion of transaction fees and suggest future research areas like extending the model to longer time horizons and higher trading frequencies.