The Journal of Supercomputing (2020) 76:8040–8057
https://doi.org/10.1007/s11227-018-2577-1

Threshold-based portfolio: the role of the threshold and its applications

Sang Il Lee[1] · **Seong Joon Yoo**[1]

Published online: 14 September 2018

Abstract

This paper aims at developing a new method by which to build a data-driven portfolio featuring a target risk–return. We first present a comparative study of recurrent neural network models (RNNs), including a simple RNN, long short-term memory (LSTM), and gated recurrent unit. The models are applied to the investment universe consisted of 10 stocks in the $S\&P500$. The experimental results show that the LSTM-based prediction model outperforms the others in terms of hit ratio of 1-month-ahead forecasts. We then build predictive threshold-based portfolios (TBPs) that are subsets of the universe satisfying given threshold criteria for the LSTM-based return forecasts. The TBPs are rebalanced monthly to restore equal weight to the constituents of the TBPs. We find that the risk and return profile of the realized TBP represents a monotonically increasing frontier on the risk–return plane, where the equally weighted universe portfolio plays a role in the lower bound of TBPs. This shows the availability of TBPs in targeting specific risk–return levels, and the EWP of an universe plays a role in the reference portfolio of the TBPs. In the process, thresholds play dominant roles in characterizing risk, return, and the prediction accuracy of the TBPs. The TBP is more data-driven in designing portfolio return and risk than existing ones, in the sense that it requires no prior knowledge of finance such as financial assumptions, financial mathematics, or expert insights. For practical uses, we present a multiperiod TBP management method and also discuss the application of TBP to mean–variance portfolios to reduce estimation risk.

Keywords Portfolio management · Recurrent neural networks · Efficient frontier · Financial time series

✉ Seong Joon Yoo
sjyoo@sejong.ac.kr

Sang Il Lee
silee@ajou.ac.kr; silee@sejong.ac.kr

[1] Department of Computer Engineering, Sejong University, Seoul 05006, Republic of Korea

1 Introduction

Today, machine learning has come to play an integral role in many parts of the financial ecosystem, from portfolio management and algorithmic trading, to fraud detection and loan/insurance underwriting. Time series are one of the most common data types encountered in finance, and so time-series analysis based on econometrics has been widely used in finance and economics. The development of machine learning algorithms has opened a new vista for modeling the complexity of financial time series as an alternative to the traditional econometric models, by effectively combining diverse data and capturing nonlinear behavior. For this reason, financial time-series modeling has been one of the most interesting topics that has arisen in the application of machine learning to finance. Researchers have successfully modeled financial time series by focusing primarily on prediction accuracy or automatic trading rules [1–16]. Nevertheless, financial modeling and applications remain daunting, given the difficulties arising from the essentially nonlinear, complex, and evolutionary characteristics of the financial market.

On the other hand, asset allocation has been traditionally considered an issue central to investment and risk management. Markowitz [17] was the first to introduce a rigorous mathematical framework for allocation, called modern portfolio theory (MPT). Based on a mean–variance optimization technique, MPT provides a method by which to assemble assets and maximize the expected return of the portfolio for a given level of risk. Following Markowitz's thinking, new portfolio models have been subsequently proposed for more practical use and to achieve a better understanding of portfolios. Examples include the three-factor asset pricing model [18], the Black–Litterman model [19], the resampled efficient frontier model [20], the global minimum variance model [21], the maximum diversification portfolio [22], and the risk-parity portfolio [23,24]. Additionally, dynamic/tactical asset allocations based on simple rules or market anomalies were developed to automatically adjust portfolios in response to market changes [25–28]. These studies show that today there is general consensus about the importance of effective combinations of assets.

In this study, we propose a new method for constructing a data-driven portfolio using recurrent neural networks (RNNs)-based future return predictions. Throughout this study, we will refer to this portfolio as a threshold-based portfolio (TBP), since its properties are characterized by the threshold levels imposed on predicted returns. In particular, this study makes the following main contributions:

- It examines the ability of RNNs to forecast 1-month-ahead stock returns.
- It develops a new asset allocation model, called TBP, and analyzes their properties.
  - The threshold can be used for a parameter to draw a TBP frontier that comprises the set of TBP points on a risk–return plane. This implies that one can build a portfolio with a specific risk–return level by selecting the appropriate threshold level.
  - The equally weighted portfolio (EWP) is the lower bound of the TBPs on the TBP frontier. This implies that the TBPs can be characterized with respect to a reference portfolio, EWP.

- For practical uses of the TBPs, it develops a TBP management process for pursuing specific risk–return levels over multiple-periods and also the method incorporating TBPs into MPT.

The remainder of this paper is organized as follows: Sect. 2 discusses some of the important work related to this area. Section 3 explains the simple recurrent neural network (S-RNN), long short-term memory (LSTM), and gated recurrent units (GRU). Section 4 provides experimental results regarding the prediction accuracy of the models and the performance of TBPs. In Sect. 5, we discuss the practical applications of TBPs. Finally, in Sect. 6, we conclude this paper and discuss possible future extensions of our work.

2 Related work

Using conventional econometric models, financial economists have found there to be statistically significant relationship between stock returns and lagged variables. For example, Campbell et al. [29] investigated the relationship between aggregate stock market trading volume and the serial correlation of daily stock returns. They provide an evidence that a stock price decline on a high-volume day is more likely than a stock price decline on a low-volume day to be associated with an increase in the expected stock return. Choueifaty and Coignard [30] show that trading volume is a significant determinant of the lead-lag patterns observed in stock returns.

For this reason, we select RNN algorithms; these are superior for modeling time-lag effects in multi-dimensional financial time series, by virtue of feeding the network activations from a previous time step as inputs into the network, to influence predictions in the current step. In contrast, feed forward neural networks (FFNNs) are not appropriate for capturing these time-dependent dynamics: they operate on a fixed-size time windows, and so they can provide only limited temporal modeling. RNNs are less commonly applied to financial time-series predictions, yet some recent studies has shown promising results for use in financial time-series prediction. We discuss some related work on stock prediction using RNN networks and also recent applications of machine learning to portfolio construction.

2.1 Financial time series prediction using RNNs

Fischer and Krauss [13] deployed LSTM networks to predict one-day-ahead directional movements in a stock universe and constructed subset portfolios by selecting constituents outperforming the cross-sectional median return of all stocks in the next day. They found that LSTM network-based portfolios outperform those of memory-free classification algorithms, including a random forest (RAF), a deep neural net (DNN), and a logistic regression classifier (LOG), on measures of the mean return per day, annualized standard deviation, annualized Sharpe ratio, and accuracy.

More recently, Bao et al. [31] developed a hybrid model called the WSAEs–LSTM, combined with wavelet transforms (WT), stacked autoencoders (SAEs), and LSTM to effectively combine diverse financial data, including historical trading data of open

price, high price, low price, closing price, and volume and technical indicators of stock market indexes and macroeconomic variables. The experimental results show that it produces more accurate one-day-ahead stock price predictions than the other similar models including WLSTM (i.e., a combination of WT and LSTM), RNN, and LSTM.

2.2 Machine learning prediction-based investment portfolios

Freitas et al. [32] present a new model of prediction-based portfolio optimization for capturing short-term investment opportunities. For the universe of Brazilian stocks, they combine their neural network predictors featuring normal prediction errors with the mean–variance portfolio model and show that the resulting portfolio outperforms the mean–variance model and beats the market index. More recently, Mishra et al. [33] developed a novel prediction-based mean–variance (PBMV) model, as an alternative to the conventional Markowitz mean–variance model, to solve the constrained portfolio optimization problem. More specifically, they present a low-complexity heuristic functional link artificial neural network (HFLANN) model to overcome the incorrect estimation taken as the mean of the past returns in the Markowitz mean–variance model and carry out the portfolio optimization task by using multi-objective evolutionary algorithms (MOEAs). Ganeshapillai et al. [34] propose a machine learning-based connectedness matrix for addressing the shortcomings of correlation in capturing events such as large losses. They show that the matrix can be used to build portfolios that not only "beat the market," but also outperform optimal (i.e., minimum–variance) portfolios.

The results of these studies show that machine learning-based estimations can be effectively used to overcome certain limitations inherent in traditional methods. Our work is in line with this motive, but is more dependent on machine learning in the sense that we only use machine learning-based return forecasts for aggregating stocks rather than incorporating machine learning to existing theoretical frameworks. This fact makes TBPs more data-driven than existing models.

3 Models: S-RNN, LSTM, GRU

S-RNNs [35] are an extension of a conventional FFNN that adds a feedback connection to a feedback network consisting of three layers: an input layer, a hidden layer, and an output layer. However, Bengio et al. [36] found it is difficult to train an S-RNN to capture long-term dependencies, because the gradients tend to either vanish or explode. Alternatively, LSTM [37] and GRU [38] have been proposed to overcome the problem by using a "gating" approach. The LSTM algorithm is local in space and time [37], which means that the computational complexity of learning LSTM models per weight and time step with the stochastic gradient descent (SGD) optimization technique is O(1), and the learning computational complexity per time step is O(W), where W is the number of weights. Hence, the LSTM, which is used to construct TBPs, is capable of handling large-scale data, as its computational complexity grows linearly with respect to the length of the input data.

3.1 LSTM architecture

LSTMs can effectively learn important pieces of information that may be found at different positions in the financial time series, by controlling what is added and removed from memory in the hidden layers. This is conducted by using a combination of three gates: (1) a forget gate, (2) an input gate, and (3) an output gate.

Forget gate An LSTM cell receives the current input $x_t \in \mathbb{R}^d$, the hidden state vectors $\mathbf{h}_{t-1} \in \mathbb{R}^n$, and a cell state $\boldsymbol{C}_{t-1} \in \mathbb{R}^n$ at time $t-1$. The forget gate then is then calculated as

$$f_t = \sigma(\boldsymbol{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f), \tag{1}$$

where:

- $\mathbf{W}_f \in \mathbb{R}^{n \times d}$ is the weight matrix from the input $\mathbf{x}_t$ to the forget gate $\mathbf{f}_t$,
- $\mathbf{U}_f \in \mathbb{R}^{n \times n}$ is the weight matrix from the previous hidden vector $\mathbf{h}_{t-1}$ to the forget gate $\mathbf{f}_t$,
- $\mathbf{b}_f \in \mathbb{R}^n$ is the forget gate bias,
- $\mathbf{f}_t \in \mathbb{R}^n$ is the output of the gate, which determines the amount to be erased from the previous cell state, and
- $\sigma(\cdot)$ is a sigma function.

Input gate The input gate $\mathbf{i}_t$, which is used to scale the candidate update vector $\widetilde{\mathbf{C}}_t \in \mathbb{R}^n$, determines what parts of $\widetilde{\mathbf{C}}_t$ are added to the corresponding memory cell element at time t, based on the recurrent connection from the hidden vector $\mathbf{h}_{t-1}$ and the input at time t, $\mathbf{x}_t$:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i), \tag{2}$$

$$\widetilde{\mathbf{C}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c), \tag{3}$$

where:

- $\mathbf{W}_i \in \mathbb{R}^{n \times d}$, $\mathbf{U}_i \in \mathbb{R}^{n \times n}$, and $\mathbf{b}_i \in \mathbb{R}^n$ are the input gate parameters,
- $\mathbf{W}_c \in \mathbb{R}^{n \times d}$, $\mathbf{U}_c \in \mathbb{R}^{n \times n}$,
- $\mathbf{b}_c \in \mathbb{R}^n$ are the parameters for selecting a candidate state, $\widetilde{\mathbf{C}}_t$, and
- $\tanh(\cdot)$ is the tanh function.

Then, the current state of the cell $\mathbf{C}_t \in \mathbb{R}^n$ is given by

$$\mathbf{C}_t = \mathbf{i}_t \odot \widetilde{\mathbf{C}}_t + \mathbf{f}_t \odot \mathbf{C}_{t-1}, \tag{4}$$

where $\odot$ represents the element-wise Hadamard product.

Output gate The output gate $\mathbf{o}_t \in \mathbb{R}^n$, which is used to calculate the output $\mathbf{h}_t \in \mathbb{R}^n$, determines the output from the current cell state:

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o), \tag{5}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t), \tag{6}$$

where $\mathbf{W}_o \in \mathbb{R}^{n\times d}$, $\mathbf{U}_o \in \mathbb{R}^{n\times n}$, and $\mathbf{b}_o \in \mathbb{R}^n$ are the output gate parameters. The hidden vector $\mathbf{h}_t$ of the memory cell can be used as the final output of the network.

3.2 GRU architecture

The structure of a GRU can be expressed as follows:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z), \tag{7}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r), \tag{8}$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tanh[\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h], \tag{9}$$

where:

- $\mathbf{x}_t$, $\mathbf{h}_t$, $\mathbf{z}_t$, and $\mathbf{r}_t$ are the input vector, output vector, update gate vector, and reset gate vector, respectively, and
- $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are forward matrices, recurrent matrices, and biases, respectively.

4 Experiment

4.1 Data

4.1.1 Universe

The asset universe consists of the top 10 stocks in Standard and Poor's 500 index ($S\&P500$):

- Apple (AAPL), Amazon (AMZN), Bank of America Corporation (BAC), Berkshire Hathaway Inc. Class B (BRK-B), General Electric Company (GE), Johnson & Johnson (JNJ), JPMorgan Chase & Co. (JPM), Microsoft Corporation (MSFT), AT & T Inc. (T), and Wells Fargo & Company (WFC).

We use data from January 1997 to December 2016 from Yahoo Finance. The daily stock dataset contains five attributes: open price, high price, low price, adjusted close price, and volume (OHLCV). Figure 1 graphically shows the normalized close price (i.e., subtract the mean from each original value and then divide by the standard deviation). We convert the daily OHLCV dataset to four different monthly OHLCV datasets by calculating the last, mean, maximum, and minimum values of the daily OHLCV dataset per month. Each monthly OHLCV dataset is used as a raw dataset for forecasting 1-month-ahead return at the end of each calendar month.

4.1.2 Preprocessing

To achieve higher quality and reliable predictions, the five attributes are preprocessed as a percentage change, $(x^{(t)} - x^{(t-1)})/x^{(t-1)}$. All data were divided into a training dataset (70%) to fit the model parameters and the test set (30%) for an out-of-sample

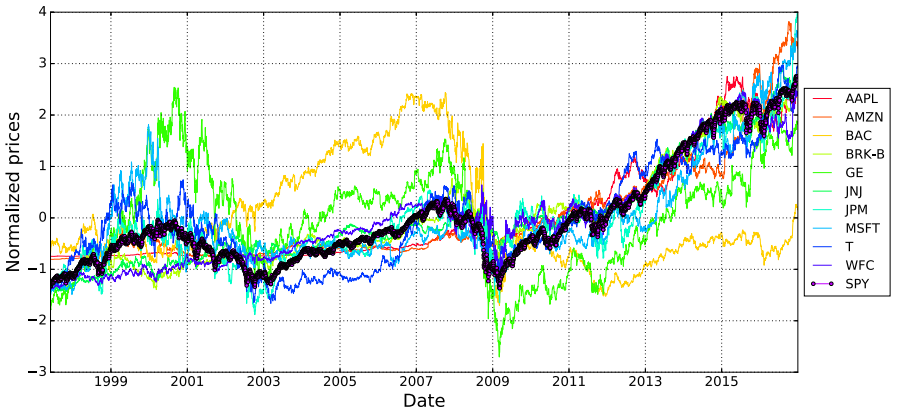

Fig. 1 (Color online) Normalized stock prices for the 10 stocks over the whole period

model evaluation. The 30% of the training set is used as the validation to evaluate a given model during training.

The statistical characteristics of the train data are shown in Table 1. We observe that the data are roughly in the range of -1 to 1 which is a usual range of features in deep learning, save for abnormal trading volume from max values of the volume data.

4.2 Experimental design

We build S-RNN, LSTM, and GRU architectures for 1-month-ahead forecasts of stock returns. Based on the validation set evaluation, we carried out a grid search to find the best hyperparameters over the number of RNN hidden layers (1,2, or 3) and the number of hidden units per layer (8, 16, 32, 64, or 128), and whether dropout is used to avoid overfitting of the models. The whole networks were trained by a backpropagation algorithm by minimizing the quadratic loss value, $L = \frac{1}{2}\sum_t^T (r(t) - \hat{r}(t))^2$ (where $\hat{r}(t)$ is the output of the last layer and $r(t)$ is the corresponding target) on the test set. The efficient ADAM (adaptive moment estimation) optimization algorithm [39] with a learning rate of 0.001 is used to fit the models in mini-batches of size 20. From the experiments, we specified the topology consisting of an input layer, an RNN layer with $h = 36$ hidden neurons, a 50% dropout layer, and an output layer with a linear activation function for regression.

The feature vectors to feed the models are overlapping sequences of 36 consecutive points (i.e., trading months in 3 years) in the preprocessed monthly OHLCV. The sequences themselves are sliding windows shifted by 1 month for each time $t \geq 36$, that is, $\{\mathbf{x}_{t-35}, \mathbf{x}_{t-34}, \ldots, \mathbf{x}_t\}$.

The experimental setup is implemented over a laboratory prototype, equipped with an Intel quad core i7-6700 processor at 3.4 GHz, Nvidia GPU (i.e., GTX 1070), and 32 GB of RAM running the Ubuntu 16.04.2 LTS x86-64 Linux distribution. Prediction models are evaluated using Keras 2.0.4 [40] and TensorFlow 0.11.0. In our approach, the stage of modeling and forecasting contributes significantly to the overall processing time and, for one asset, is obtained in an approximate processing time of 109 s.

Table 1 Statistics of the preprocessed close price and volume

	AAPL	AMZN	BAC	BRK-B	GE	JNJ	JPM	MSFT	T	WFC
Train data										
Mean	0.01, 0.45	0.01, 0.12	0.01, 0.124	0.00, 0.21	0.00, 0.17	0.007, 0.186	0.00, 0.16	− 0.00, 0.13	− 0.00, 0.20	0.01, 0.13
Std.	0.16, 2.22	0.20, 0.59	0.08, 0.549	0.06, 0.79	0.07, 0.74	0.069, 0.901	0.11, 0.83	0.12, 0.73	0.10, 0.84	0.07, 0.62
Min.	− 0.57, − 0.88	− 0.41, − 0.75	− 0.22, − 0.76	− 0.12, − 0.76	− 0.17, − 0.617	− 0.157, − 0.66	− 0.28, − 0.80	− 0.34, − 0.80	− 0.18, − 0.76	− 0.16, − 0.84
Max	0.45, 16.68	0.62, 1.71	0.17, 2.198	0.26, 2.34	0.19, 2.93	0.174, 5.603	0.25, 5.17	0.40, 4.84	0.29, 4.56	0.23, 2.93
Validation data										
Mean	0.06, 0.31	− 0.00, 0.11	0.01, 0.152	0.00, 0.13	0.00, 0.082	0.00, 0.14	0.01, 0.21	0.00, 0.32	0.015, 0.20	0.01, 0.12
Std.	0.12, 1.06	0.14, 0.60	0.03, 0.822	0.02, 0.52	0.02, 0.393	0.027, 0.68	0.04, 0.77	0.05, 1.66	0.03, 0.878	0.02, 0.58
Min.	− 0.15, − 0.72	− 0.30, − 0.70	− 0.04, − 0.68	− 0.05, − 0.50	− 0.065, − 0.48	− 0.04, − 0.65	− 0.06, − 0.59	− 0.11, − 0.79	− 0.06, − 0.73	− 0.02, − 0.67
Max.	0.35, 5.03	0.36, 1.96	0.09, 3.62	0.05, 1.30	0.058, 1.19	0.05, 2.90	0.09, 2.85	0.08, 8.50	0.08, 4.30	0.07, 1.41
Test data										
Mean	0.03, 0.11	0.04, 0.36	− 0.00, 0.22	0.00, 0.27	− 0.011, 0.11	0.00, 0.10	0.00, 0.20	0.00, 0.15	0.00, 0.05	0.00, 0.23
Std.	0.12, 0.54	0.14, 1.40	0.21, 0.68	0.06, 1.06	0.11, 0.51	0.04, 0.51	0.11, 0.75	0.08, 0.70	0.06, 0.40	0.14, 0.80
Min.	− 0.32, − 0.82	− 0.25, − 0.72	− 0.53, − 0.60	− 0.14, − 0.86	− 0.27, − 0.58	− 0.12, − 0.61	− 0.23, − 0.73	− 0.16, − 0.69	− 0.15, − 0.63	− 0.35, − 0.74
Max.	0.23, 1.71	0.54, 6.45	0.73, 2.25	0.12, 5.64	0.25, 1.87	0.07, 1.49	0.24, 2.39	0.24, 3.05	0.09, 1.50	0.40, 3.65

Table 2 Mean and SD of hit ratios for each of the 10 assets. The mean and SD values in bold mean the best values

	Last	Mean	Max	Min
S-RNN	0.559, **0.040**	0.555, 0.066	0.555, 0.059	0.555, 0.048
LSTM	**0.604**, 0.042	0.536, 0.083	0.569, 0.048	0.584, 0.039
GRU	0.573, 0.053	0.550, 0.079	0.586, 0.049	0.575, 0.051

4.3 Prediction accuracy

We evaluate the predictive ability of the three models using the hit ratio, which is defined as follows:

$$\text{Hit ratio} = \frac{1}{N}\sum_{t=1}^{N-1} P_t, \tag{10}$$

where N is total trading months during the test period and P_t is defined as:

$$P_t = \begin{cases} 1 & \text{if } r_{t+1} \cdot \hat{r}_{t+1} > 0, \\ 0 & \text{otherwise,} \end{cases}$$

where y_{t+1} and $\hat{y}_{t+1}$ are the realized return at the last business day of month $t+1$ and the 1-month-ahead return predicted at the last business day of month t, respectively.

Table 2 shows the mean and standard deviation (SD) of the hit ratios for the 10 assets and combining the last business day and the LSTM model generates the best prediction accuracy value (0.604). Therefore, we will use the LSTM model and the last business day OHLCV of each month for building TBPs in the next subsections.

4.4 Role of threshold in TBP

We present the three types of TBPs imposing the positive and negative threshold levels (θ^+ and θ^-, respectively) on the predicted returns. Given the 1-month-ahead predicted return $\hat{r}_i$ of an asset $S_i (i = 1, 2, \ldots, n)$, the predictive TBPs are defined as the subset of the universe:

- Long-only TBP: $\{S_i \in \text{Universe} \mid \hat{r}_i \geq \theta^+\}$
- Short-only TBP: $\{S_i \in \text{Universe} \mid \hat{r}_i < \theta^-\}$
- Long–short TBP: $\{S_i \in \text{Universe} \mid \hat{r}_i \geq \theta^+ \text{and } \hat{r}_i < \theta^-\}$

To illustrate, the thresholds are used to classify assets as long and short positions. A long (short) equity portfolio consists of assets whose prediction is higher (lower) than θ^+ (θ^-). Here, the thresholds are exogenous variables, and as explained in the next sections, we can determine proper threshold values for the target portfolio through backtesting.

4.5 Portfolio weight

As classical portfolio models, the TBPs are built on the following underlying assumptions: (i) all stocks are infinitely divisible; (ii) there are no restrictions on buying and selling any selected portfolio; (iii) there is no friction (transactions costs, taxation, commissions, liquidity, etc.); and (iv) it is possible to buy and sell stocks at closing prices at any time t.

We adapt the periodic rebalancing strategy: the investor adjusts the weights in his/her portfolio at the close price on the last business day of every month, as academic research typically assumes monthly portfolio rebalancing. Throughout this study, we provide the results of experiments on the long-only TBP (TBP in short), and other TVPs can be easily built by adjusting the thresholds. Let w_i denote the TBP weight on the i^{th} asset. The TBPs ($w_i \geq 0$) are subjected to the budget constraint $\sum_i^P w_i = 1$, where P is the number of assets in the TBP. For all the TBPs, w_i is defined as $|w_i| = 1/P$, that is, equally weighted TBPs.

4.6 Simulation results

4.6.1 Experiment 1: performance of TBPs

Table 3 provides the mean and SD (standard deviation) values of the monthly returns of the individual assets, the EWP of the universe, and the TBPs with different thresholds. The explanation is as follows:

- AAPL and AMZN achieve higher returns (0.02 and 0.023) with higher volatilities (0.271 and 0.299).
- The EWP achieves lower volatility (0.039) by diversification effect, which leads higher Sharpe ratio (0.368).
- The EWP and TBPs overly outperform individual stocks in terms of the Sharpe ratios (0.368 and 0.269∼0.383).
- As shown in the TBPs, an increase in θ results in an increase in the return and volatility of TBPs.

The remarkable fact is that the EWP serves as a benchmark for evaluating the TBPs, in the sense that there is a (roughly) consistent up–right shift from the point of the EWP on the risk–return plane: as θ^+ increases, the return increases from 0.014 (EWP) to 0.015 ($\theta^+ = 0.00$), and then to 0.018 ($\theta^+ = 0.02$); the risk increases from 0.039 (EWP) to 0.04 ($\theta^+ = 0.00$), and then to 0.069 ($\theta^+ = 0.02$). This is graphically revealed in the monthly return distributions (Fig. 2) and the cumulative returns (Fig. 3) of the TBPs, which is more clearly elucidated by the risk–return frontiers in the following section (Fig. 5).

The EWP has been frequently used as a proxy for the risk–return ratio of the financial market, by both academia and the financial industry [41,42]. It is more diversified than a value-weighted portfolio, which is heavily concentrated into just the largest companies, so that it is being widely traded in the real financial industry (e.g., the NASDAQ-100 equal-weighted index allots the same weight to each stock in the

Table 3 Performance of the individual assets, EWP, and TBPs

Asset/portfolios	Threshold	Mean	SD	Mean/SD	Average assets
AAPL	–	0.02	0.075	0.271	–
AMZN	–	0.023	0.08	0.299	–
BAC	–	0.013	0.102	0.128	–
BRK	–	0.01	0.0378	0.275	–
GE	–	0.01	0.053	0.202	–
JNJ	–	0.0129	0.036	0.353	–
JPN	–	0.014	0.075	0.19	–
MSFT	–	0.017	0.062	0.275	–
T	–	0.01	0.042	0.239	–
WFC	–	0.011	0.048	0.237	–
EWP	–	0.014	0.039	0.368	–
TBP	0	0.015	0.04	0.381	9.072
	0.005	0.017	0.044	0.381	6.637
	0.01	0.02	0.052	0.383	4.376
	0.015	0.02	0.06	0.347	2.855
	0.02	0.018	0.069	0.269	2.173

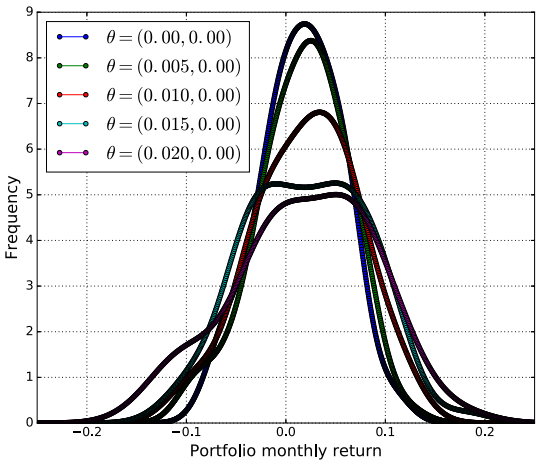

Fig. 2 (Color online) Distributions of TBP monthly returns at the different thresholds ($\theta = (\theta^+, \theta^-)$)

index). Therefore, the fact that such a well-known EWP serves as a benchmark for TVPs allows us to more quantitatively characterize TBPs.

We examine the relationship between the magnitude of predictive returns and the prediction accuracy. Table 4 shows the correct forecasts among all forecasts whose value is larger than θ^+, where the prediction accuracy ranges over 0.61–0.63, independent of θ^+.

We calculate the accumulated returns of TVPs by using the closing prices on the last trading day of each month. We rebalance all portfolios on the last trading day of each month, based on the 1-month-ahead prediction; we then reinvest according to

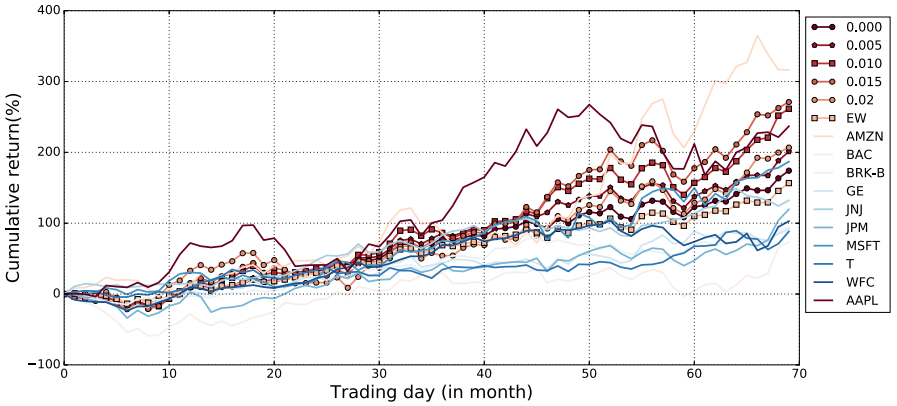

Fig. 3 (Color online) Experiment 1 results: cumulative returns of individual assets, EWP, and TBPs

Table 4 Experiment 1 results: prediction accuracy values for assets whose predictive return is higher than θ^+ over the test period

θ^+	No. of correct forecasts	No. of total forecasts	Accuracy
0	343	562	0.61
0.0025	321	521	0.616
0.005	225	405	0.629
0.0075	204	326	0.625
0.01	171	272	0.628
0.0125	134	216	0.62
0.015	110	177	0.621
0.0175	95	152	0.625
0.02	83	134	0.619
0.0225	74	117	0.632
0.025	67	108	0.62

a weight vector that divides the accumulated wealth equally among the constituents. The accumulated return R_t is defined as:

$$R_t = \prod_{i=0}^{t}(1 + r_i), \tag{11}$$

where r_i is the arithmetic return at time i. This is a standard performance measure for comparing investments, and it relates the wealth at time t, W_t, to the initial wealth, W_0, as $W_t = W_0 \times R_t$. All experiments in this study used an initial wealth value of $W_0 = 1$. Figure 4 shows the cumulative returns of the individual assets, EWP, and TBPs.

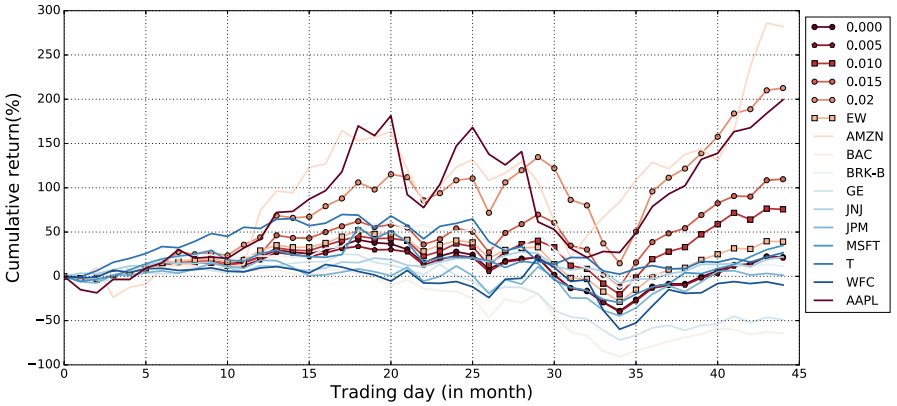

Fig. 4 (Color online) Experiment 2 results: cumulative returns of individual assets, EW, and TBPs

Table 5 Experiment 2 results: prediction accuracies for the assets whose predictive return is higher than θ over the test period

θ^+	No. of correct forecasts	No. of total forecasts	Accuracy
0	142	268	0.529
0.0025	124	238	0.521
0.005	115	218	0.527
0.0075	104	192	0.541
0.01	96	172	0.558
0.0125	79	143	0.552
0.015	73	130	0.561
0.0175	63	108	0.583
0.02	57	95	0.6
0.0225	52	85	0.611
0.025	46	73	0.605

4.7 Experiment 2: robustness test

As a further check, we conduct a similar experiment with the different period (i.e., January 1, 2006 to December 31, 2014). Figure 4 graphically shows the cumulative returns over the test period (i.e., 30% of the period). Over the test period, the market is more volatile than the previous one, and the LSTM-based predictors shows a poor predictive accuracy value of 0.495. Much of our analysis generated results similar to those in the experiment 1, but interestingly, there is a positive relation between the magnitude of predictive return and the prediction accuracy: that is, the accuracy consistently increases from 0.529 ($\theta^+ = 0.00$) to 0.611 at ($\theta^+ = 0.225$), as shown in Table 5.

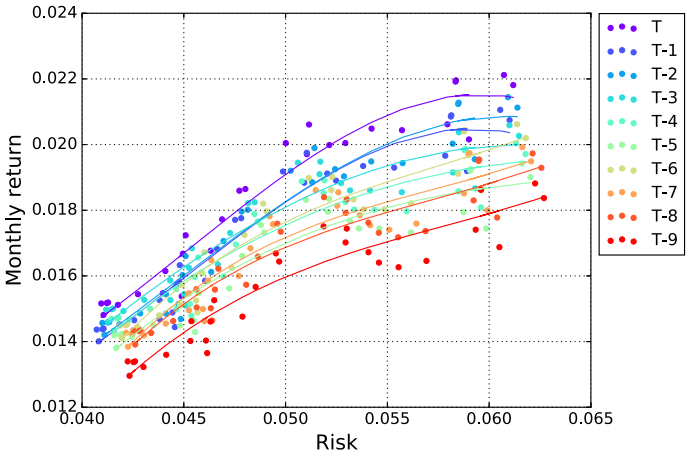

Fig. 5 (Color online) The realized monthly return versus risk of the TBPs that are constructed using 1-month-ahead return predictions at $\theta^+ = 0.000, 0.0025, \ldots, 0.025$ on the last 10 months of the test period of experiment 1. T is the last business month

5 Applications

Regarding the practical use of TBPs, we provide illustrations on how to manage them over multiple-periods and how to incorporate them into an MPT optimization portfolio.

5.1 TBP management

Figure 5 is a scatter plot of the risk–return profiles of the TBPs that we are constructed in experiment 1, at different θ^+s $(0.000, 0.0025, \ldots, 0.025)$ over the last 10 months, along with the lines fitted to the polynomial of degree 3. We will refer to the line as the "TBP frontier." Note that the points indicate the realized monthly returns and risk of the (predictive) TBPs that are built using 1-month-ahead predictive returns. The TBP lines are concave, moving upward and to the right as θ^+ increases, thus indicating that the greater the amount of risk by the increase of θ^+, the greater the realized returns. This characteristics allow for the design of a TBP with a target risk–return.

To illustrate, let us suppose that an investor at time $T - 9$ hopes to build a TBP featuring a target risk–return at time $T - 8$. If the target is the monthly risk of 0.05 and the monthly return of 0.016, the investor can estimate the $\hat{\theta}^+$, which corresponds to the target from the TBP frontier drawn at $T - 9$; the investor can then build a TBP with the target, using the predictive return generated and the estimated $\hat{\theta}^+$ at time $T - 9$. Then, at time $T - 8$, the investor will obtain an approximate return of 0.017 and risk of 0.05, as seen in the TBP frontier moving upward over the period from time $T - 9$ to $T - 8$. This difference, $0.017 - 0.015$, is the estimation risk of the TBP. As seen in the continual shift of TBPs as time passes, to maintain a target risk–return, the corresponding $\hat{\theta}^+$ needs to be updated. The estimation risk of TBP can be quantified by calculating the average of the differences between forecasts and their realizations for return and risk over a past period. There is the estimation risk,

but it is sufficiently small to classify TBPs as having different risk aptitude. The TBP frontier can be broader, combined with riskless assets. In summary, we illustrate the TBP management process:

step 1 Set a investment universe (stocks, bonds, ETFs, etc.)
step 2 Build forecasting models for future stock return or price
step 3 Select a trading position (long-only, short-only, or long–short) and a weighting method (equal-weighted, prediction-weighed, etc.)
step 4 (Backtest) Draw the TBP frontiers at different thresholds
step 5 Select a target risk–return value and find its corresponding $\hat{\theta}$ on the TBP frontier
step 6 (Actual investment) Invest in the TBP with the $\hat{\theta}$
step 7 (Realization) Estimate the TBP at $\hat{\theta}$, and reinvest in the TBP with the updated $\hat{\theta}$ from the realized TBP frontier

5.2 Mean–variance portfolio

MPT is a mathematically elegant framework for building a portfolio with specific risk-return level. However, it is well known that it is more difficult to estimate the means than the covariances of asset returns [43], and errors in the estimates of means will have a greater impact on portfolio weights than errors in the estimates of covariances. Furthermore, as mean–variance optimization is extremely sensitive to expected returns, any errors therein might make outcomes far from optimal [44,45]. For this reason, although theoretical and empirical academic studies have examined various MPT aspects, its real-life practical applications have mostly focused on minimum variance portfolios. This estimation error invariably leads to inefficient portfolios, which can be explained by considering the following three sets of portfolios [46].

- True efficient frontier (TEF): the efficient frontier based on true (but unknown parameters)
- Estimated frontier (EF): the frontier based on the estimated (and hence incorrect) parameters
- Actual frontier (AF): the frontier comprising the true portfolio mean and variance points corresponding to portfolios on the estimated frontier

The use of thresholds on predicted returns can help mitigate inefficiency by screening a subset to be predicted more accurately, as shown in Table 5. Figure 6 shows a schematic scenario that EF_T, which is estimated for the assets screened by a threshold, is located more closely to the TEF. (Here, for simplicity, we ignore the shift in EF due to the change in the asset number.)

6 Conclusion

This study proposes a novel framework by which to construct portfolios that target specific risk–return levels. We evaluated the RNN networks while examining the hit ratios of the 1-month-ahead forecasts of stock returns, and then constructed TBPs by imposing thresholds for the potential return. The TBP are more data-driven in building

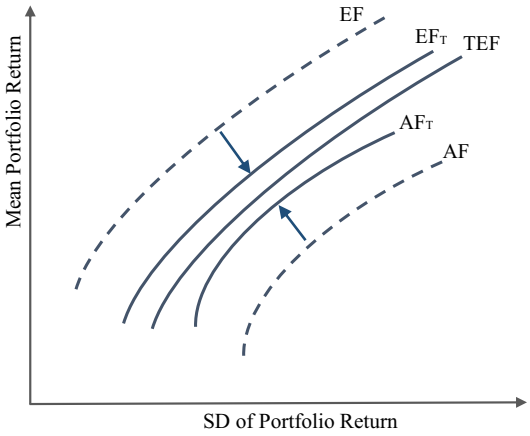

Fig. 6 (Color online) Schematic scenario for shifting EF to EF_T by screening the universe at a threshold

a portfolio than in existing methods, in the sense that they are constructed purely on the basis of forecasts that are generated by a deep learning technique, in the absence of any financial mathematics or knowledge. We showed that the EWP of the universe plays the role of the reference portfolio to TBPs and thus serves to quantitatively characterize the TBP. The TBP frontiers show that the threshold is a parameter which can control trade-off between the risk and the return of portfolios. Furthermore, we discussed how to practically manage TBPs to maintain a target risk–return over multiple-periods; we also discussed the benefit of incorporating TBPs into MPT.

The TBP is promising, since it provides a simple and straightforward way to build portfolios with target risk–returns, using predictions alone. Any prediction model can be basically applied to construct TBPs. As predictors become more accurate, TBPs can achieve greater returns, given a certain level of risk. In this respect, we believe that the TBP is a valuable application of machine learning to modern-day investment practice.

Acknowledgements This work was supported by the ICT R & D program of MSIP/IITP (2017-0-00302, Development of Self Evolutionary AI Investing Technology).

References

1. Atsalakis GS, Valavanis KP (2009) Surveying stock market forecasting techniques—Part II: soft computing methods. Expert Syst Appl 36(3):5932–5941
2. Dixon M, Klabjan D, Bang JH (2015) Implementing deep neural networks for financial market prediction on the Intel Xeon Phi. https://ssrn.com/abstract=2627258
3. Huck N (2009) Pairs selection and outranking: an application to the S&P100 index. J Oper Res 196(2):819–825
4. Huck N (2010) Pairs trading and outranking: the multi-step-ahead forecasting case. J Oper Res 207(3):1702–1716
5. Krauss C, Do XA, Huck N (2017) Deep neural networks, gradient-boosted trees, random forests: statistical arbitrage on the S&P500. Eur J Oper Res 259(2):689–702
6. Moritz B, Zimmermann T (2014) Deep conditional portfolio sorts: the relation between past and future stock returns. Working paper

7. Sermpinis G, Theofilatos KA, Karathanasopoulos AS, Georgopoulos EF, Dunis CL (2013) Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization. Eur J Oper Res 225(3):528–540
8. Takeuchi L, Lee Y-Y (2013) Applying deep learning to enhance momentum trading strategies in stocks. Working paper
9. Cavalcante RC, Brasileiro RC, Souza VLF, Nbrega JP, Oliveira ALI (2016) Computational intelligence and financial markets: a survey and future directions. Expert Syst Appl 55:194–211
10. Aggarwal S, Aggarwal S (2017) Deep investment in financial markets using deep learning models. Int J Comput Appl 162(2):40–43
11. Gao T, Li X, Chai Y, Tang Y (2016) Deep learning with stock indicators and two-dimensional principal component analysis for closing price prediction system. In: 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016. IEEE
12. Zhang Y (2015) Using financial reports to predict stock market trends with machine learning techniques. Oxford University, Oxford
13. Fischer T, Krauss C (2017) Deep learning with long short-term memory networks for financial market predictions. FAU discussion papers in economics 11/2017. Erlangen. http://hdl.handle.net/10419/157808
14. Pang X, Zhou Y, Wang P (2018) An innovative neural network approach for stock market prediction. J Supercomput. https://doi.org/10.1007/s11227-017-2228-y
15. Hu Z, Liu W, Bian J, Liu X, Liu T-Y (2018) Listening to chaotic whispers: a deep learning framework for news-oriented stock trend prediction. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp 261–269
16. Singh R, Srivastava S (2017) Stock prediction using deep learning, multimedia tools and application. Multimed Tools Appl 76(18):18569–18584
17. Markowitz H (1952) Portfolio selection. J Finance 17(1):77–91
18. Fama E, French K (1992) The cross-section of expected stock returns. Expert Syst Appl 47:427–465
19. Black F, Litterman R (1992) Global portfolio optimization. Financ Anal J 48(5):28–43
20. Michaud R (1998) Efficient asset management: a practical guide to stock portfolio optimization and asset allocation. Harvard Business School Press, Boston
21. Haugen R, Baker N (1991) Pairs selection and outranking: an application to the S&P100 index. J Oper Res 17(3):35–40
22. Choueifaty Y, Coignard Y (2008) Toward maximum diversification. J Portf Manag 35(1):40–51
23. Qian E (2005) Risk parity portfolios: efficient portfolios through true diversification. Research paper. https://www.panagora.com/assets/PanAgora-Risk-Parity-Portfolios-Efficient-Portfolios-Through-True-Diversification.pdf
24. Qian E (2005) Risk parity portfolios: the next generation. Research paper. https://www.panagora.com/assets/PanAgora-Risk-Parity-The-Next-Generation.pdf
25. Faber MT (2014) A quantitative approach to tactical asset allocation. J Wealth Manag 9(4):69–79
26. Keller WJ, Keuning JW (2014) Momentum, Markowitz, and Smart Beta: a tactical, analytical and practical look at modern portfolio theory. https://ssrn.com/abstract=2759734 or https://doi.org/10.2139/ssrn.2759734
27. Keller WJ, Bulter A (2014) A century of generalized momentum; from flexible asset allocations (FAA) to elastic asset allocation (EAA). https://ssrn.com/abstract=2543979 or https://doi.org/10.2139/ssrn.2543979
28. Keller WJ, Keuning JW (2009) Protective asset allocation (PAA): a simple momentum-based alternative for term deposits. Expert Syst Appl 36(3):5932–5941
29. Campbell JY, Sanford JG, Jiang W (1993) Trading volume and serial correlation in stock returns. Q J Econ 108:905–939
30. Choueifaty Y, Coignard Y (2000) Trading volume and cross-autocorrelations in stock returns. J Finance 55:913–935
31. Bao W, Yue J, Rao Y (2017) A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS ONE 12(7):e0180944. https://doi.org/10.1371/journal.pone.0180944
32. Freitas FD, De Souza AF, De Almeida AR (2009) Prediction-based portfolio optimization model using neural networks. Neurocomputing 72:2155–2170
33. Mishra SK, Panda G, Majhi B (2016) Prediction based mean-variance model for constrained portfolio assets selection using multiobjective evolutionary algorithms. Swarm Evol Comput 28:117–130

34. Ganeshapillai G, Guttag J, Lo A (2013) Learning connections in financial time series. In: Proceedings of the 30th International Conference on Machine Learning (ICML13), pp 109–117
35. Elman J (1990) Finding structure in time. Cogn Sci 14:179–211
36. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 5(2):157–166
37. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
38. Cho K, Merrienboer B, Gülçehre C, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR. arXiv:1406.1078
39. Diederik PK, Jimmy B (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
40. Chollet F (2016) Keras: deep learning library for theano and tensorflow. https://keras.io
41. Jegadeesh N, Titman S (2001) Profitability of momentum strategies: an evaluation of alternative explanations. J Finance 56:699–720
42. Plyakha Y, Uppal R, Vilkov G (2012) Why does an equal-weighted portfolio outperform value- and price-weighted portfolios? https://ssrn.com/abstract=2724535 or https://doi.org/10.2139/ssrn.2724535
43. Merton RC (1980) On estimating the expected return on the market: an explanatory investigation. J Financ Econ 8:323–361
44. Jorion P (1985) International portfolio diversification with estimation risk. J Bus 58(3):259–278
45. Best M, Grauer R (1992) Positively weighted minimum-variance portfolios and the structure of asset expected returns. J Financ Quant Anal 27(4):513–537
46. Broadie M (1993) Computing efficient frontiers using estimated parameters. Ann Oper Res 45:21–58