Intelligent Decision Technologies 1–14
IOS Press

Forecasting stock price index movement using a constrained deep neural network training algorithm

I.E. Livieris [a,*], T. Kotsilieris [b] S. Stavroyiannis [c] P. Pintelas [a]
[a] *Department of Mathematics, University of Patras, GR 265-00, Greece.*
[b] *Department of Business Administration, University of Peloponesse, GR 241-00, Greece*
[c] *Department of Accounting & Finance, University of Peloponesse, GR 241-00, Greece*

Abstract. The prediction of stock index movement is considered a rather significant objective in the financial world, since a reasonably accurate prediction has the possibility of gaining profit in stock exchange, yielding high financial benefits and hedging against market risks. Undoubtedly, the area of financial analysis has been dramatically changed from a rather qualitative science to a more quantitative science which is also based on knowledge extraction from databases. During the last years, deep learning constitutes a significant prediction tool in analyzing and exploiting the knowledge acquired from financial data. In this paper, we propose a new Deep Neural Network (DNN) prediction model for forecasting stock exchange index movement. The proposed DNN is characterized by the application of conditions on the weights in the form of box-constraints, during the training process. The motivation for placing these constraints is focused on defining the weights in the trained network in more uniform way, by restricting them from taking large values in order for all inputs and neurons of the DNN to be efficiently exploited and explored. The training of the new DNN model is performed by a Weight-Constrained Deep Neural Network (WCDNN) training algorithm which exploits the numerical efficiency and very low memory requirements of the L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) matrices together with a gradient-projection strategy for handling the bounds on the weights of the network. The performance evaluation carried out on three popular stock exchange indices, demonstrates the classification efficiency of the proposed algorithm.

Keywords: Deep learning, neural networks, constrained optimization, L-BFGS, financial data, stock index movement.

1. Introduction

Stock markets constitute a fundamental component of financial markets and play a significant role in the countries' economies. They facilitate international trade, aggregate and convoy information about companies funds and enable economic growth. The stock market index is one of the main tools used by investors and financial managers to describe the market and compare the returns on specific investments. Forecasting stock index value and/or movement is considered essential for portfolio management and banks' investments, constituting a key element for better decision-making, in the increasing financial market volatility and internationalized capital flows. More specifically, the accurate prediction of the trends of the stock index can assist investors to acquire opportunities for gaining profit in stock exchange, minimizing investment risk and maximizing return. The traditional prediction approach is based on historical numerical data such as the previous trend, trading volume, earnings surprise and some other numerical information. Notice that efficient market hypothesis states that "*stock prices are informationally efficient*" which means that it is possible to predict stock prices based on trading data [10, 17, 18].

The vigorous advances in digital technologies as well as the significant storage capabilities of electronic

*Corresponding author. E-mail: livieris@gmail.com

media have ultimately led to the accumulation and maintenance of large repositories of data from daily activity of different resources by the stock markets. Therefore, researchers and financial stockholders are under extreme pressure to analyze them for extracting useful knowledge in order to support policy decisions. However, financial data has enormous noise and complex dimensionality due to the volatile and chaotic dynamics of the markets and the many non-decidable, nonstationary stochastic factors involved. As a result, the process of leveraging and analyzing these data constitutes an attractive and challenging task for many financial experts and investors which often require huge efforts [4, 13].

During the last decades, several techniques have been applied and developed to predict stock trends, while most sophisticated analysis employs machine learning algorithms in order to gain a more meticulous view into the stock index. Artificial Neural Networks (ANNs) constitute a machine learning technique which has been widely utilized for predicting stock values and stock index movement [2, 12, 15, 16]. Due to their excellent capability of self-adopting and self-learning, they have been successfully applied to exploit financial data while in most cases they were found to be more accurate than other prediction models [7, 19, 22] Nevertheless, recent studies have shown that ANNs exhibit inconsistent and unpredictable behavior on stock market data [23]. The major drawback of ANNs is that they are characterized by shallow architectures since they are mostly constituted by one hidden layer. Furthermore, another drawback is the poor performance of the training algorithm since it may be trapped into a local optimal solution during the training process, therefore over-fitting. Deep learning has been proposed as a new direction for addressing these drawbacks and it has been successful applied in many real-world application areas (see [6, 8, 21] and the references there in). Deep learning networks are characterized by deeper architectures with many hidden layers, producing even better performance than shallow models to financial time series data [1,7,16,17,25]. To this end, during the last decade, the developments and advances of deep learning in decision-making have gained popularity, addressing many issues in banking and finance.

Recently, there has been proposed a new approach for improving the generalization ability of ANNs. More specifically, Livieris [9] applied additional conditions on the weights in the form of bound-constraints, during the training process. Thus, re-formulating the problem of training a neural network to a constrained optimization problem, that is

$$\min\{E(w) : w \in \mathscr{B}\}, \tag{1}$$

with

$$\mathscr{B} = \{w \in \mathbb{R}^n : l \leq w \leq u\}, \tag{2}$$

where $E(w)$ is the error function which depends on the connection weights w of the network and the vectors $l, u \in \mathbb{R}^n$ denote the lower and upper bounds on the weights, respectively. Furthermore, in order to evaluate the efficacy and the efficiency of this approach, a weight constrained training algorithm was proposed, presenting some interesting and promising results. The rationale behind this approach aims in defining the weights of the trained network in a more uniform way in order to explore and exploit all inputs and neurons of the network.

Motivated by the previous works, we propose a deep learning neural network prediction model with three hidden layers for forecasting stock exchange index movement. The classification efficiency of the proposed model is based on a new weight-constrained deep neural network training algorithm which adopts the advantages of the approach presented in [9]. The performance evaluation carried out on three of the most popular stock exchange indices, the Dow Jones Industrial Average (DJIA) index, the National Association of Securities Dealers Automated Quotations (NASDAQ) index and the Standard & Poor's (S&P) 500 index, demonstrates the classification efficiency of the proposed algorithm. The aim of this work focuses on demonstrating the superiority of constrained neural networks in stock market data analysis.

The remainder of this paper is organized as follows: Section 2 presents a survey of recent studies concerning the application of deep learning in stock market analysis. Section 3 presents a detailed description of the proposed weight constrained deep learning algorithm. Section 4 presents a brief description of the data collection and data preparation utilized in our study. Section 5 presents our experimental results, utilizing the performance profiles Dolan and Morè [3]. Finally, Section 6 presents the concluding remarks and our proposals for future research.

2. Related work

Research on the predictability of stock markets has a long history in financial economics; while opinions differ on the efficiency of markets, many widely accepted empirical studies show that financial markets are to some extent predictable. Recently, there has been a resurgence of interest in deep learning which has been effectively applied to financial problems. A number of rewarding studies have been carried out in recent years and some useful outcomes are briefly presented below.

Patel et al. [17] aimed on addressing the problem of predicting direction of movement of stock and stock price index for Indian stock markets. Their approaches for the input data consists of two methodologies in which the first involves computation of ten technical parameters using stock trading data while the second focuses on representing these technical parameters as trend deterministic data. Furthermore, they investigated four prediction models: artificial neural networks, support vector machines, random forests and Naive-Bayes. Their extensive numerical experimental revealed that for the first approach of input data random forests presented the best overall classification efficiency and the performance of models improved when the technical parameters were represented as trend deterministic data.

Tsantekidis et al. [25] proposed a deep learning methodology, based on recurrent neural networks, for predicting future mid-price movements from large-scale high-frequency limit order data. Their proposed prediction model consists of a Long Short-Term Memory (LSTM) neural network which was evaluated on a large-scale dataset of limit order book events. The presented experimental results illustrated that it significantly outperforms other prediction models such as support vector machines and artificial neural networks.

Singh et al. [23] presented a novel hybrid methodology utilizing 2-Directional 2-Dimensional Principal Component Analysis for dimensionality reduction together with deep learning for stock data forecasting. The data utilized in their study were collected from NASDAQ concerning 2843 working days of the Google stock multimedia (chart). Moreover, the authors presented some encouraging experimental results, revealing that their proposed methodology outperforms state-of-the-art prediction models.

Chong et al. [1] presented a systematic analysis of the use of deep learning networks for stock market analysis. Additionally, they described in detail a comprehensive and objective assessment of both the advantages and drawbacks of deep learning algorithms for stock market analysis and prediction. Based on their numerical experiments the authors presented the superiority of deep learning networks and also suggested promising extensions and directions of further investigation.

In more recent works, Pang et al. [16] focused on achieving better stock market predictions by developing an innovative neural network approach. More specifically, they illustrated the concept of "stock vector", based on the development of word vector in deep learning, which implies that the input is the multi-stock high-dimensional historical data instead of a single index or single stock index. Additionally, they proposed a deep LSTM neural network with one embedded layer and a LSTM neural network with automatic encoder to predict the stock market. In both models the embedded layer and the automatic encoder were utilized to vectorize the data, in a bid to forecast the stock via LSTM neural network. The presented accuracy of these two prediction models for the Shanghai A-shares composite index was 57.2% and 56.9%, respectively; while for individual stocks was 52.4% and 52.5%, respectively.

Fisher et al. [5] evaluated the performance of LSTM neural networks to financial time series prediction tasks. More specifically, their research was focused on predicting out-of-sample directional movements for the constituent stocks of the S&P 500 from 1992 until 2015. They compared the performance of the LSTM network against a random forest, a standard deep net, as well as a simple logistic regression. Their extensive numerical experiments revealed that the LSTM networks were the most efficient and accurate machine learning technique. Based on the common patterns of the LSTM portfolio, they also devised a simplified rules-based trading strategy. Additionally, the authors stated that their proposed methodology was able to effectively extract meaningful information from noisy financial time series data.

In this work, we propose a new weight-constrained DNN model with three hidden layers for forecasting stock exchange index movement which constitutes the main novelty of this work. The model is characterized by the application of conditions on the weights in the form of box-constraints, during the training process. The motivation for placing constraints on the values of weights is to considerably reduce the likelihood that some weights will "blow up" to unrealistic values by restricting them from taking large values. The training of the new model is performed by a new Weight-

Algorithm 1: Weight-Constrained Deep Neural Network (WCDNN)

Step 1: Initiate w_0 and vectors l and u.

Step 2: Set $k = 0$.

Step 3: **repeat**

Step 4: Calculate the error function value E_k and its gradient ∇E_k at w_k.

Step 5: Set the quadratic model (3) at w_k.

Step 6: Calculate the generalized Cauchy point w^C. (*Stage I*)

Step 7: Define the active set $\mathscr{A}(w^C)$.

Step 8: Minimize the quadratic model (3) with respect to the non-active variables (*Stage II*)

$$\overline{w}_{k+1} = \arg\min_{w \in D_S} m_k(w)$$

where $D_S = \left\{ w \in \mathbb{R} \,|\, l_i \leq w_i \leq u_i, \forall i \notin \mathscr{A}(w^C) \right\}$.

Step 9: Set $d_k = \overline{w}_{k+1} - w_k$. (*Stage III*)

Step 10: Compute the learning rate η_k satisfying the strong Wolfe line search conditions (4) and (5).

Step 11: Update the weights $w_{k+1} = w_k + \eta_k d_k$ and set $k = k+1$.

Step 12: **until** (stopping criterion).

Constrained Deep Neural Network (WCDNN) training algorithm which utilized a gradient-projection strategy for handling the bounds on the weights of the network together with the computational efficiency of the L-BFGS matrices.

3. Weight constrained deep neural network training algorithm

In this section, we present the proposed Weight-Constrained Deep Neural Network (WCDNN) training algorithm for the prediction of stock index movement. For completeness, a high level description of the proposed algorithm is presented in Algorithm 1.

At each iteration, the proposed algorithm WCDNN calculates the error function value E_k and its gradient ∇E_k at w_k (Step 4). Then, WCDNN approximates the error function $E(w)$ by a quadratic model $m_k(w)$ at a point w_k, namely

$$m_k(w) = E_k + g_k^T(w - w_k) + \frac{1}{2}(w - w_k)^T B_k (w - w_k) \quad (3)$$

where $E_k = E(w_k)$, g_k is the gradient of the error function at w_k and B_k is a positive-definite L-BFGS Hessian approximation [11] (Step 5).

In the sequel, the algorithm performs a minimization procedure of the approximation model $m_k(w)$ to compute the new vector of weights, which consists of three stages: the generalized Cauchy point, the subspace minimisation; and the line search.

Stage I: Cauchy point computation. The quadratic model (3) is approximately minimized subject to the feasible domain $\mathscr{B}$ utilizing the gradient projection method in order to compute the generalized Cauchy point w^C (Step 6). Eventually, the active set $\mathscr{A}(w^C)$ is calculated which consists of the indices of weights whose values at w^C are at lower or upper bound; thus these weights are held fixed (Step 7).

Stage II: Subspace minimization. After the active set of variables is obtained, then the model (3) is approximately minimized with respect to the non-active variables utilizing a direct primal method [11] (Step 8). It is worth mentioning that the minimization is performed in the subspace composed by non-active variables, allowing savings in computational time, especially when dealing with large and deep neural networks.

Stage III: Line search. The new vector of weights w_{k+1} is computed by performing a line search along the search direction d_k (Step 9) which satisfies the strong Wolfe line search conditions

$$E_{k+1} \leq E_k + c_1 \eta_k \nabla E_k^T d_k, \quad (4)$$

$$|\nabla E_{k+1}^T d_k| \leq c_2 |\nabla E_k^T d_k|. \quad (5)$$

with $0 \leq c_1 \leq c_2 < 1$ (Step 10).

Index	Training set				Testing test			
	Decrease	%	Increase	%	Decrease	%	Increase	%
DJIA10	916	45.73%	1087	54.27%	114	47.50%	126	52.50%
DJIA20	910	45.66%	1083	54.34%	110	47.62%	121	52.38%
DJIA30	905	45.64%	1078	54.36%	116	50.22%	115	49.78%
NASDAQ10	880	43.93%	1123	56.07%	114	47.50%	126	52.50%
NASDAQ20	874	43.85%	1119	56.15%	110	47.62%	121	52.38%
NASDAQ30	871	43.92%	1112	56.08%	104	47.06%	117	52.94%
S&P10	912	45.53%	1091	54.47%	117	48.75%	123	51.25%
S&P20	906	45.46%	1087	54.54%	114	49.35%	117	50.65%
S&P30	901	45.44%	1082	54.56%	108	48.87%	113	51.13%

Table 1

The number of up and down movements of DJIA index, NASDAQ index and S&P 500 index

4. Datasets

For the purpose of this study, we have used data from 2264 trading days from January 4, 2009 to December 31, 2018, including daily indicators from the daily price of the DJIA index, NASDAQ index and S&P 500 index[1]. The data were divided in training set and testing set. The training set consists of data from January 4, 2009 to December 31, 2017 (9 years) which ensures a substantial amount of data for training whereby 9 years of index movements should cover a wide range of long term and short term trends. The testing set comprises data from January 4, 2018 to December 31, 2018 which ensures that we forecast our evaluation metrics on unseen out-of-sample data."

Each record of includes daily information about the stock market. Table 2 presents a set of the five specific attributes utilized in our study. The first two attributes concern the price of the index at the beginning and at the end of the day. The following two attributes represent the highest and the lowest price of the index during the day. The last attribute concerns the number of shares of stock (in thousands) that traded hands in each day.

Summarizing, nine datasets have been created, three for each financial index, for a window size of 10, 20 and 30. *Window size* is the number of days for which the data is being taken into account for predicting the next day's data. For example, window size 10 means data is being taken for 10 days and results are predicted for 11th day.

[1]All data were obtained from `http://finance.yahoo.com` website.

Attribute	Values
Opening price	Numeric
Closing price	Numeric
High price	Numeric
Low price	Numeric
Volume (in thousands)	Numeric

Table 2

Attribute description

- The datasets DJIA10, DJIA20 and DJIA30 contain the attributes concerning indicators from 10, 20 and 30 days of the DJIA index, respectively.
- The datasets NASDAQ10, NASDAQ20 and NASDAQ30 contain the attributes concerning indicators from 10, 20 and 30 days of the NASDAQ index, respectively.
- The datasets S&P10, S&P20 and S&P30 contain the attributes concerning indicators from 10, 20 and 30 days of the S&P 500 index, respectively.

Thus, we utilized various window sizes to test which sizes gives the better result. In our numerical experiments, we included windows size of 10, 20 and 30, which provide us better classification performance.

5. Experimental results

In this section, we conduct a performance evaluation of the proposed WCDNN algorithm against the state-of-art training algorithms Resilient backpropagation [20] and Stochastic Gradient Descent with momentum [24].

Table 3 presents the number of features (#Features), the deep network architectures (with 3 hidden layers) and the total number of weights for each window size.

Window size	#Features	Neural network architecture	Total number of weights
10	50	50-100-50-20-2	11212
20	100	100-120-60-15-2	20327
30	150	150-200-75-20-2	46837

Table 3

Brief description of datasets and neural network architectures used in our study

All neural networks utilized logistic sigmoid activation functions, received the same sequence of input patterns and the initial weights were initiated using the Nguyen-Widrow method [14]. The implementation code was written in Matlab 7.6 and the Resilient backpropagation and the Stochastic Gradient Descent with momentum were utilized with their default optimized parameter settings [20, 24]. The simulations have been carried out on a PC (2.66GHz Quad-Core processor, 4GB RAM) running Linux operating system while the results have been averaged over 100 simulations.

The performance of the training algorithms was evaluated utilizing the following two performance metrics: F_1*-score* and *accuracy*. It is worth noticing that F_1-score consists of a harmonic mean of precision and recall while accuracy is the ratio of correct predictions of a classification model [9].

Furthermore, since a small number of simulations tend to dominate these results, the cumulative total for a performance metric over all simulations does not seem to be too informative. Therefore, similar to [9], we also utilized the performance profiles of Dolan and Morè [3] relative to both performance metrics, to present perhaps the most complete information in terms of robustness, efficiency and solution quality. The use of performance profiles eliminates the influence of a small number of simulations on the benchmarking process and the sensitivity of results associated with the ranking of solvers [3]. The performance profile plots the fraction P of simulations for which any given algorithm is within a factor τ of the best training algorithm.

The curves in the following figures have the following meaning:

- "RPROP" stands for Resilient backpropagation.
- "SGD" stands for Stochastic Gradient Descent with momentum.
- "WCDNN_1" stands for Algorithm 1 with bounds on the weights $-1 \leq w_i \leq 1$.
- "WCDNN_2" stands for Algorithm 1 with bounds on the weights $-2 \leq w_i \leq 2$.

5.1. Performance evaluation on DJIA index

Tables 4 and 5 present the performance of the training algorithms RPROP, SGD, WCDNN_1 and WCDNN_2 for DJIA index, regarding F_1-score and accuracy, respectively. Notice that the highest classification accuracy is highlighted in bold for each window size and performance metric. It is worth mentioning that WCDNN_1 and WCDNN_2 outperformed the classical training algorithms, relative to all window sizes. In more detail, WCDNN_2 reported the highest F_1-score for windows size 10 and 20, followed by WCDNN_1; while for window size 30, WCDNN_1 exhibited the best F_1-score. Additionally, WCDNN_1 presented 54.04%, 56%, 53.64% average classification accuracy for window size 10, 20 and 30, respectively, exhibiting the highest performance.

Algorithm	DJIA10	DJIA20	DJIA30
RPROP	62.14%	61.12%	57.51%
SGD	61.86%	61.79%	59.06%
WCDNN_1	63.33%	63.03%	**60.50%**
WCDNN_2	**64.07%**	**63.10%**	60.36%

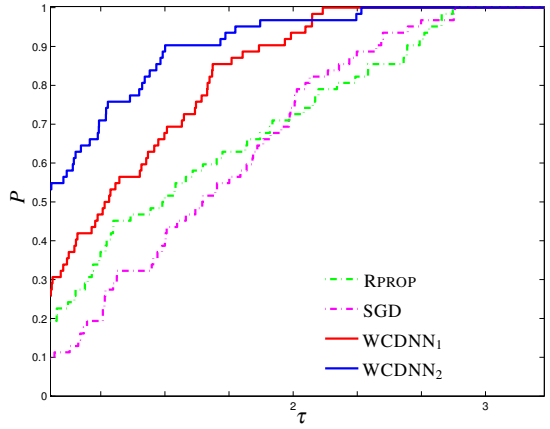
Table 4

Performance evaluation of training algorithms for DJIA index, relative to F_1-score

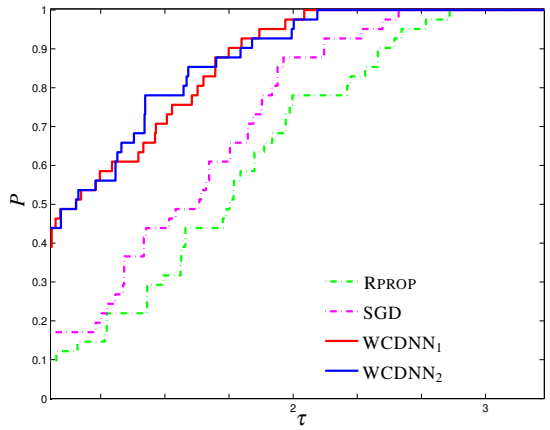
Algorithm	DJIA10	DJIA20	DJIA30
RPROP	52.28%	54.17%	51.03%
SGD	52.13%	54.43%	52.29%
WCDNN_1	**54.04%**	**56.00%**	**53.64%**
WCDNN_2	53.32%	55.43%	53.36%

Table 5

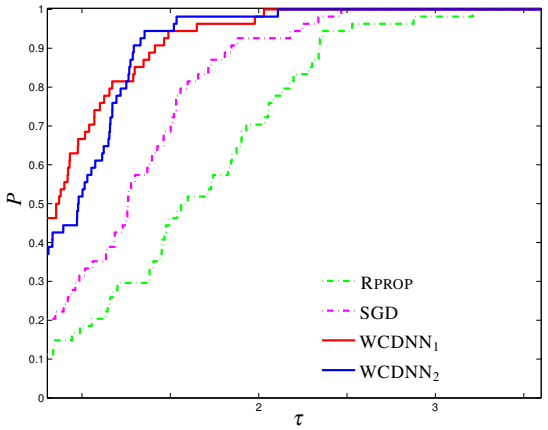
Performance evaluation of training algorithms for DJIA index, relative to accuracy



(a) Window size = 10

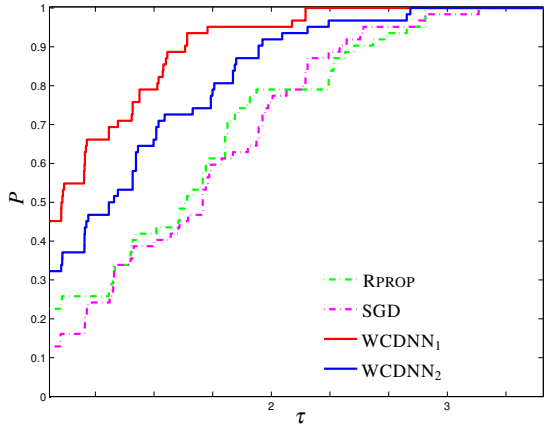


(b) Window size = 20

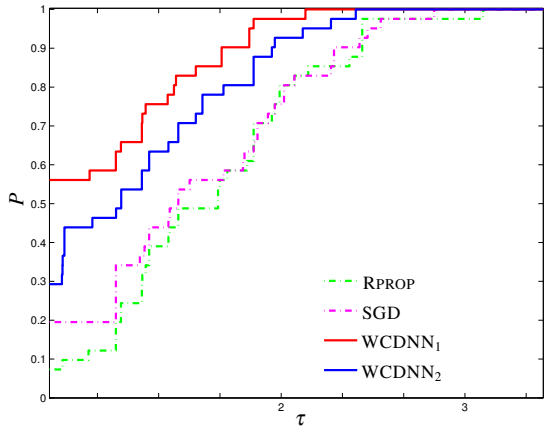


(c) Window size = 30

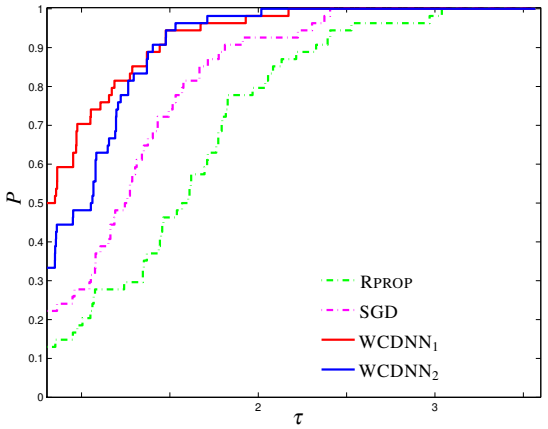
Fig. 1. Log_{10} scaled performance profiles for DJIA index based on F_1-score



(a) Window size = 10



(b) Window size = 20



(c) Window size = 30

Fig. 2. Log_{10} scaled performance profiles for DJIA index based on accuracy

Figures 1 and 2 present the performance profiles for DJIA index of all deep learning algorithms, based on F_1-score and accuracy, respectively. Clearly, the proposed algorithm WCDNN$_1$ presented the highest probability of being the optimal solver, regarding all window sizes, followed by WCDNN$_2$. More specifically, regarding the F_1-score, WCDNN$_2$ exhibited the best performance for window size 10; while for window size 20 exhibited almost identical performance with WCDNN$_1$. Furthermore, WCDNN$_1$ reported 45%, 56% and 50% of simulations with the highest classification accuracy, for windows size 10, 20 and 30, respectively; while WCDNN$_2$ reported 32%, 29% and 33% in the same situations. Based on the above discussion, we concluded that the bounds on the weights, substantially led to the development of trained deep neural networks with improved classification accuracy. Furthermore, the tighter the bounds get, the higher the chance for good classification generalisation.

5.2. *Performance evaluation on NASDAQ index*

Tables 6 and 7 present the average performance of each training algorithm for NASDAQ index, relative to all windows sizes. As mentioned above, the accuracy measure of the best performing algorithm is highlighted in bold. Regarding F_1-score, WCDNN$_1$ presented the best performance for window size 10 and 30 (66.41% and 61.76%, respectively) while WCDNN$_2$ presented the best performance for window size 20 (62.08%). Relative to the classification accuracy, the aggregated results show that WCDNN$_1$ is the most effective algorithm for all windows sizes. Finally, it is worth mentioning that all training algorithms exhibited the best performance for window size 10.

Algorithm	NASDAQ10	NASDAQ20	NASDAQ30
RPROP	64.32%	60.90%	59.42%
SGD	63.98%	59.82%	59.43%
WCDNN$_1$	**66.41%**	61.67%	**61.76%**
WCDNN$_2$	65.92%	**62.08%**	61.67%

Table 6

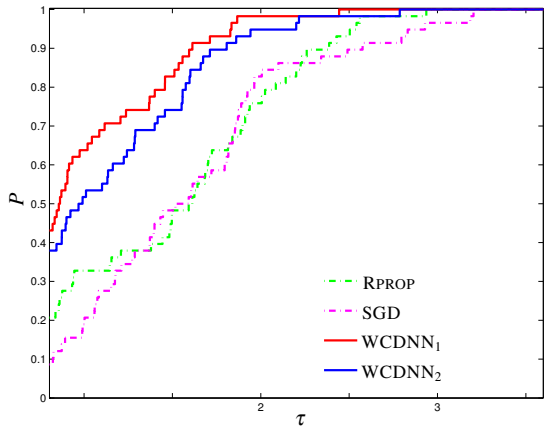
Performance evaluation of training algorithms for NASDAQ index, relative to F_1-score

Algorithm	NASDAQ10	NASDAQ20	NASDAQ30
RPROP	53.99%	52.06%	51.59%
SGD	54.89%	52.12%	52.48%
WCDNN$_1$	**57.10%**	**54.08%**	**54.48%**
WCDNN$_2$	56.18%	53.94%	54.35%

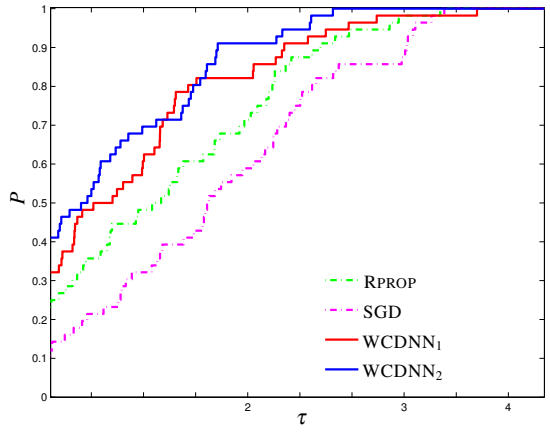
Table 7

Performance evaluation of training algorithms for NASDAQ index, relative to accuracy

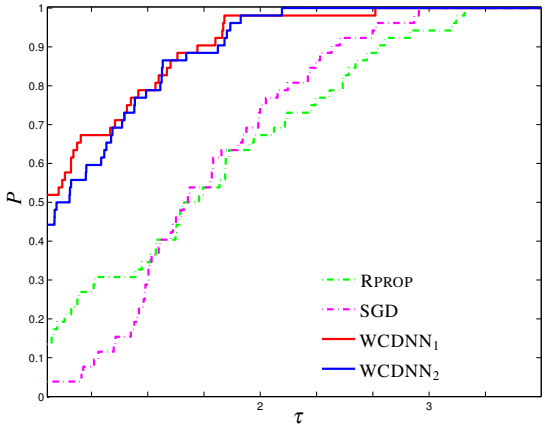
Figures 3 and 4 illustrate the performance profiles for NASDAQ index, F_1-score and accuracy, investigating the classification efficiency of each deep learning algorithm. Firstly, it is worth noticing that WCDNN$_1$ and WCDNN$_2$ outperformed the classical training algorithms, regarding both performance metrics. As regards the F_1-score metric, WCDNN$_1$ illustrated the best performance for window size 10 and 30; while WCDNN$_2$ presented the best performance for window size 20. Relative to accuracy, WCDNN$_1$ and WCDNN$_2$ reported 52% and 24% of simulations with the highest classification accuracy for windows size 10, respectively while RPROP and SGD reported 16% and 10%, respectively. For windows size 20, WCDNN$_1$ and WCDNN$_2$ presented 38% and 42% of simulations with the highest classification accuracy, respectively while RPROP and SGD presented 16% and 14%, respectively. For windows size 30, WCDNN$_1$ and WCDNN$_2$ reported 52% and 44% of simulations with the highest classification accuracy, respectively while RPROP and SGD reported 10% and 6%, respectively. Therefore, we conclude the tighter the bounds get, the higher the classification accuracy.



(a) Window size = 10

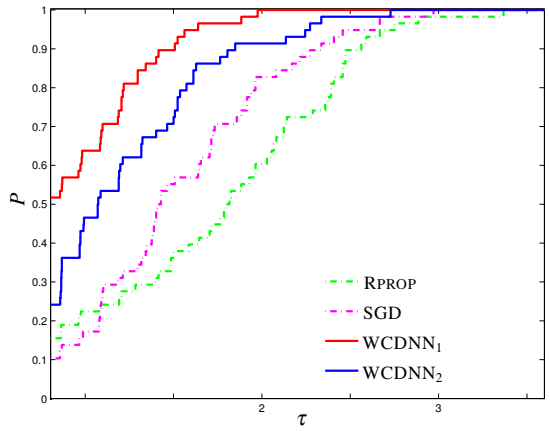


(b) Window size = 20

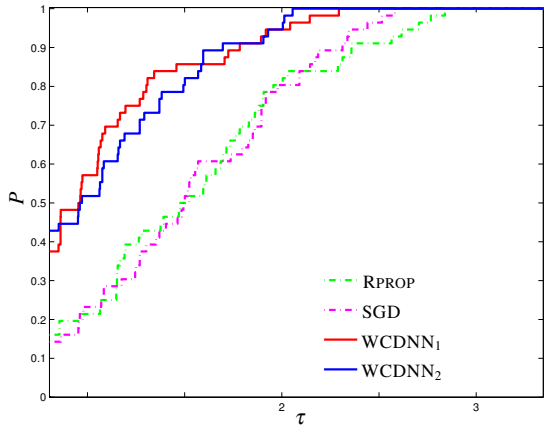


(c) Window size = 30

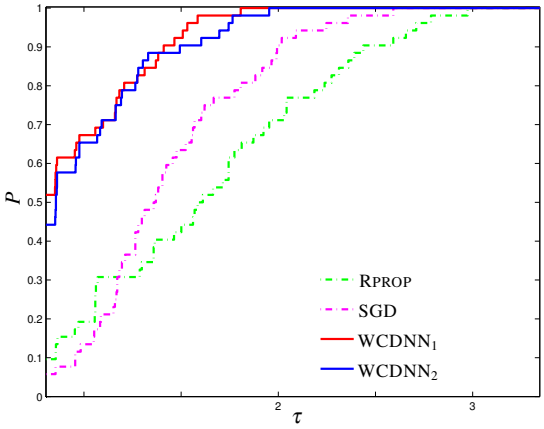
Fig. 3. Log_{10} scaled performance profiles for NASDAQ index based on F_1-score



(a) Window size = 10



(b) Window size = 20



(c) Window size = 30

Fig. 4. Log_{10} scaled performance profiles for NASDAQ index based on accuracy

5.3. Performance evaluation on S&P 500 index

Tables 8 and 9 present the average performance of each training algorithm for S&P 500 index, relative to F_1-score and accuracy, respectively. As regards the performance the proposed algorithm, WCDNN_1 and WCDNN_2 outperformed the classical training algorithms, regarding all window sizes and performance metrics. Furthermore, for window size 10, WCDNN_2 reported the highest F_1-score; while WCDNN_1 presents the best classification accuracy. For window size 20 and 30, the best performance was presented by WCDNN_1 and WCDNN_2, respectively, relative to both performance metrics. Thus, we conclude that in contrast to the previous benchmarks, in case the bounds are too tight, this substantially did not benefit much the classification performance of the networks.

Finally, it is worth mentioning that all algorithms exhibited the best performance for window size 10.

Algorithm	S&P10	S&P20	S&P30
RPROP	63.05%	58.98%	57.83%
SGD	61.58%	58.77%	58.15%
$WCDNN_1$	63.56%	**60.26%**	59.35%
$WCDNN_2$	**64.07%**	59.84%	**59.77%**

Table 8

Performance evaluation of training algorithms for S&P 500 index, relative to F_1-score

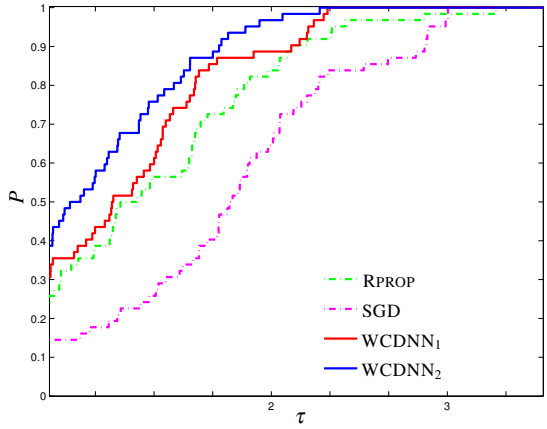
Algorithm	S&P10	S&P20	S&P30
RPROP	52.46%	52.20%	52.05%
SGD	52.82%	52.03%	52.42%
$WCDNN_1$	**54.63%**	**54.06%**	53.69%
$WCDNN_2$	54.13%	52.94%	**53.83%**

Table 9

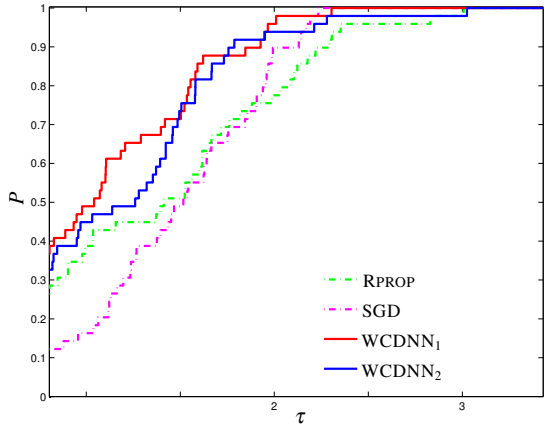
Performance evaluation of training algorithms for S&P 500 index, relative to accuracy

Figure 5 presents the performance profiles for S&P 500 index based on the F_1-score metric. Both versions of the proposed algorithm illustrated the highest probability of being the optimal training algorithms. In more detail, $WCDNN_2$ exhibited the best performance, reporting 37% and 52% of simulations with the best F_1-score for window size 10 and 30, respectively; followed by $WCDNN_1$ which reported 30% and 38% in the same situations. For window size 20, $WCDNN_1$ and $WCDNN_2$, exhibited almost identical performance, presenting 34% and 32% of simulations with the best F_1-score, respectively while RPROP and SGD presented 26% and 12%, respectively.

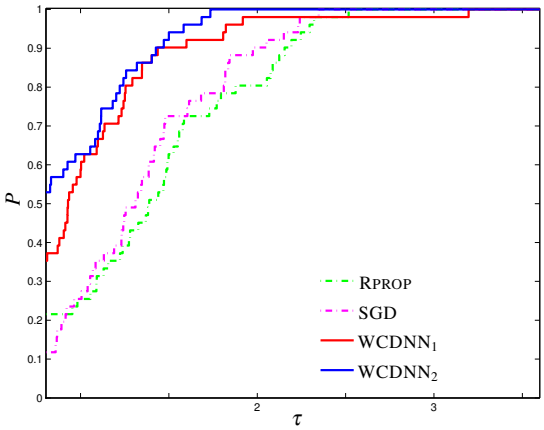
Figure 6 presents the performance profiles for S&P 500 index based on the accuracy metric investigating the efficiency of each training algorithm. Similar conclusions can be made with the previous analysis. Both $WCDNN_1$ and $WCDNN_2$ exhibited the best performance, regarding all window sizes. More specifically, $WCDNN_1$ reported 40%, 45% and 38% of simulations with the highest classification accuracy for windows size 10, 20 and 30, respectively; while $WCDNN_2$ reported 37%, 29% and 41% in the same situations. As regards the classical training algorithms, RPROP presented 13%, 24% and 21% of simulations with the best accuracy for windows size 10, 20 and 30, respectively; while SGD presented 18%, 8% and 10% in the same situations.



(a) Window size = 10

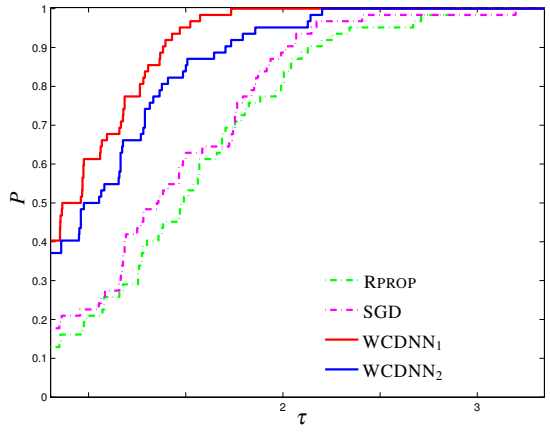


(b) Window size = 20

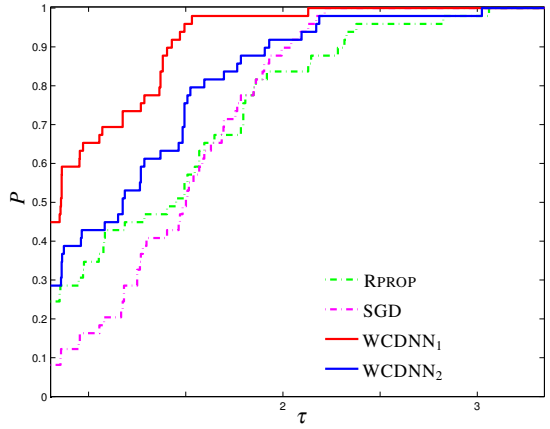


(c) Window size = 30

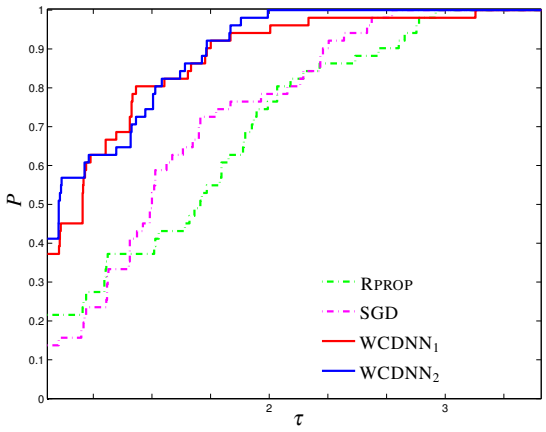
Fig. 5. Log_{10} scaled performance profiles for S&P 500 index based on F_1-score



(a) Window size = 10



(b) Window size = 20



(c) Window size = 30

Fig. 6. Log_{10} scaled performance profiles for S&P 500 index based on accuracy

Summarizing, the interpretation of Tables 4-9 and Figures 1-6 demonstrate that WCDNN performed better than the classical training algorithms Rprop and SGD. This better performance can be attributed to the application of box-constraints on the weights, which inhibits weights to take up unrealistic (large) values. Therefore, the preliminary numerical experiments provide empirical evidence that the proposed training algorithm trains deep neural networks with better classification performance on average.

Regarding the selection of the bounds on the weights, we observed that WCDNN presented the best average performance utilizing the bounds $[-1,1]$ in five datasets i.e. DJIA20, DJIA30, NASDAQ10, NASDAQ30, S&P20. In contrast, WCDNN using the bounds $[-2,2]$, exhibited the best average performance in two datasets i.e. DJIA10, NASDAQ20 and S&P30; while for datasets NASDAQ20 and S&P10 WCDNN reported similar performance, regarding both selected weight bounds. Therefore, we are able to conclude that the tighter the bounds get, the higher the chance for good generalization performance in most cases; nevertheless, this is not a general case.

6. Conclusions and future research

In this work, we proposed a new weight-constrained DNN model with three hidden layers for forecasting stock exchange index movement. The proposed model is characterized by the application of box-constraints on its weights during the training process. The rationale for placing additional constraints on the values of weights is to define the weights of the trained network in a more uniform way in order to explore and exploit all inputs and neurons of the deep neural network. In other words, our approach aims in considerably reducing the likelihood that some weights will "blow up" to unrealistic values by restricting them from taking large values. The training of the new model is performed by a new training algorithm, called WCDNN, which exploits the computational efficiency and very low memory requirements of the L-BFGS matrices together with a gradient-projection strategy for handling the bounds on the weights of the network. Our numerical experiments illustrated the classification efficiency of the proposed algorithm, as confirmed statistically by the performance profiles. Therefore, we are able to conclude that the proposed algorithm, appears to efficiently train deep neural networks with improved clas-

sification ability in domains such as forecasting stock index movement.

In our future work, we intent to incorporate our proposed methodology to more advanced and complex neural network architectures such as recurrent and LSTM neural networks, together with regularization techniques. Furthermore, another interesting direction for future research could be the evaluation the proposed methodology versus relevant frameworks addressing the forecasting stock exchange index movement. Finally, since our experimental results are quite encouraging, a next step could be the application of our proposed framework for predicting the value stock price indices and prices.

References

[1] Eunsuk Chong, Chulwoo Han, and Frank C Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017.

[2] Thuy-An Dinh and Yung-Keun Kwon. An empirical study on importance of modeling parameters and trading volume-based features in daily stock trading using neural networks. In *Informatics*, volume 5, page 36, 2018.

[3] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[4] Jie Du and Roy Rada. Dilemmas in knowledge-based evolutionary computation for financial investing. *Intelligent Decision Technologies*, 7(2):123–136, 2013.

[5] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[6] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[7] Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, 38(5):5311–5319, 2011.

[8] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[9] Ioannis E. Livieris. Improving the classification efficiency of an ANN utilizing a new training methodology. *Informatics*, 6(1), 2018.

[10] Burton G Malkiel and Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

[11] José Luis Morales and Jorge Nocedal. Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Transactions on Mathematical Software (TOMS)*, 38(1):7, 2011.

[12] Binoy B Nair and VP Mohandas. An intelligent recommender system for stock trading. *Intelligent Decision Technologies*, 9(3):243–269, 2015.

[13] Keng-Hoong Ng and Kok-Chin Khor. Stockprof: a stock profiling framework using data mining approaches. *Information Systems and e-Business Management*, 15(1):139–158, 2017.

[14] Derrick Nguyen and Bernard Widrow. Improving the learning speed of 2-layer neural network by choosing initial values of adaptive weights. *Biological Cybernetics*, 59:71–113, 1990.

[15] Trilok Nath Pandey, Alok Kumar Jagadev, Satchidananda Dehuri, and Sung-Bae Cho. A review and empirical analysis of neural networks based exchange rate prediction. *Intelligent Decision Technologies*, (Preprint):1–17, 2019.

[16] Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, and Victor Chang. An innovative neural network approach for stock market prediction. *The Journal of Supercomputing*, pages 1–21, 2018.

[17] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259–268, 2015.

[18] Roy Rada and Lidan Ha. Intelligent technologies for investing: a review of engineering literature. *Intelligent Decision Technologies*, 2(3):167–177, 2008.

[19] Bernardete Ribeiro and Ning Chen. Financial credit risk assessment via learning-based hashing. *Intelligent Decision Technologies*, 11(2):177–186, 2017.

[20] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE, 1993.

[21] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[22] Alaa F Sheta, Sara Elsir M Ahmed, and Hossam Faris. A comparison between regression, artificial neural networks and support vector machines for predicting stock market index. *Soft Computing*, 7(8), 2015.

[23] Ritika Singh and Shashi Srivastava. Stock prediction using deep learning. *Multimedia Tools and Applications*, 76(18):18569–18584, 2017.

[24] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[25] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning to detect price change indications in financial markets. In *Signal Processing Conference (EUSIPCO), 2017 25th European*, pages 2511–2515. IEEE, 2017.