




# Learning fused lasso parameters in portfolio selection via neural networks

Stefania Corsaro<sup>1</sup> · Valentina De Simone<sup>2</sup> · Zelda Marino<sup>1</sup> · Salvatore Scognamiglio<sup>1</sup> 

Accepted: 9 February 2024  
© The Author(s) 2024

## Abstract

In recent years, fused lasso models are becoming popular in several fields, such as computer vision, classification and finance. In portfolio selection, they can be used to penalize active positions and portfolio turnover. Despite efficient algorithms and software for solving non-smooth optimization problems have been developed, the amount of regularization to apply is a critical issue, especially if we have to achieve a financial aim. We propose a data-driven approach for learning the regularization parameters in a fused lasso formulation of the multi-period portfolio selection problem, able to realize a given financial target. We design a neural network architecture based on recurrent networks for learning the functional dependence between the regularization parameters and the input data. In particular, the Long Short-Term Memory networks are considered for their ability to process sequential data, such as the time series of the asset returns. Numerical experiments performed on market data show the effectiveness of our approach.

**Keywords** Regularization parameters · Fused lasso · Portfolio selection · Neural network · Long short-term memory

---

✉ Zelda Marino  
zelda.marino@uniparthenope.it

Stefania Corsaro  
stefania.corsaro@uniparthenope.it

Valentina De Simone  
valentina.desimone@unicampania.it

Salvatore Scognamiglio  
salvatore.scognamiglio@uniparthenope.it

<sup>1</sup> Department of Management and Quantitative Studies, Parthenope University of Naples, Via Generale Parisi 13, 80132 Napoli, Italy

<sup>2</sup> Department of Mathematics and Physics, University of Campania “Luigi Vanvitelli”, Viale Lincoln 5, 81100 Caserta, Italy

## 1 Introduction

Numerous real-life problems can be recast into the fused lasso form:

$$\begin{aligned} \min_x & f(x) + \tau_1 \|x\|_1 + \tau_2 \|Lx\|_1 \\ \text{s.t.} & Ax = b, \end{aligned} \quad (1)$$

where  $f : \mathbb{R}^n \mapsto \mathbb{R}$  is a twice continuously differentiable convex function,  $L \in \mathbb{R}^{l \times n}$ ,  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$ ,  $m \leq n$ , and  $\tau_1, \tau_2 > 0$ . The lasso term  $\|x\|_1$  and the fusion term  $\|Lx\|_1$  induce sparsity in the vector  $x$  and in some dictionary  $Lx$ , respectively. Among the application areas, there are image processing, classification and finance (see De Simone et al. 2022 and the references therein). The non-smoothness of the  $l_1$ -type regularization terms needs different specialized variants of first and second-order numerical methods. First-order methods based on Bregman iteration (Corsaro et al. 2021b; De Simone et al. 2020; Goldstein and Osher 2009; Osher et al. 2005) have proved to be efficient for the solution of this type of problem. The Bregman iterative scheme requires the solution of an unconstrained subproblem at each step, which does not need to be computed exactly. It is possible to use iterative methods suited to deal with the  $l_1$  term, such as the alternating direction method of multipliers (ADMM), that guarantees convergence, provided that the inexactness of the solution can be controlled. The success of this approach is based also on the availability in closed (and cheap) form of the proximal operator of the  $l_1$  norm using the well-known soft-thresholding operator (Corsaro et al. 2021b). Recently, specialized second-order methods have been proposed, that can offer an attractive alternative for large-scale problems (De Simone et al. 2022). In this case, a proper choice of the linear algebra solver allows one to efficiently solve the larger but smooth optimization problems coming from a standard reformulation of the original one.

Thus, very efficient methods are available to perform the numerical solution, whereas setting the regularization parameters is still a challenging task strongly related to the specific application. The most generally applicable calibration schemes for this setting are typically based on Cross-Validation (Beer et al. 2019; Dijkstra 2014) or BIC-type criteria (Lee and Chen 2020). These methods provide optimal parameter values with respect to a certain loss function; they do not allow one to assume specific properties on the solution obtained by using those parameters. Our contribution is a procedure that automatically provides the regularization parameters, in a multi-period mean-variance portfolio optimization framework. In this context, a suitable choice of  $\tau_1$  and  $\tau_2$  allows one to build optimal portfolios that satisfy a fixed financial request. We refer to a composite financial requirement, comprising a maximum number of active positions, which correspond to non-null weights in the portfolio, and a maximum number of transactions. This allows one to reduce both holding and transaction cost, which is of great significance, especially for small investors (Lajili-Jarjir and Rakotonratsimba 2008; Ding 2006; Torrente and Uberti 2023). The fused lasso approach was introduced in the context of multi-period portfolio optimization problem in Corsaro et al. (2021a, 2021b). In those papers, authors show that the formulation (1) of the portfolio selection problem, where  $f$  is a dynamic risk measure, allows one to produce optimal cost-limited strategies, if  $\tau_1$  and  $\tau_2$  are properly chosen. The authors of this work previously explored the automatic regularization parameter computation in the context of the lasso portfolio selection problem in Corsaro and De Simone (2019), Corsaro et al. (2020) and Corsaro et al. (2022). In Corsaro and De Simone (2019) the single period case was considered. In that paper an adaptive procedure for parameter setting was presented; the procedure was then extended to the multi-period framework in Corsaro et al. (2020). Despite its efficiency, this procedure generally exhibits overestimates of the optimal regularization parameter, leading to

portfolios with fewer active positions than desired. For this reason, the use of Neural Networks (NN) was discussed in Corsaro et al. (2022). In that work, the authors show the effectiveness of this approach, which allows one to obtain accurate estimates. In this paper, we investigate the use of Neural Networks for learning the regularization parameters in the context of a fused lasso formulation of the multi-period portfolio selection problem, where one aims at computing optimal medium and long-term investment strategies. In the last few decades, NN and Deep learning models are becoming very popular in economics and finance since their ability of processing high-dimensional data and modeling complex phenomena (Slavici et al. 2016; Wang 2009).

In this paper, we design a Recurrent Neural Network (RNN) model for computing the regularization parameters. This class of NN are specifically designed to analyze and extract patterns from data with a sequential structure. In particular, modern RNN architectures, such as the Long Short-Term Memory (LSTM) networks, are a promising tool for learning the complex dependence structure between the regularization parameters and the time series of the asset returns. The paper is organized as follows: in Sect. 2 we present the reference model for portfolio selection and discuss its numerical solution; in Sect. 3 we briefly introduce neural networks, with special regard to recurrent ones; in Sect. 5 we show the results of tests that validate the approach; finally, we give some conclusions and outline future work.

## 2 Mathematical model

We consider either a medium or long-term investment, where the investor has the opportunity to exit before the term. We define our model in a multi-period setting, that is, the investment period is partitioned into sub-periods, delimited by the rebalancing dates, at which decisions are taken. The optimal portfolio is defined by the vector

$$w = [w_1^T, \dots, w_m^T]^T \in \mathbb{R}^N,$$

where  $m$  is the number of rebalancing dates,  $n$  is the number of assets and  $N = m \cdot n$ . The dynamic risk measure is an additive one; this kind of measures arise when the risk of losses is estimated separately in different periods, and then the time contributions are aggregated (Chen et al. 2017). It is given by the following quadratic function:

$$J(w) = \sum_{i=1}^m w_i^T C_i w_i$$

where  $C_i$  is the covariance matrix estimated at the beginning of the  $i$ -th period. The optimal portfolio is the solution to the following non-smooth optimization problem:

$$\begin{aligned} \min_w \quad & J(w) + \tau_1 \|w\|_1 + \tau_2 \|Lw\|_1 \\ \text{s.t.} \quad & Aw = b_{\text{eq}} \\ & Gw \geq b_{\text{ineq}}. \end{aligned} \tag{2}$$

Problem (2) is a fused lasso one. The lasso term  $\|u\|_1$  allows one to obtain small portfolios, that is, a small number of active positions, thus reducing the holding cost.  $L$  is a first-order finite-difference operator, so the fusion term  $\|Lu\|_1$  controls the transaction cost, acting on the portfolio turnover. The penalizing effect of each regularization term increases with respect to the related parameter; moreover, the interaction between the two terms must be considered. Thus, the choice of  $\tau_1$  and  $\tau_2$  is a key issue.

Both equality and inequality linear constraints define the feasible set. Equality constraints establish the budget constraint and the self-financing property. Inequality constraints state the minimum expected wealth at all the rebalancing dates to prevent severe loss in the case of an early exit and at the end of the investment period.

## 2.1 Numerical solution

We consider the alternating split Bregman algorithm used in Ma et al. (2021), based on a further reformulation of problem (2) in terms of equality constraints only:

$$\begin{aligned} \min_{(\mathbf{w}, \mathbf{s}, \mathbf{d}, \mathbf{z})} \quad & \frac{1}{2} J(\mathbf{w}) + \tau_1 \|\mathbf{z}\|_1 + \tau_2 \|\mathbf{d}\|_1 + \mathcal{I}_D(\mathbf{s}) \\ \text{s.t.} \quad & A\mathbf{w} = \mathbf{b} \\ & G\mathbf{w} - \mathbf{s} = \mathbf{w}_{\min} \\ & L\mathbf{w} = \mathbf{d} \\ & \mathbf{w} = \mathbf{z}. \end{aligned}$$

where  $\mathcal{I}_D(\mathbf{s})$  the indicator function of the slack variable  $\mathbf{s}$  on  $D = \{\mathbf{s} \in \mathfrak{R}^m : \mathbf{s} \geq \mathbf{0}\}$ . Alternating split Bregman splits the minimization process into four parts. At each iteration, closed-form solutions can be obtained for the minimization with respect to  $\mathbf{s}$ ,  $\mathbf{d}$  and  $\mathbf{z}$ . Minimization with respect to  $\mathbf{d}$  and  $\mathbf{z}$  can be efficiently done using the soft operator, defined as:

$$\mathcal{S}(\mathbf{x}, \gamma) = \frac{\mathbf{x}}{|\mathbf{x}|} \max(|\mathbf{x} - \gamma|, \mathbf{0}),$$

where the proximal mapping of the indicator function on a given set is the orthogonal projection operator onto the same set. Regarding the quadratic minimization with respect to  $\mathbf{w}$ , we note that at each step  $k$ , the optimal value can be obtained by solving the linear system

$$H\mathbf{w} = \mathbf{rhs}_k,$$

where only  $\mathbf{rhs}_k$  depends on the current iteration. The matrix  $H$  is symmetric positive definite, sparse, and banded; its sparse Cholesky factorization can be compute once, and two triangular systems are solved at each iteration. The method is outlined in Algorithm 1.

**Algorithm 1** Split Bregman for Portfolio Selection

---

```

1:  $\lambda_1, \lambda_2, \lambda_3, \lambda_4 > 0;$ 
2:  $\mathbf{w}_0 = \mathbf{0}, \mathbf{s}_0 = \mathbf{0}, \mathbf{d}_0 = \mathbf{0}, \mathbf{z}_0 = \mathbf{0}$ 
3:  $\mathbf{b}_0^{\mathbf{w}} = \mathbf{0}, \mathbf{b}_0^{\mathbf{s}} = \mathbf{0}, \mathbf{b}_0^{\mathbf{d}} = \mathbf{0}, \mathbf{b}_0^{\mathbf{z}} = \mathbf{0}$ 
4:  $H = \text{diag}(C_1, \dots, C_m) + \lambda_1 A^T A + \lambda_2 G^T G + \lambda_3 L^T L + \lambda_4 I$ 
5: Compute the Cholesky factorization of  $H$ 
6: for  $k = 0, 1, 2, \dots$  do
7:    $\mathbf{rhs}_k = \lambda_1 A^T \mathbf{b}_k^{\mathbf{w}} + \lambda_2 G^T (\mathbf{s} - \mathbf{b}_k^{\mathbf{s}}) + \lambda_3 L^T (\mathbf{d} - \mathbf{b}_k^{\mathbf{d}}) + \lambda_4 (\mathbf{z} - \mathbf{b}_k^{\mathbf{z}})$ 
8:    $\mathbf{w}_{k+1} = H^{-1} \mathbf{rhs}_k$ 
9:    $\mathbf{s}_{k+1} = \max(\mathbf{0}, G \mathbf{w}_{k+1} + \mathbf{b}_k^{\mathbf{s}})$ 
10:   $\mathbf{d}_{k+1} = \mathcal{S}(L \mathbf{w}_{k+1} + \mathbf{b}_k^{\mathbf{d}}, \tau_2 / \lambda_3)$ 
11:   $\mathbf{z}_{k+1} = \mathcal{S}(\mathbf{w}_{k+1} + \mathbf{b}_k^{\mathbf{z}}, \tau_1 / \lambda_4)$ 
12:   $\mathbf{b}_{k+1}^{\mathbf{w}} = \mathbf{b}_k^{\mathbf{w}} + \lambda_1 (b - A \mathbf{w}_{k+1})$ 
13:   $\mathbf{b}_{k+1}^{\mathbf{s}} = \mathbf{b}_k^{\mathbf{s}} + \lambda_2 (G \mathbf{w}_{k+1} - \mathbf{w}_{\min} - \mathbf{s}_{k+1})$ 
14:   $\mathbf{b}_{k+1}^{\mathbf{d}} = \mathbf{b}_k^{\mathbf{d}} + \lambda_3 (L \mathbf{w}_{k+1} - \mathbf{d}_{k+1})$ 
15:   $\mathbf{b}_{k+1}^{\mathbf{z}} = \mathbf{b}_k^{\mathbf{z}} + \lambda_4 (\mathbf{w}_{k+1} - \mathbf{z}_{k+1})$ 
16: end for

```

---

### 3 Deep neural networks

Deep Learning represents a class of algorithms replicating the human brain’s learning mechanism. They consist of interconnected computational units, called neurons, arranged in multiple layers that process data and learn from it. In Hornik et al. (1989), the authors proved that a deep neural network with a linear output layer, at least one hidden layer and a suitable activation function can approximate any continuous function defined on a closed and bounded subset of  $\mathbb{R}^n$ . The structure of connections of the units defines different types of neural networks; see Goodfellow et al. (2016) for a detailed description. In the feed-forward neural networks, the information propagates among the different layers only forward.

Let  $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{q_0}$  and  $\mathbf{y} \in \mathcal{Y} \subseteq \mathbb{R}^{q_D}$  be the input and output values. A deep feed-forward neural networks with  $D$  layers and  $(q_1, q_2, \dots, q_D) \in \mathbb{N}^D$  units, can be formalised as follows:

$$\begin{aligned}
 \mathbf{z}^{(1)}(\mathbf{x}) &= \phi^{(1)}\left(\mathbf{w}_0^{(1)} + W^{(1)}\mathbf{x}\right), \\
 \mathbf{z}^{(2)}(\mathbf{x}) &= \phi^{(2)}\left(\mathbf{w}_0^{(2)} + W^{(2)}\mathbf{z}^{(1)}(\mathbf{x})\right), \\
 &\dots \\
 \mathbf{y} = \mathbf{z}^{(D)}(\mathbf{x}) &= \phi^{(D)}\left(\mathbf{w}_0^{(D)} + W^{(D)}\mathbf{z}^{(D-1)}(\mathbf{x})\right).
 \end{aligned}$$

We denote with  $W^{(k)} \in \mathbb{R}^{q_k \times q_{k-1}}$  the weight matrices,  $\mathbf{w}_0^{(k)} \in \mathbb{R}^{q_k}$  the bias vectors and  $\phi^{(k)}$  some non-linear *activation function*, for  $k = 1, \dots, D$ . Popular choices for the activation function are the rectifier linear unit (*relu*), the *sigmoid*, and the hyperbolic tangent (*tanh*) functions. The output of each layer  $\mathbf{z}^{(k)}(\mathbf{x})$  is a set of new *features* obtained as a

transformation of the input variables. Through this process, the explanatory power of the features on the response variable  $\mathbf{y}$  is progressively improved. The calibration of the network weights  $(\mathbf{w}_0^{(k)}, W^{(k)})_{1 \leq k \leq D}$  is performed through the Back-Propagation (BP) algorithm. It is an iterative process in which the network weights are progressively adjusted to minimize a specific loss function measured on a data sample  $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^I$ .

In addition to the classical connections, recurrent Neural Networks (RNN) have some additional synapses that connect neurons cyclically (Elman 1990). In this framework, the unit's output is reprocessed as input in the following time steps. In this way, predictions are formulated by considering what has been processed in the past. The recurrent nature of these networks makes them promising tools for processing sequential data.

Let  $\mathbf{x}_t \in \mathbb{R}^{q_0}$ ,  $0 < t \leq T$ , be a multivariate time-series. RNN generally contains only one layer that exploits the sequential structure of the input data. One often chooses the first layer since it directly works on the input data. In this case, the mechanism of an RNN layer can be formulated as follows:

$$\begin{aligned} \mathbf{z}_t^{(1)} &= \phi^{(1)} \left( \mathbf{w}_0^{(1)} + W^{(1)} \mathbf{x}_t + U \mathbf{z}_{t-1}^{(1)} \right), \quad t = 1, \dots, T; \\ \mathbf{z}_0^{(1)} &= \mathbf{0}, \end{aligned}$$

where  $U \in \mathbb{R}^{q \times q}$  are the weights associated to the output of the previous time-step. In this setting, the last activation of the RNN layer,  $\mathbf{z}_T^{(1)}$ , becomes the input of the second layer.

However, the calibration of the weights of the RNN generally presents vanishing gradient issues. In order to overcome these problems, more sophisticated RNN architectures were introduced. The Long Short-Term Memory (LSTM) networks, in addition to the recurrent synapses containing the short-term memory, present additional *memory cells* that store and release the long-term information through some functions called *gates*. Combining short- and long-term memories, LSTM networks appear able to learn complex dynamics (Abedin et al. 2021; Jauhar et al. 2022; Perla et al. 2021; Sun et al. 2021). A graphical representation of the LSTM cell is depicted in Figure where  $U \in \mathbb{R}^{q \times q}$  are the weights associated to the output of the previous time-step. In this setting, the last activation of the RNN layer,  $\mathbf{z}_T^{(1)}$ , becomes the input of the second layer.

However, the calibration of the weights of the RNN generally presents vanishing gradient issues. In order to overcome these problems, more sophisticated RNN architectures were introduced. The Long Short-Term Memory (LSTM) networks, in addition to the recurrent synapses containing the short-term memory, present additional *memory cells* that store and release the long-term information through some functions called *gates*. Combining short- and long-term memories, LSTM networks appear able to learn complex dynamics (Abedin et al. 2021; Jauhar et al. 2022; Perla et al. 2021; Sun et al. 2021). A graphical representation of the LSTM cell is depicted in Fig. 1.

If we denote with  $W^{(p)} \in \mathbb{R}^{q \times d}$ ,  $U^{(p)} \in \mathbb{R}^{q \times q}$  and  $\mathbf{w}_0^{(p)} \in \mathbb{R}^q$  the weights associated to each subnet  $p \in \{i, o, f, z\}$ , the mechanism of an LSTM layer can be described, for  $t = 1, \dots, T$ , as follows:

$$\mathbf{i}_t = \sigma \left( \mathbf{w}_0^{(i)} + W^{(i)} \mathbf{x}_t + U^{(i)} \mathbf{z}_{t-1} \right), \quad (3)$$

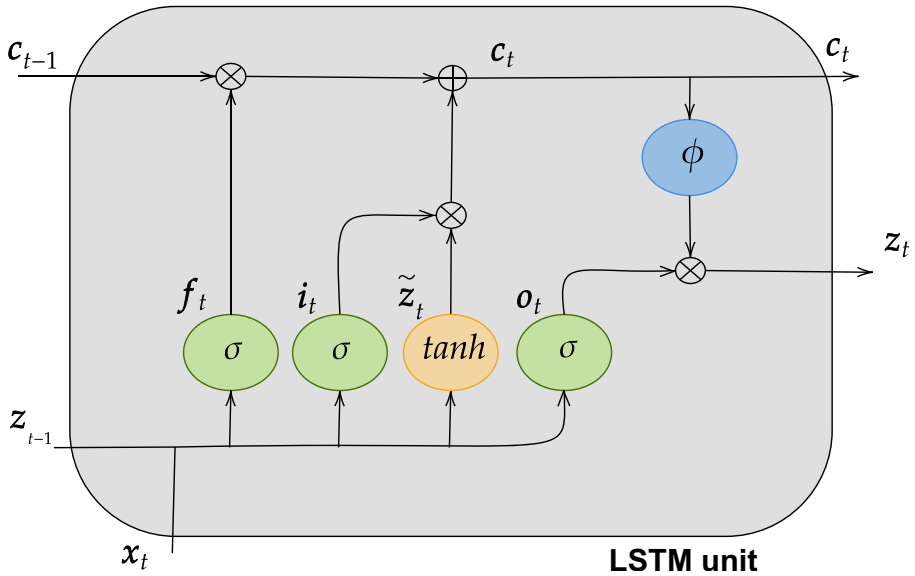


Fig. 1 A graphical representation of the LSTM cell

$$o_t = \sigma \left( w_0^{(o)} + W^{(o)}x_t + U^{(o)}z_{t-1} \right), \tag{4}$$

$$f_t = \sigma \left( w_0^{(f)} + W^{(f)}x_t + U^{(f)}z_{t-1} \right), \tag{5}$$

$$\tilde{z}_t = \tanh \left( w_0^{(z)} + W^{(z)}x_t + U^{(z)}z_{t-1} \right), \tag{6}$$

$$c_t = c_{t-1} \odot f_t + i_t \odot \tilde{z}_t, \tag{7}$$

$$\begin{aligned} z_t &= \phi(c_t) \odot o_t, \\ z_0 &= \mathbf{0}, \\ c_0 &= \mathbf{0}. \end{aligned} \tag{8}$$

where  $\sigma(\cdot) : \mathbb{R} \mapsto (0, 1)$  and  $\tanh(\cdot) : \mathbb{R} \mapsto (-1, 1)$  are respectively the sigmoid and the hyperbolic tangent activation function,  $c_t$  is the state memory cell at time  $t$ , and  $\odot$  is the Hadamard product.

Three gates regulate the mechanism of storing and releasing information. They are generally called *forget*, *input* and *output* gates. Specifically, the forget gate (eq. (5)), which has sigmoid activation, defines the percentage of information considered obsolete and must be deleted. The input gate (eq. (3)) selects new information from the input data that have to be merged with the output of the forget gate (eq. (7)). Finally, the

output value of the LSTM cell (eq. (8)) is computed by combining the state of the memory cell  $c^{(t)}$  with the output gate (eq. (6)). The number of parameters to optimize in an LSTM layer equals  $4 \times q \times (d + q + 1)$ .

## 4 A LSTM-based model for regularization parameters

In this section we describe the NN-based model. The section is split in two parts. In Sect. 4.1 we describe the network architecture; in Sect. 4.2 we discuss the network calibration.

### 4.1 Network architecture

We aim to learn the functional relationship between the regularization parameters  $\tau = (\tau_1, \tau_2) \in \mathbb{R}_+^2$  and the financial target, given in terms of percentage of portfolio sparsity and transaction costs, and the asset returns  $R = (r_t)_t \in \mathbb{R}^{n \times T}$ ,  $0 < t \leq T$ . We use two values,  $l_s \in [0, 1]$  and  $l_c \in [0, 1]$ , to define lower and/or upper bounds on sparsity and transaction costs. We formulate a regression problem where the response variable is the vector of regularization parameters  $\tau$  allowing to achieve the desired financial targets, and the regressor is the matrix past returns  $R$ .

We design a neural network consisting of one LSTM layer and one fully-connected layer. In general, deeper neural network architectures could be used for learning this function. However, this architecture realizes a good trade-off between accuracy and efficiency. First, an LSTM layer of size  $q_{LSTM}$  is used to process the multivariate time series of the returns, as described in equations (3)–(8). The output of the LSTM layer is the activation in the last time-step  $z_T$ . This vector can be interpreted as a set of features that summarises the information related to the asset return time series.

We assume that the regularization parameters depend on asset returns and on the financial target. In this framework,  $\tau$  is computed by applying a 2-dimensional fully-connected layer to the vector  $(z_T(R), l_s, l_c)$ :

$$\tau^{(NN)} = \phi^{(\tau)} \left( w_0^{(\tau)} + W^{(\tau)} z_T(R) + l_s w_s^{(\tau)} + l_c w_c^{(\tau)} \right),$$

where  $w_0^{(\tau)}, w_s^{(\tau)}, w_c^{(\tau)} \in \mathbb{R}^2$  and  $W^{(\tau)} \in \mathbb{R}^{2 \times q_{LSTM}}$  are the networks parameters. In particular,  $w_0^{(\tau)} = (w_0^{(\tau_1)}, w_0^{(\tau_2)})$  is the bias term related to the regularization parameters,  $W = (w^{(\tau_1)}, w^{(\tau_2)})$  are the coefficients associated to the features extracted by the asset returns  $z_T(R)$ ,  $w_s^{(\tau)} = (w_s^{(\tau_1)}, w_s^{(\tau_2)})$  are the coefficients associated to the required sparsity, and  $w_c^{(\tau)} = (w_c^{(\tau_1)}, w_c^{(\tau_2)})$  are the coefficients associated to the required cost rate.

### 4.2 Network calibration

The elements of the matrices and the bias vectors of the different layers of the NN architectures need to be appropriately calibrated. Denoting by  $\theta$  the vector containing all the network parameters, one could argue that the training process consists of an unconstrained optimization problem, where a suitable loss function  $\mathcal{L}(\theta)$  is chosen. The NN training is generally carried out using the Back Propagation (BP) algorithm where the updating of the weights is based on the gradient of the loss function  $\mathcal{L}(\theta)$ . The weights are iteratively adjusted to decrease the



error of the network outputs with respect to some reference values. To train the Neural Network, we collect a sample set

$$\mathcal{S} = \{(\beta_s^j, \beta_c^j, R^j, \tau_1^j, \tau_2^j), j = 1, \dots, L\}, \quad (9)$$

where  $\beta_s^j, \beta_c^j$  define the financial target,  $R^j$  is the time series of  $n$  asset returns. The couple  $(\tau_1^j, \tau_2^j) \in [\tau_{min}, \tau_{max}] \times [\tau_{min}, \tau_{max}]$  is computed using a random grid search. We define a nonuniform grid to guarantee the same number of grid points for consecutive magnitude orders of parameters. Then, we recursively sample grid points that are used to compute the optimal portfolios by means of Algorithm 1. We choose the first point that produces an optimal portfolio satisfying the financial target.

## 5 Numerical experiments

In this section, we show some results of tests that we perform on real market data. The Neural Network algorithm is applied to several portfolios, generated using the real-world price values.

### 5.1 SP 500

We start the discussion by considering a real dataset containing weekly returns of assets included in the S &P 500 index, widely regarded as the most significant index of large-cap U.S. equities. We consider the data provided in Bruni et al. (2016). Returns are obtained from daily prices obtained by Thomson Reuters Datastream; data are filtered to check and correct missing or inaccurate values. Moreover, data are adjusted for dividends and stock splits. Figure 2 shows the time series of the SP500 index for the considered period. It is interesting to note that some volatility clusters are visible since the time span covers some

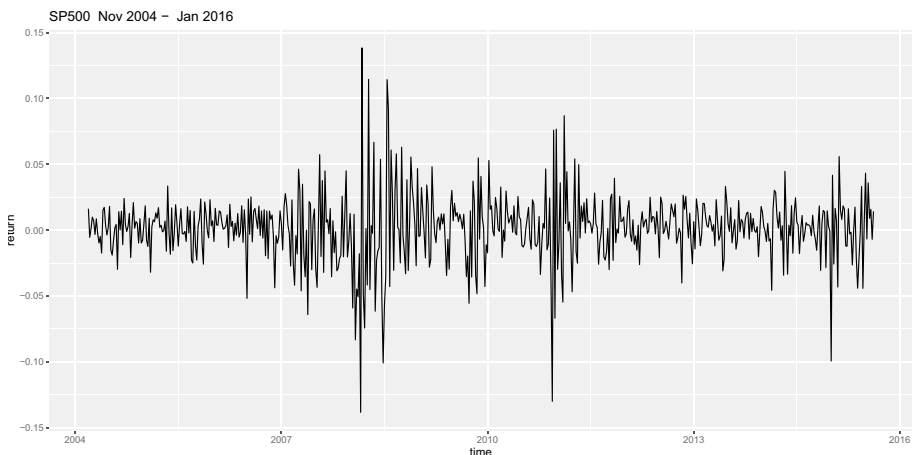


Fig. 2 Time-series of the returns related to the SP500 index

periods of market instability. The first volatility cluster matches the period of the bankruptcy of Lehman Brothers (De Haas and Van Horen 2012), which occurred on September 15, 2008, during the subprime crisis (2007–2009). The second cluster is framed into the Sovereign Debt Crisis (2010–2011), which led to an increased heterogeneity of financial markets conditions (Ehrmann and Fratzscher 2017).

We simulate 5 years (2007–2012) investment strategies, where the investor revises decisions twice a year so that we have  $m = 10$  rebalancing dates.

We compute the sparsity and the transaction costs. The sparsity is computed as follows:

$$\mathcal{SP} = \frac{N_{sparse}}{N},$$

where  $N_{sparse}$  is the number of zeros in the optimal portfolio.

To evaluate the transaction costs, we count the number of changes in the wealth associated with a fixed asset across successive rebalancing dates; we assume that each change in wealth corresponds to a transaction on the asset. The number of transactions associated with the optimal strategy is given by

$$T_{opt} = \sum_{i=1}^n \sum_{j=1}^{m-1} G_{ij},$$

where

$$G_{ij} = \begin{cases} 0 & \text{if } |(\mathbf{w}_{j+1})_i - (\mathbf{w}_j)_i| < \epsilon * \xi_{init} \\ 1 & \text{otherwise} \end{cases}$$

for  $i = 1, \dots, n$  and  $j = 1, \dots, m - 1$ . We count only variations that are significant from the financial point of view, that is, we do not consider differences below  $10^{-6} \times \xi_{init}$ , where  $\xi_{init}$  is the initial investment.

The percentage of transactions of the optimal strategy is estimated as:

$$\mathcal{T} = \frac{T_{opt}}{N},$$

where  $N$  is the number of transactions of the portfolio with full turnover.

As already said, we use  $l_s$  and  $l_c$  to define the financial target used in our experiments. We require that the sparsity and the transaction costs are bounded as follows:

$$l_s - tol_s < \mathcal{SP} < l_s + tol_s; \quad 0 < \mathcal{T} < l_c + tol_c, \quad (10)$$

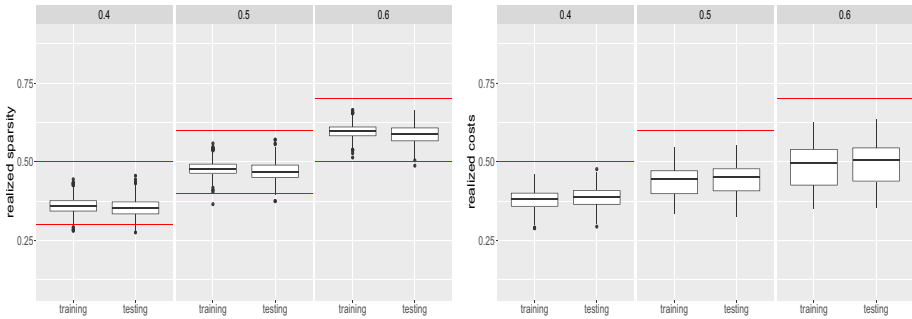
where  $tol_c$  and  $tol_s$  are acceptable levels of tolerance.

In our experiments, the sample (9) contains  $L = 9000$  elements. For  $j = 1, \dots, L$   $l_s^j$  and  $l_c^j$  vary in the set  $F = \{0.4, 0.5, 0.6\}$  and  $tol_c$  and  $tol_s$  are equal to 0.1.  $R^j$  is the return time series of  $n = 100$  assets randomly extracted from the S &P 500 basket. The couple  $(\tau_1^j, \tau_2^j)$ , that satisfies (10), is computed by means of the random grid search, setting  $\tau_{min} = 10^{-5}$ ,  $\tau_{max} = 10^{-2}$ . We discretize the square  $[10^{-5}, 10^{-2}] \times [10^{-5}, 10^{-2}]$  using 3600 points not evenly spaced. In particular, for both dimensions we consider 20 points in each one of the intervals  $[10^{-5}, 10^{-4}]$ ,  $[10^{-4}, 10^{-3}]$ ,  $[10^{-3}, 10^{-2}]$ .

We assume that the investor has one unit of wealth at the beginning of the planning horizon, that is,  $\xi_{init} = 1$ . We set  $\lambda_i = 1$ ,  $\forall i = 1, \dots, 4$  in Algorithm 1. Iterations are stopped

**Table 1** Loss function in the training and testing sets; the values are multiplied by  $10^3$

Loss	Training	Testing
$\mathcal{L}_{\tau_1}(\theta)$	0.1076	0.1402
$\mathcal{L}_{\tau_2}(\theta)$	0.0902	0.0909



**Fig. 3** box plots of the realized sparsities (left) and costs (right) produced by the NN in both the training and testing set for different values of  $l_s$  and  $l_c$

as soon as all the constraints are satisfied within constraint tolerance  $Tol = 10^{-6}$ . We consider a NN model with an LSTM layer of size  $q_{LSTM} = 2^4$ . The NN model is calibrated to minimize the Mean Absolute Error (MAE) between the network predictions and the reference values. In such a case, the training induces the minimization of the following loss function:

$$\mathcal{L}(\theta) = \mathcal{L}_{\tau_1}(\theta) + \mathcal{L}_{\tau_2}(\theta) = \sum_{l=1}^L |\tau_1 - \tau_1^{(NN)}| + \sum_{l=1}^L |\tau_2 - \tau_2^{(NN)}|.$$

It is equivalent to minimising the sum of the  $l_1$ -norm of the error related to  $\tau_1$  and  $\tau_2$ . The NN was fit for 100 epochs using the ADAM algorithm (Kingma and Ba 2014). The training is carried out considering the 75% of the total sample ( $L_{train} = 6750$ ); it represents the training set. To analyse the ability of the network to generalise to new portfolios, the remaining 25% is used as testing set ( $L_{test} = 2250$ ). We select the weight configuration that presents the lowest out-sample error. It is measured on the *validation set*, that is, a small portion of the training set which is not used for the training. In our case, the validation set size is 5% of the training sample one.

In Table 1, we present the values of the two components of the loss function in the whole training and testing set. Overall, the losses are quite low. Furthermore, the losses on the testing set are comparable to the losses on the training set. This result highlights that the NN model has successfully learnt the functional relationship between input data and regularization parameters.

In Fig. 3, we analyze the percentage of sparsity (left) and costs (right) realized by the NN on the training and testing set for different values of  $l_s$  and  $l_c$ . The red lines define the non-zero bounds in (10). In almost all cases, the regularization parameters produced by the NN allow for achieving the financial target. In particular, the target in terms of transaction costs is always satisfied, while the bounds related to the sparsity target sometimes are

**Table 2** Percentage of portfolios in which the couple  $(\tau_1, \tau_2)$  computed by the NN provides optimal portfolios that satisfy the financial target on the training and testing sets

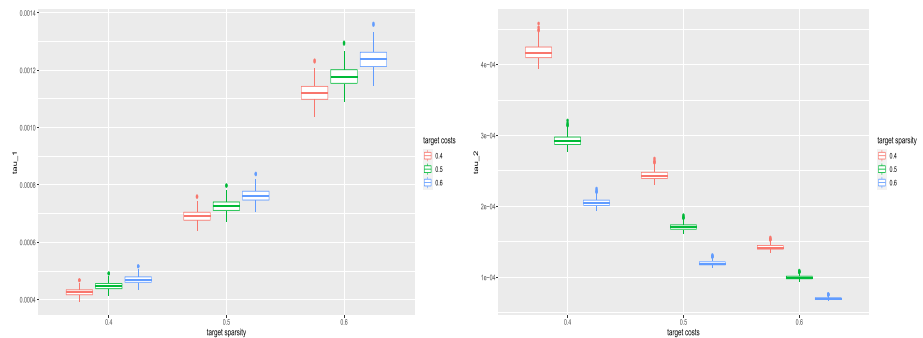
Training set	$l_c = 0.40$	$l_c = 0.50$	$l_c = 0.60$	tot
$l_s = 0.40$	99.33%	99.73%	99.87%	99.64%
$l_s = 0.50$	99.87%	100.00%	100.00%	99.96%
$l_s = 0.60$	100.00%	100.00%	100.00%	100.00%
tot	99.73%	99.91%	99.96%	99.87%
Testing set	$l_c = 0.40$	$l_c = 0.50$	$l_c = 0.60$	tot
$l_s = 0.40$	94.40%	98.40%	98.40%	97.07%
$l_s = 0.50$	99.60%	100.00%	100.00%	99.47%
$l_s = 0.60$	99.60%	100.00%	100.00%	99.87%
tot	97.73%	99.33%	99.33%	98.80%

violated. This happens especially when low sparsity is required. However, the maximum violation is about  $10^{-2}$ .

Table 2 reports the percentage of cases in which the parameters produced by the NN model allow to satisfy (10) on training and testing sets, varying  $l_c$  and  $l_s$  jointly. The overall success rate decreases marginally on the testing (98.80%) with respect to the training set (99.87%). Furthermore, success rates are lower when  $l_c = 0.4$ . This requirement on transaction cost is the most stringent one among the tested values since it allows, at most, a rate of transactions in the case of total turnover equal to 40%. The lowest success rate is realized when  $l_s = 0.4$  where more active positions are allowed, thus the trade-off between density and drastic reduction costs is more complicated.

We now want to investigate the mutual impact of the requirements on sparsity and transaction costs. As said, the main purpose of sparsity requirement is holding cost reduction. However, high sparsity level requirements also affect transaction cost since zeros are kept across time, avoiding transactions. Therefore, we expect that the request on sparsity  $l_s$  also affects the output value of  $\tau_2$ . This is confirmed by Fig. 4 that shows how  $\tau_1$  behaves in dependence of  $l_c$  and how  $\tau_2$  behaves in dependence of  $l_s$ , for the testing set.

More precisely, Fig. 4 depicts the box plot of  $\tau_1$  on the left side and the box plot of  $\tau_2$  on the right side. On the left side, we report the sparsity target on the  $x$ -axis, while the



**Fig. 4** Values of  $\tau_1$  (left) and  $\tau_2$  (right) for different financial targets

different colours refer to the target cost levels. As expected, the value of  $\tau_1$  provided by the neural network increases with respect to the sparsity level request. The figure also shows that for a fixed level of target sparsity,  $\tau_1$  increases marginally as the target cost increases. Looking at the right side of Fig. 4, we observe that  $\tau_2$  is decreasing with respect to the target cost since when more cost is allowed, less penalization is required. However, differently from what we observed on  $\tau_1$ ,  $\tau_2$  strongly depends on the sparsity target too: the values of  $\tau_2$  decrease considerably as the target sparsity increases.

To study the portfolio performance, in Table 3, we report the average Information Ratio (IR) (the average excess return per unit of volatility) and the Sharpe Ratio (SR) (the ratio between the average of the expected return of the portfolio and its standard deviation) for the different values of  $l_s$  and  $l_c$ . At each rebalancing date, the expected minimum wealth  $w_{min}$  is set to the expected wealth of the market index. We estimate the IR according to the following formula:

$$IR = \frac{\frac{1}{m} \sum_{j=1}^m AER_j}{\sigma(\mathbf{AER})}$$

where  $\mathbf{AER} = (AER_1, \dots, AER_m)$ , and

$$AER_j = \frac{\mathbf{w}_j^T \mathbf{1}_n - (\mathbf{w}_{min})_j}{(\mathbf{w}_{min})_j}, \quad j = 1, \dots, m.$$

The SR is measured as follows:

$$SR = \frac{\frac{1}{m} \sum_{j=1}^m R_j}{\sigma(\mathbf{R})}$$

where  $\mathbf{R} = (R_1, \dots, R_m)$ , and

$$R_j = \mathbf{r}_j^T \frac{\mathbf{w}_j}{\mathbf{w}_j^T \mathbf{1}_n}, \quad j = 1, \dots, m.$$

We report the average IR and SR of optimal portfolios obtained using Algorithm 1 with the regularization parameters provided by the random grid search (denoted with  $IR_{RG}$ ) and those provided by the NN (denoted with  $IR_{NN}$ ). We observe that the values of both ratios

**Table 3** Average Information Ratio and Sharpe Ratio of the optimal portfolios, for different levels of sparsity and costs, for the random grid search (third and fifth columns) and for the NN model (fourth and sixth columns)

$l_s$	$l_c$	$IR_{RG}$	$IR_{NN}$	$SR_{RG}$	$SR_{NN}$
0.4	0.4	0.59	0.59	1.73	1.73
0.4	0.5	0.57	0.54	1.73	1.73
0.4	0.6	0.49	0.50	1.72	1.72
0.5	0.4	0.58	0.54	1.74	1.73
0.5	0.5	0.50	0.50	1.73	1.73
0.5	0.6	0.42	0.47	1.72	1.72
0.6	0.4	0.51	0.51	1.73	1.74
0.6	0.5	0.43	0.48	1.74	1.73
0.6	0.6	0.39	0.44	1.73	1.73

**Table 4** Percentage of success in achieving the financial target for different values of  $l_s$  and  $l_c$ , and for time windows shifted by one (third column), two (fourth column) and three (fifth column) quarters

$l_s$	$l_c$	3 months (%)	6 months (%)	9 months(%)
0.4	0.4	98.60	96.00	96.80
0.4	0.5	99.20	99.20	97.70
0.4	0.6	99.70	99.20	98.60
0.5	0.4	99.80	99.00	99.30
0.5	0.5	99.90	99.40	99.90
0.5	0.6	99.80	99.90	100.00
0.6	0.4	99.90	99.90	100.00
0.6	0.5	100.00	100.00	100.00
0.6	0.6	100.00	100.00	99.90
Total		99.66	99.07	99.10

are similar, highlighting that the NN approach allows for obtaining optimal portfolios with comparable financial performance.

Finally, we investigate the ability of the network trained on data examples related to time  $t_{ref}$ , to provide regularization parameters that allow obtaining portfolios with desired financial properties on a future date  $t_{ref} + \delta$  with  $\delta > 0$ . In Table 4, we report the percentage of success in achieving the financial target for different values of  $l_s$  and  $l_c$ , and  $\delta = 0.25, 0.5, 0.75$  years. We see that in some cases, the percentage of success decreases as  $\delta$  grows. However, the percentage of success decreases slowly. This result suggests that the function learned by the NN can also be used in future dates, saving much computational time since the training process does not have to be repeated every time. In addition, this evidence suggests that the functional relationship between the asset returns and the regularization parameters does not substantially change over time.

## 5.2 FTSE MIB

To confirm the effectiveness of the proposed approach in achieving financial targets and obtaining portfolios with valuable financial performance, we test our model on the data related to the equities of an alternative index—the FTSE MIB. This index serves as the primary stock market indicator for the Italian stock exchange, Borsa Italiana, capturing the performance of Italy's largest and most liquid companies. We collected the historical weekly return time series from November 2018 to November 2023 for 38 of the 40 securities (2 were excluded due to insufficient data). In Table 5, we report detailed information about the dataset used in this analysis: a number identifying the asset, the name of the assets included in the index, with the mean and the standard deviation of the returns time series and the average market capital. We solve the portfolio selection problem for different combinations of financial targets using the regularization parameters obtained through the Neural Network. The investment horizon is 2.5 years with  $m = 5$  half-year rebalancing dates. In Table 6, we report the performance in terms of Information Ratio and Sharpe Ratio. For completeness, we also report the values for  $\tau_1$  and  $\tau_2$  obtained by the Neural Network for each pair of financial targets. In all tests, the IR is slightly higher than 0.4, and the SR is always greater than one, confirming the effectiveness of the proposed approach.

Finally, in Fig. 5 we illustrate the optimal portfolio composition for  $l_s = 0.5$  and  $l_c = 0.5$ . On the left side of Fig. 5a, we show the sparsity pattern of weights for FTSE-MIB. Assets

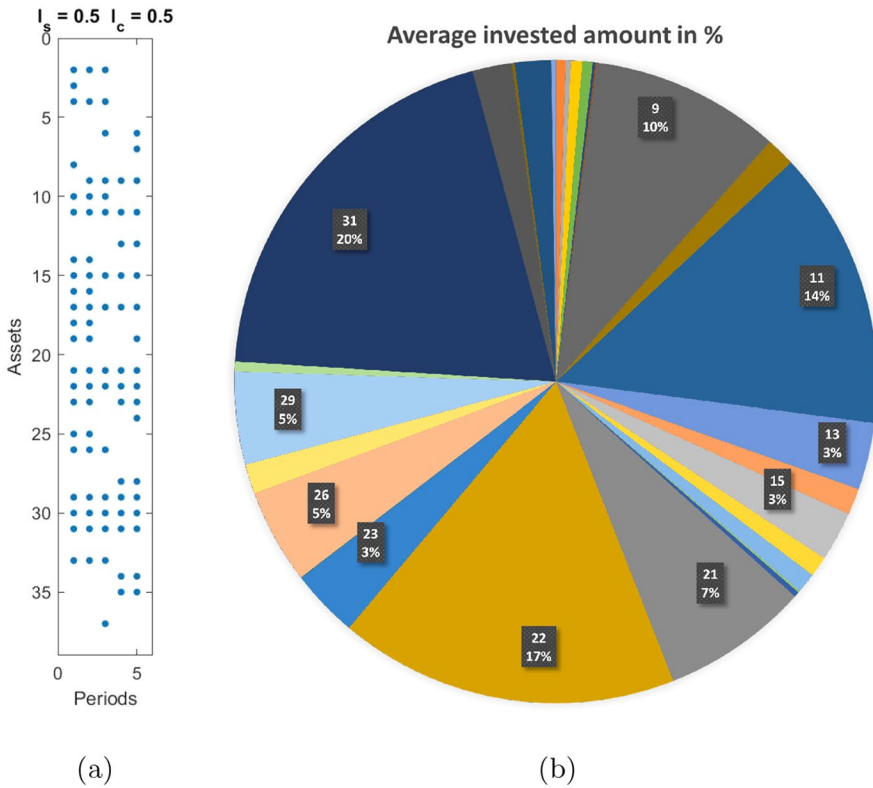
**Table 5** FTSE MIB dataset. First column: number identifying the asset. Second column: name of the asset. Third-fourth columns: mean and standard deviation of the return time series. Fifth column: average market value

ID	Company	Mean	SD	Market cap
1	A2A	0.0017	0.0389	4723.27
2	AMPLIFON	0.0035	0.0492	6605.90
3	AZIMUT HOLDING	0.0037	0.0476	2711.25
4	BANCA GENERALI	0.0031	0.0441	3416.12
5	BANCA MEDIOLANUM	0.0026	0.0449	5449.27
6	BANCA MPS	-0.0057	0.0842	1605.51
7	BANCO BPM	0.0060	0.0617	3832.88
8	BPER BANCA	0.0034	0.0657	2370.61
9	CAMPARI	0.0023	0.0332	11285.85
10	CNH.INDUSTRIAL	0.0024	0.0522	15252.05
11	DIASORIN	0.0012	0.0470	7231.70
12	ENEL	0.0020	0.0369	65948.92
13	ENI	0.0010	0.0446	43508.73
14	ERG	0.0024	0.0420	3561.41
15	FERRARI	0.0048	0.0399	37620.70
16	FINCOBANK	0.0018	0.0450	7649.94
17	ASSICURAZIONI GENERALI	0.0017	0.0345	25955.68
18	HERA	0.0011	0.0353	4701.53
19	INTERPUMP GROUP	0.0029	0.0468	4244.16
20	INTESA SANPAOLO	0.0021	0.0468	39267.01
21	INFR. WIRELESS SPA NPV	0.0028	0.0388	8131.03
22	ITALGAS	0.0006	0.0299	4411.81
23	LEONARDO	0.0032	0.0586	4971.44
24	MEDIOBANCA	0.0026	0.0473	7859.69
25	MONCLER	0.0028	0.0446	12355.62
26	PIRELLI & C	-0.0003	0.0472	4776.74
27	POSTE.ITALIANE	0.0023	0.0402	12351.59
28	PRYSMIAN	0.0035	0.0412	7233.40
29	RECORDATI INDUA CHIMICA	0.0021	0.0352	8852.47
30	SAIPEM	-0.0016	0.0954	2701.11
31	SNAM	0.0011	0.0286	15686.65
32	STELLANTIS	0.0032	0.0522	35171.96
33	STMICROELECTRONICS	0.0056	0.0561	27505.45
34	TELECOM	-0.0005	0.0693	5748.38
35	TENARIS	0.0025	0.0570	12686.98
36	TERNA	0.0023	0.0306	13018.18
37	UNICREDIT	0.0046	0.0617	25180.78
38	UNIPOL	0.0024	0.0458	3197.57

versus periods are represented, thus a dot at position  $(i, j)$  is an active position in asset  $i$  in period  $j$ . On the right side of Fig. 5b, we report the average investment in percentage. We note that the two highest values correspond to the assets with the lowest volatilities

**Table 6** Information Ratio and Sharpe Ratio of the optimal portfolios, for different levels of sparsity and costs for the NN model on FTSE-MIB data

$l_s$	$l_c$	$\tau_1$	$\tau_2$	$IR_{NN}$	$SR_{NN}$
0.4	0.4	1.53E-03	5.20E-04	0.4065	1.3085
0.4	0.5	1.55E-03	3.45E-04	0.4068	1.2939
0.4	0.6	1.58E-03	2.43E-04	0.4019	1.3096
0.5	0.4	2.20E-03	3.95E-04	0.4064	1.3102
0.5	0.5	2.24E-03	2.72E-04	0.4071	1.3175
0.5	0.6	2.27E-03	2.01E-04	0.4067	1.3257
0.6	0.4	2.64E-03	3.07E-04	0.4074	1.3212
0.6	0.5	2.70E-03	2.21E-04	0.4052	1.3281
0.6	0.6	2.76E-03	1.71E-04	0.4099	1.3298



**Fig. 5** Left: Sparsity pattern of the optimal portfolio FTSE MIB with investment horizon 5 years. Right: Average amounts invested in the FTSE MIB companies. Target are fixed to  $l_s = 0.5$  and  $l_c = 0.5$

(SNAM, asset 31 and ITALGAS, asset 22), according to the mean-variance framework principle. In Fig. 5a, we observe that assets 22 and 31 are kept along all the investment strategies. On the other hand, the assets that are not selected in the optimal portfolio, such as 1, 5, and 27, generally exhibit high positive correlations with the others and low average returns.



## 6 Conclusion

In this work, we present a data-driven approach for the automatic computation of the regularization parameters in a fused lasso portfolio selection problem where a financial target is fixed. Starting from the results obtained in Corsaro et al. (2022) to detect the regularization parameter in a lasso model, we extend the use of NN to problem (1). The increased complexity of the model motivates the use of more sophisticated NN. Moreover, we propose to use Long Short-Term Memory networks, specifically designed for processing sequential data. This design allows for the direct application of the networks to time series data of log-returns, eliminating the need for a priori identification of relevant features. Results show that the network effectively learns the functional relation between the regularization parameters and input data. Moreover, preliminary tests show that LSTM networks allow one, at least under stable market conditions, to use the learnt function in future periods, that is, successively to the investment period employed for training the network. Whether one can assume that the output of the training process can be kept over time is to be investigated and will be the subject of future work. In future research, we also intend to investigate the use of other Deep Learning models such as Convolutional Neural Networks and Self-Attention-based mechanism (Vaswani et al. 2017) to learn the optimal regularization to apply and to use functional data clustering methods (Levantesi et al. 2023) to identify similarities among the assets and develop a more appropriate portfolio selection strategy.

**Acknowledgements** This work was partially supported by Italian Ministry of University and Research (MIUR), PRIN Projects: *Numerical Optimization with Adaptive Accuracy and Applications to Machine Learning*, grant n. 2022N3ZNAX, *Modeling and valuation of financial instruments for climate and energy risk mitigation*, grant n. 2022FPLY97, and by Istituto Nazionale di Alta Matematica-Gruppo Nazionale per il Calcolo Scientifico (INdAM-GNCS), by MIUR. We thank the anonymous reviewers for their careful reading of the manuscript and their insightful remarks and suggestions, which allowed us to improve the quality of our work.

**Funding** Open access funding provided by Università Parthenope di Napoli within the CRUI-CARE Agreement. The authors have not disclosed any funding.

## Declarations

**Conflict of interest** Authors declare that they have no conflict of interest.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Abedin, M.Z., Moon, M.H., Hassan, M.K., Hajek, P.: Deep learning-based exchange rate prediction during the covid-19 pandemic. *Ann. Oper. Res.* 1–52 (2021)
- Beer, J.C., Aizenstein, H.J., Anderson, S.J., Krafty, R.T.: Incorporating prior information with fused sparse group lasso: application to prediction of clinical measures from neuroimages. *Biometrics* **75**(4), 1299–1309 (2019)
- Bruni, R., Cesarone, F., Scozzari, A., Tardella, F.: Real-world datasets for portfolio selection and solutions of some stochastic dominance portfolio models. *Data in Brief* 185–209 (2016)
- Chen, Z., Consigli, G., Liu, J., Li, G., Fu, T., Hu, Q.: Multi-period risk measures and optimal investment policies. In: Consigli, G., Kuhn, P.B.D. (eds.) *Optimal Financial Decision Making Under Uncertainty*, pp. 1–34. Springer, Cham (2017)
- Corsaro, S., De Simone, V.: Adaptive  $l_1$ -regularization for short-selling control in portfolio selection. *Comput. Optim. Appl.* **72**, 457–478 (2019)
- Corsaro, S., De Simone, V., Marino, Z., Perla, F.:  $L_1$ -regularization for multi-period portfolio selection. *Ann. Oper. Res.* **294**, 75–86 (2020)
- Corsaro, S., De Simone, V., Marino, Z.: Fused lasso approach in portfolio selection. *Ann. Oper. Res.* **299**, 47–59 (2021a)
- Corsaro, S., De Simone, V., Marino, Z.: Split Bregman iteration for multi-period mean variance portfolio optimization. *Appl. Math. Comput.* **392**, 125715 (2021b)
- Corsaro, S., De Simone, V., Marino, Z., Scognamiglio, S.:  $l_1$ -regularization in portfolio selection with machine learning. *Mathematics* **10**, 4 (2022)
- De Haas, R., Van Horen, N.: International shock transmission after the lehman brothers collapse: Evidence from syndicated lending. *Am. Econ. Rev.* **102**(3), 231–37 (2012)
- De Simone, V., di Serafino, D., Viola, M.: A subspace-accelerated split Bregman method for sparse data recovery with joint  $l_1$ -type regularizers. *Electron. Trans. Numer. Anal.* **53**, 406–425 (2020)
- De Simone, V., di Serafino, D., Gondzio, J., Pougkakiotis, S., Viola, M.: Sparse approximations with interior point methods. *SIAM Rev.* **64**(4), 954–988 (2022)
- Dijkstra, T.K.: Ridge regression and its degrees of freedom. *Qual. Quant.* **48**, 3185–3193 (2014)
- Ding, Y.: Portfolio selection under maximum minimum criterion. *Qual. Quant.* **40**(3), 457–468 (2006)
- Ehrmann, M., Fratzscher, M.: Euro area government bonds-fragmentation and contagion during the sovereign debt crisis. *J. Int. Money Financ.* **70**, 26–44 (2017)
- Elman, J.L.: Finding structure in time. *Cogn. Sci.* **14**(2), 179–211 (1990)
- Goldstein, T., Osher, S.: The split Bregman method for  $L_1$ -regularized problems. *SIAM J. Imaging Sci.* **2**(2), 323–343 (2009)
- Goodfellow, I.J., Bengio, Y., Courville, A.: *Deep Learning*. MIT Press, Cambridge (2016)
- Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**(5), 359–366 (1989)
- Jauhar, S.K., Raj, P.V.R.P., Kamble, S., Pratap, S., Gupta, S., Belhadi, A.: A deep learning-based approach for performance assessment and prediction: a case study of pulp and paper industries. *Ann. Oper. Res.* 1–27 (2022)
- Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
- Lajili-Jarjir, S., Rakotondratsimba, Y.: The number of securities giving the maximum return in the presence of transaction costs. *Qual. Quant.* **42**, 613–644 (2008)
- Lee, J., Chen, J.: A modified information criterion for tuning parameter selection in  $l_1$  fused lasso for inference on multiple change points. *J. Stat. Comput. Simul.* **90**(8), 1496–1519 (2020)
- Levantesi, S., Nigri, A., Piscopo, G., Spelta, A.: Multi-country clustering-based forecasting of healthy life expectancy. *Qual. Quant.* 1–27 (2023)
- Ma, Y., Han, R., Wang, W.: Portfolio optimization with return prediction using deep learning and machine learning. *Expert Syst. Appl.* **165**, 113973 (2021)
- Osher, S., Burger, M., Goldfarb, D., Xu, J., Yin, W.: An iterative regularization method for total variation-based image restoration. *Multiscale Model Sim.* **4**(2), 460–489 (2005)
- Perla, F., Richman, R., Scognamiglio, S., Wüthrich, M.V.: Time-series forecasting of mortality rates using deep learning. *Scand. Actuar. J.* **2021**(7), 572–598 (2021)
- Slavici, T., Maris, S., Pirtea, M.: Usage of artificial neural networks for optimal bankruptcy forecasting case study: Eastern European small manufacturing enterprises. *Qual. Quant.* **50**, 385–398 (2016)
- Sun, X., Xu, W., Jiang, H., Wang, Q.: A deep multitask learning approach for air quality prediction. *Ann. Oper. Res.* **303**(1), 51–79 (2021)
- Torrente, M.-L., Uberti, P.: Risk-adjusted geometric diversified portfolios. *Qual. Quant.* 1–21 (2023)

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30** (2017)
- Wang, Y.-H.: Using neural network to forecast stock index option price: a new hybrid Garch approach. *Qual. Quant.* **43**, 833–843 (2009)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.