mathematics

MDPI

Article

l_1-Regularization in Portfolio Selection with Machine Learning

Stefania Corsaro [1,†], Valentina De Simone [2,†], Zelda Marino[1,†] and Salvatore Scognamiglio [1,*,†]

[1] Department of Management and Quantitative Studies, Parthenope University of Naples, 80133 Naples, Italy; stefania.corsaro@uniparthenope.it (S.C.); zelda.marino@uniparthenope.it (Z.M.)
[2] Department of Mathematics and Physics, University of Campania "Luigi Vanvitelli", 81100 Caserta, Italy; valentina.desimone@unicampania.it
* Correspondence: salvatore.scognamiglio@uniparthenope.it
† These authors contributed equally to this work.

Abstract: In this work, we investigate the application of Deep Learning in Portfolio selection in a Markowitz mean-variance framework. We refer to a l_1 regularized multi-period model; the choice of the l_1 norm aims at producing sparse solutions. A crucial issue is the choice of the regularization parameter, which must realize a trade-off between fidelity to data and regularization. We propose an algorithm based on neural networks for the automatic selection of the regularization parameter. Once the neural network training is completed, an estimate of the regularization parameter can be computed via forward propagation. Numerical experiments and comparisons performed on real data validate the approach.

Keywords: deep learning; multi-period portfolio optimization; l_1-norm; split Bregman

Citation: Corsaro, S.; De Simone, V.; Marino, Z.; Scognamiglio, S. l_1-Regularization in Portfolio Selection with Machine Learning. *Mathematics* **2022**, *10*, 540. https://doi.org/10.3390/math10040540

Academic Editors: Catalin Stoean and J. E. Trinidad-Segovia

Received: 30 December 2021
Accepted: 7 February 2022
Published: 9 February 2022

1. Introduction

Regularization parameter selection is one of the most essential tasks in solving large-scale ill-posed problems. Problems of this kind arise in various applications, and generally, their solution requires some regularization; that is, the problem is substituted by a related one with better numerical properties. A common approach is to add a penalty term that enforces uniqueness and stability. The penalty term is tuned using a regularization parameter. It must realize a trade-off between fidelity to data and regularization. If the regularization parameter is too small, the model has numerical features similar to the unregularized one; on the other hand, the solution does not fit the original model if it is too big. In the context of Portfolio optimization, different regularization techniques have been suggested for the mean-variance Markowitz model [1]. Among these, l_1 penalization has been considered. This is an effective technique to obtain sparse portfolios that allow the investor to reduce both the number of positions to be monitored and the holding costs [2–5]. While the literature offers many methods for Tikhonov regularization, l_1 regularization parameter selection is often based on problem-dependent criteria and related to iterative empirical estimates, requiring a high computational cost. In [2], a least-angle regression algorithm is presented, which starts from large values and proceeds by decreasing them. In [3,5], the authors propose a modified version of the Bregman iteration procedure, which includes an adaptive rule for the selection of the regularization parameter, respectively, in the single-period and multi-period framework. In that algorithm, the starting value is very small and is increased within an iterative procedure. In this work, we explore the use of supervised Machine Learning (ML) for the automatic selection of the regularization parameter in the multi-period portfolio selection model presented in [5]. ML provide methods which are data-driven; it is actually extensively applied to Finance [6–9], in particular to the portfolio selection problem [10]. Several contributions concern the use of ML models to predict stock returns [11–13]. In [14], the authors adopt deep learning models to optimize the portfolio Sharpe Ratio directly. The idea of using ML for tuning parameters is investigated in [15,16].

In [15], the authors use deep learning networks to compute the regularization parameter for solving inverse problems, while in [16], ML techniques are applied to adjust the risk aversion coefficient in the portfolio optimization context. The aim of ML is to develop algorithms which can learn and progress over time and can be used for predictions. In particular, the goal of supervised learning is to predict the value of one or more outputs for a set of inputs. Thus, we aim at approximating $f : \mathcal{X} \subseteq \mathbb{R}^q \longrightarrow \mathcal{Y} \subseteq \mathbb{R}^t$ with a function

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{\theta}),$$

where $f_{\boldsymbol{\theta}} : \mathcal{X} \longrightarrow \mathcal{Y}$ is usually nonlinear and $\boldsymbol{\theta} \in \mathbb{R}^q$ is a large set of unknown parameters. The learning phase uses a training set to produce a set of parameters $\boldsymbol{\theta}$ that minimize a loss (or cost) function $\mathcal{L}$ that measures the accuracy of the predicted $f_{\boldsymbol{\theta}}(\mathbf{x})$ with respect to $f(\mathbf{x})$. There are many kinds of loss functions in supervised learning, such as the square Euclidean distance, crossentropy, contrast loss, hinge loss, information gain, Poisson deviance, and so on [17].

In this paper, we consider the Neural Networks (NN) that have become particularly popular among Machine Learning methods [18,19]. NNs were originally conceived as models that would imitate the function of the human brain, that is, a set of neurons joined together by a bunch of connections. Neurons, in this context, are a weighted sum of inputs; they are intended as derived features. A nonlinear activation function is applied to neurons to compute the output (the target). These ideas have existed since the 1960s, but their power has been fully exploited with the advent of modern computing architectures. They have shown their ability to perform well on supervised learning tasks, mainly when training data are abundant. There are typically three parts in a NN: an input layer, one or more hidden layers, and an output layer. Each layer is a collection of neurons (units) connected with the previous layer with synapses (or weights) optimized during the training process. The network learns by examining data examples, generates a prediction for each unit, and makes adjustments to the weights whenever it makes an incorrect prediction. This process is repeated many times, and the network continues to improve its predictions until one or more stopping criteria have been met. The simplest networks contain no hidden layers and are equivalent to linear regressions.

The paper is organized as follows. In Section 2, we recall the multi-period l_1-regularized model and the algorithm based on Bregman iteration used to solve it. In Section 3, we describe how NN is used for the selection of the regularization parameter. In Section 4, we show numerical experiments that validate our approach. Finally, in Section 5, some conclusions are given.

2. Multi-Period l_1-Regularized Mean-Variance Markowitz Model

In this section, we recall the l_1-regularized model for multi-period portfolio selection introduced in [5]. The investment strategy is either a medium or a long term one; thus, decisions can change according to the time evolution of available information.

In a multi-period setting, time is assumed to evolve continuously, but the investment period is split into m sub-periods, delimited by the so-called rebalancing dates j, $j = 1, \ldots, m$; decisions are taken at rebalancing dates and kept within sub-periods. Let n be the number of assets and $\mathbf{u}_j^T \in \Re^n$ the portfolio held at the rebalancing date j. The optimal portfolio referred to the overall investment period is thus defined by the vector $\mathbf{u} = (\mathbf{u}_1^T, \mathbf{u}_2^T, \ldots, \mathbf{u}_m^T)^T \in \Re^N$, where $N = m \cdot n$; $\mathbf{u}_j$ is the portfolio at time j; thus, the element $u_{i+(j-1)\cdot n}$ is the amount invested on asset i at time j. We develop our model in a mean-variance Markowitz framework; thus, we aim at minimizing the risk of the strategy, imposing constraints on expected wealth. The multi-period portfolio modelling in a Markowitz framework has been extensively analyzed; we address to [20,21] and references therein for the theoretical aspects of the formulation of the Markowitz model in the multiperiod case.

We represent the risk measure as a sum of terms. Each one provides a risk estimate in one investment period, using information available at the beginning of the period. In

particular, we consider the variance as a single-period risk measure. The risk of the strategy is then given by:

$$F(\mathbf{u}) = \sum_{j=1}^{m} \mathbf{u}_j^T C_j \mathbf{u}_j,$$

where $C_j \in \mathbb{R}^{n\times n}$ is the covariance matrix at time j, assumed to be positive definite. The choice of functional $F(\mathbf{u})$ allows, on one hand, to deal with quadratic programming, on the other, to use a time consistent risk measure, as discussed in [22]. l_1-regularization is applied to improve conditioning, to produce sparse portfolios [3–5,23,24]. The penalty term is weighted by means of the regularization parameter, denoted with τ in the following.

The problem is a non-smooth optimization one, with equality constraints:

$$\min_{\mathbf{u}} \sum_{j=1}^{m} \left[\mathbf{u}_j^T C_j \mathbf{u}_j + \tau \|\mathbf{u}_j\|_1 \right] \tag{1a}$$

s.t.

$$\mathbf{u}_1^T \mathbf{1}_n = \xi_{\text{init}} \tag{1b}$$

$$\mathbf{u}_j^T \mathbf{1}_n = (\mathbf{1}_n + \mathbf{r}_{j-1})^T \mathbf{u}_{j-1}, \quad j = 2, \ldots, m \tag{1c}$$

$$\mathbf{u}_m^T \mathbf{1}_n = \xi_{\text{term}}. \tag{1d}$$

ξ_{init} is the invested amount, that is, constraint (1b) fixes the initial wealth. $\mathbf{r}_j$ is the expected return vector estimated at time j, so equation (1c) imposes that the strategy is self-financing: the portfolio at time j is the revaluation of its value at time $j-1$. ξ_{term} is the target wealth that the investor gains when the investment ends, as stated in (1d). Since the target wealth is fixed only at the final date, it is assumed that the investor does not exit the investment before the end.

Both the objective function and the constraints can be reformulated in compact form, leading to the following equivalent problem:

$$\begin{array}{l} \min_{\mathbf{u}} \; J(\mathbf{u}) \equiv \mathbf{u}^T C \mathbf{u} + \tau \|\mathbf{u}\|_1 \\ \text{s.t.} \;\; A\mathbf{u} = \mathbf{b} \end{array} \tag{1}$$

where $C = diag(C_1, C_2, \ldots, C_m) \in \mathbb{R}^{N\times N}$ is a $m \times m$ diagonal block matrix, A is a $m \times m$ lower bidiagonal block matrix, with blocks of dimension $1 \times n$:

$$\begin{array}{lcl} diag(A) & = & (\mathbf{1}_n^T, \mathbf{1}_n^T, ..., \mathbf{1}_n^T) \\ subdiag(A) & = & (-(\mathbf{1}_n + \mathbf{r}_1)^T, ..., -(\mathbf{1}_n + \mathbf{r}_{m-1})^T) \end{array}$$

and $\mathbf{b} = (\xi_{\text{init}}, 0, \ldots, \xi_{\text{term}})^T \in \mathbb{R}^m$. The linear system in (1) incorporates the temporal constraints (1a–c). In particular, the first and the last rows of the system state the requirements on the initial and the terminal wealth, respectively. Intermediate equations state the self-financing property of the strategy. We assume that A is full rank to guarantee the existence of solutions [25]. In this work, we present a procedure based on Bregman iteration [26]. It is a well-established method for the solution of l_1-regularized optimization problems, successfully applied in different fields, including finance [27]. Methods based on Bregman iteration have proved to be efficient for the solution of problem (1) as well [3,5,23,24]. Bregman iterative scheme requires at each step the solution of a l_1-regularized unconstrained optimization subproblem, in which the functional J is replaced by its Bregman distance at the current iterate $\mathbf{u}$

$$D_J^{\mathbf{p}}(\mathbf{v}, \mathbf{u}) = J(\mathbf{u}) - J(\mathbf{v}) - < \mathbf{p}, \mathbf{u} - \mathbf{v} >,$$

where $\mathbf{p}$ is a subgradient in the subdifferential of J at point $\mathbf{u}$. Bregman iteration applied to problem (1) produces the Algorithm 1.

Algorithm 1 Bregman Iteration for Portfolio Selection (BIPS).

Given τ, λ
$k := 0$
$\mathbf{u}_0 := \mathbf{0}, \mathbf{p}_0 := \mathbf{0}$
while not convergence **do**

% solve the inner minimization problem

$$\mathbf{u}_{k+1} = \text{argmin}_{\mathbf{u}} D_J^{\mathbf{p_k}}(\mathbf{u}, \mathbf{u}_k) + \frac{\lambda}{2}\|A\mathbf{u} - \mathbf{b}\|_2^2 \tag{2}$$

% update subgradient

$$\mathbf{p}_{k+1} = \mathbf{p}_k - \lambda A^T(A\mathbf{u}_{k+1} - \mathbf{b}) \tag{3}$$

$k := k + 1$
end while
$it_{opt} := k$
$u_{opt} := u_k$

Note that a cheaper version of the Bregman iteration can be obtained. Following [28], thanks to the linearity of the equality constraints, a simplified formulation can be derived; Equations (2) and (3) can be replaced by:

$$\begin{aligned} \mathbf{u}_{k+1} &= \text{argmin}_{\mathbf{u}} J(\mathbf{u}) + \tfrac{\lambda}{2}\|A\mathbf{u} - \mathbf{b}_k^u\|_2^2 \\ \mathbf{b}_{k+1}^u &= \mathbf{b}_k^u + (\mathbf{b} - A\mathbf{u}_{k+1}); \;\; \mathbf{b}_0^u = \mathbf{b}. \end{aligned}$$

In this version, the Bregman vectors $\mathbf{b}_k^u$ inside the quadratic penalty function enforce the equality constraints and allow the use of functional J in place of its Bregman distance.

A crucial issue in the solution of (1) is the choice of a suitable value for the regularization parameter τ that realizes a trade-off between sparsity (requiring sufficiently large values) and fidelity to data (requiring small values). Several approaches have been proposed, both in the single and multi-period case [2,3,5], all based on an adaptive rule to select the regularization parameter. The idea is to change the value of τ during the optimization process until a financial target is satisfied. In [5] a modified version of Algorithm 1, which we here denote with A-BIPS (Adaptive BIPS) is presented, developed in a multi-period setting. The algorithm A-BIPS produces an increasing sequence of values $\tau_0 \leq \tau_1 \leq \ldots \tau_k$, where

$$\tau_i = \begin{cases} \mu\tau_{i-1} & \text{if the financial target is not met} \\ \tau_{i-1} & \text{otherwise} \end{cases} \tag{4}$$

with $\mu > 1$ and $0 < i \leq k$.

To make the Bregman distance associated with $J_k(\mathbf{u}) = \mathbf{u}^T C\mathbf{u} + \tau_k\|\mathbf{u}\|_1$ at point u_k well defined if $\tau_k \neq \tau_{k-1}$, the subgradient p_k must be updated as follows:

$$\mathbf{p}_k = \frac{\tau_k}{\tau_{k-1}}\mathbf{p}_k + \left(1 - \frac{\tau_k}{\tau_{k-1}}\right)C\mathbf{u}_k,$$

before computations in (2) and (3). So, the method presented in [5] produces the optimal portfolio as well as the optimal parameter:

$$\tau_{opt} = \mu^h \tau_0, \tag{5}$$

where h is not greater than the total number of Bregman iterations. Numerical results show that the τ_{opt} computed by procedure A-BIPS generally raise portfolios that exhibit a percentage of sparsity significantly greater than the required one; this could affect diversification. For this reason, in the next section, we propose a NN approach to approximate τ_{opt} using a grid method based on Algorithm 1.

3. Neural Networks for Regularization Parameter Selection

The value h in (5) is unknown a priori, and it is strongly dependent on data and the financial target. In this work, we assume that the financial target is defined in terms of the maximum number of active positions in the optimal portfolio; that is, we require a minimum number N_{sparse} of zeros in the solution. It is then reasonable to assume that h is a function of certain features of data, that is, we assume that a nonlinear target function $f : \mathcal{X} \longrightarrow \mathcal{Y}$ exists, such that $h = f(\mathbf{x})$, where $\mathcal{X} \subseteq \mathbb{R}^q$, $\mathcal{Y} \subseteq \mathbb{N}_0$ and q is the number of features. The function f is unknown and could be complex; we employ ML techniques to derive its approximation. In this framework, deep NNs are a natural candidate to perform this task since they are known as universal function approximators [29].

It is mandatory to select features containing significant statistical information on data to train the network effectively. Let n be the number of assets and M the length of the return time series. The dataset is represented by the matrix $R = (R_{*1}, R_{*2}, \ldots, R_{*n}) \in \mathbb{R}^{M \times n}$, which has the complete return time series of each asset along the columns. We denote with $\bar{\mathbf{r}} \in \mathbb{R}^n$ the vector of asset average returns. In this work, we select $q = 6$ features that are significant in a mean-variance framework, where only the first and second-order moments are considered, and take into account the financial target. The feature vector is defined component-wise as:

$$
\begin{aligned}
x_1 &= \max(\bar{\mathbf{r}}) \\
x_2 &= E[\bar{\mathbf{r}}] \\
x_3 &= \max_{i=1,\ldots,n} \sqrt{E[(R_{*i} - \bar{r}_i \mathbf{1}_m)^2]} \\
x_4 &= \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \sqrt{E[(R_{*i} - \bar{r}_i \mathbf{1}_m)^2)^2]} \\
x_5 &= \max_{i=1,\ldots,n, k<i} \sqrt{E[(R_{*i} - \bar{r}_i \mathbf{1}_m)^2 (R_{*k} - \bar{r}_k \mathbf{1}_m)^2)]} \\
x_6 &= \tfrac{N_{sparse}}{N}
\end{aligned}
$$

where E is the expectation operator. More in details, x_1 and x_2 are respectively the maximum and the mean of the asset average returns; x_3 and x_4 are respectively the maximum and the mean standard deviations of asset returns; x_5 is the maximum covariance of asset returns, and finally x_6 is the desired sparsity degree in the solution.

The NN requires a suitable set, containing L samples $\{\mathbf{x}_l, h_l\}$, $l = 1, \ldots, L$, which is split into the training and the testing sets. To build it, we generate L equally sized subsets of asset returns and compute the related features $\mathbf{x}$. For each subset we iterate Algorithm 1 to compute the corresponding value h_{opt}. In more detail, starting from $\tau = \tau_0$ we compute the solution to problem (1); if the financial target is not met, we increase τ according to (4) and solve the problem again. The procedure is summarized in Algorithm 2.

Algorithm 2 Iterative BIPS (I-BIPS).

Given τ_0, τ_{max}, $\mu > 1$
$\tau = \tau_0$
$flag = 0$
repeat
  compute u using BIPS
  if $nnz(u) > N - N_{sparse}$ **then**
    % increase τ since the target is not satisfied
    $\tau = \mu \cdot \tau$
  else
    $flag = 1$
  end if
until $flag = 1$ or $\tau > \tau_{max}$
$\tau_{opt} = \tau$

Once the training set has been obtained, we use NNs to approximate the functional relationship between $\mathbf{x}$ and h. The procedure is described in Algorithm 3.

In particular, we use a deep NN, described below.

Algorithm 3 Neural Networks for Portfolio Selection (NNPS).

Given L datasets, τ_0, $\mu > 1$
define L_{train} and L_{test} s.t. $L = L_{train} + L_{test}$
% compute $\mathbf{x}_l$ and h_l for each group of assets in the training set
for l=1, . . ., L_{train} **do**
  compute features $\mathbf{x}_l$
  % find the minimum value of τ that realizes the target sparsity
  compute $\tau_{opt,l}$ using I-BIPS
  compute $h_l = \log_\mu \left(\frac{\tau_{opt,l}}{\tau_0} \right)$
end for
% training phase
learn the approximation of $\hat{f}^{(NN)}$ via NNs from the sample $\{\mathbf{x}_l, h_l\}_l^{L_{train}}$
% testing phase
% compute the optimal portfolio for each group of assets in the testing set
for l=1, . . ., L_{test} **do**
  compute features $\mathbf{x}_l$
  compute $h_l^{(NN)} = \hat{f}^{(NN)}(\mathbf{x}_l)$
  compute $u_{opt,l}^{(NN)}$ and $it_{opt,l}^{(NN)}$ using BIPS with $\tau_{opt,l}^{(NN)} = \tau_0 \mu^{h_l^{(NN)}}$
end for

Deep Neural Network Architecture

In this section, we introduce deep NNs in a general setting. The deep NNs provide a sequence of stacked layers where each layer processes as input the output of the previous one. Let $D \in \mathbb{N}$ be the number of hidden layers and $q_k \in \mathbb{N}$ for $k = 1, \ldots, D$ be a sequence of integers indicating the number of units in each layer; let $q_0 = q$ and $q_D = 1$, the mechanism behind a deep learning network can be formalized as follows:

$$\begin{aligned}
\mathbf{z}^{(1)}(\mathbf{x}) &= \phi_1\left(\mathbf{w}_0^{(1)} + W^{(1)}\mathbf{x}\right) \\
\mathbf{z}^{(2)}(\mathbf{x}) &= \phi_2\left(\mathbf{w}_0^{(2)} + W^{(2)}\mathbf{z}^{(1)}(\mathbf{x})\right) \\
&\ldots \\
\mathbf{z}^{(D)}(\mathbf{x}) &= \phi_D\left(\mathbf{w}_0^{(D)} + W^{(D)}\mathbf{z}^{(D-1)}(\mathbf{x})\right),
\end{aligned}$$

where $\mathbf{w}_0^{(k)} \in \mathbb{R}^{q_k}$, $W^{(k)} \in \mathbb{R}^{q_k \times q_{k-1}}$ are the weights and $\phi_k(\cdot)$ for $k = 1, \ldots, D$ denote the activation functions. Possible choices for the activation function are:

- sigmoid function $\phi(x) = \frac{1}{1+e^{-x}}$,
- tanh function $\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$,
- relu function $\phi(x) = max(0, x)$.

The performance of the network depends on the value of the weights $w_{l,j}^{(k)}$, which must be properly calibrated. This training process involves an unconstrained optimization problem where, chosen a suitable loss function $\mathcal{L}(w_{l,j}^{(k)}, \cdot)$, its minimum is sought. The back-propagation algorithm is the most used for the training of feed-forward NNs. The algorithm compares the predicted values against the desired ones (objective) and modifies the weights

by back-propagating the gradient of the loss function. A first-order stochastic gradient-based optimization algorithm, based on adaptive estimates of lower-order moments, was used [30]. However, when D is very large, the gradients often get smaller and smaller and approach zero, which eventually leaves the weights of the first layers unchanged. This issue is known as the vanishing gradients problem. To fight the vanishing gradient problem that affects deeper learning networks, skip connections were introduced [31]. They are additional connections that directly connect the input layer to the output layer and concatenate the input data with the output of the last hidden layer. In case of skip connections, the output layer reads:

$$\mathbf{z}^{(D)}(\mathbf{x}) = \phi_D\left(\mathbf{w}_0^{(D)} + W^{(D)}\mathbf{z}^{(D-1)}(\mathbf{x}) + \langle \mathbf{w}^{(skip)}, \mathbf{x}\rangle\right)$$

where $\mathbf{w}^{(skip)} \in \mathbb{R}^{q_0}$ are the weights associated to the input in the output layer and $\langle\cdot,\cdot\rangle$ denotes the scalar product. Several possible choices for the loss function L are possible. Since we desire to model the number of iterations h, the response variable is an integer. For these kinds of problems, some authors have successfully applied the Poisson loss function to train the network [32,33]. Following [34], the Poisson loss of the model can be written as follows:

$$\mathcal{L}(\cdot) = \sum_l (\hat{h}_l^{(NN)} - h_l \cdot \log \hat{h}_l^{(NN)}),$$

where $\hat{h}_l^{(NN)} = \mathbf{z}^{(D)}(\mathbf{x})$ is the output of the NN.

4. Numerical Experiments

In this section, we present numerical experiments to validate the NN-based approach. We run all the tests on a desktop computer with an Intel(R) Core(TM) i9-8950HK CPU @2.90 GHz 16.00 GB RAM. We consider the data provided in [35,36] that contain return time series of assets belonging to several major stock markets across the world, cleaned from errors.

In particular, we focus on the assets that make up the S&P 500 index. We simulate 10 years investment strategies where the investor revises decisions once a year. In our tests, the target sparsity percentage varies in $\{30\%, 35\%, 40\%, 45\%, 50\%, 55\%, 60\%\}$. For each value of target sparsity, we generate a learning sample of 1000 portfolios by applying the procedure described in Section 3 on the full set of S&P 500 assets, with $n = 100$. Then, the overall learning sample contains 7000 portfolios, among which the 75% makes the training set ($L_{train} = 5250$) and the remaining 25% is used as testing set ($L_{test} = 1750$).

In Algorithm 1 we set $\lambda = 1$, $\tau_0 = 10^{-5}$, $\tau_{max} = 0.5$ and $\mu = 1.05$. Iterations are stopped as soon as $\|A\mathbf{u_k} - \mathbf{b}\|_2 \leq tol$ with $tol = 10^{-4}$. We assume that one unit of wealth is invested, so we fix $\xi_{\text{init}} = 1$.

Tests have been performed in MatlabR2020 and R environments. To solve the minimization problem (4) we use the Fast Proximal Gradient method implemented in FOM, a Matlab toolbox containing a collection of first-order methods for solving mainly convex optimization problems [37]. The implementation of the NNs has been carried out by using the Keras package [38].

4.1. Neural Network Calibration

When NNs are employed to approximate functions, the values to be assigned to hyper-parameters such as number of layers, number of units per layer, activation function, and others must be chosen carefully. For example, too many layers and/or units per layer can lead to over-fitting, while few layers and/or units can lead to under-fitting. For this reason, these parameters are typically calibrated on data. We carried out a preliminary analysis to explore how these hyper-parameters affect the performance of the NN models. In particular, we analyzed the role of

- The number of layers (depth) $D \in \mathcal{D} = \{1, 2, 3, 4\}$;

- The number of units per layer $q \in \mathcal{Q} = \{2^4, 2^5, 2^6\}$ (where $q = q_k$ for all the hidden layers $k = 1 \ldots, D$);
- The activation function $\phi \in \mathcal{P} = \{'relu', 'sigmoid', 'tanh'\}$ (where $\phi = \phi_k$ for all the hidden layers $k = 1 \ldots, D$).

In summary, we consider $36 = |\mathcal{D} \times \mathcal{Q} \times \mathcal{P}|$ different NN architectures. Furthermore, the analyses of a single NN training could not be sufficient to make a judgment since it could vary among different training attempts (depending on the initial value of the optimization algorithm and on the random selection of batches of training data to calculate the gradients used in back-propagation). So, we fit each network 10 times and compare the results of the different architectures. In each run, the network weights are recursively adjusted to minimize the Poisson loss between the predicted and the reference values. The models were each fit for 500 epochs, and the model with the best performance during these epochs, measured on the validation set, was used. The validation set consists of the last 5% of the sample, which means that the networks fit 95% of the training examples, and performance was assessed on the remaining 5%.

We first compare the NN architectures analyzing the loss function values on the validation set in the 10 different training attempts. Figure 1 shows the boxplot of the Poisson loss in the validation set for the other architectures.

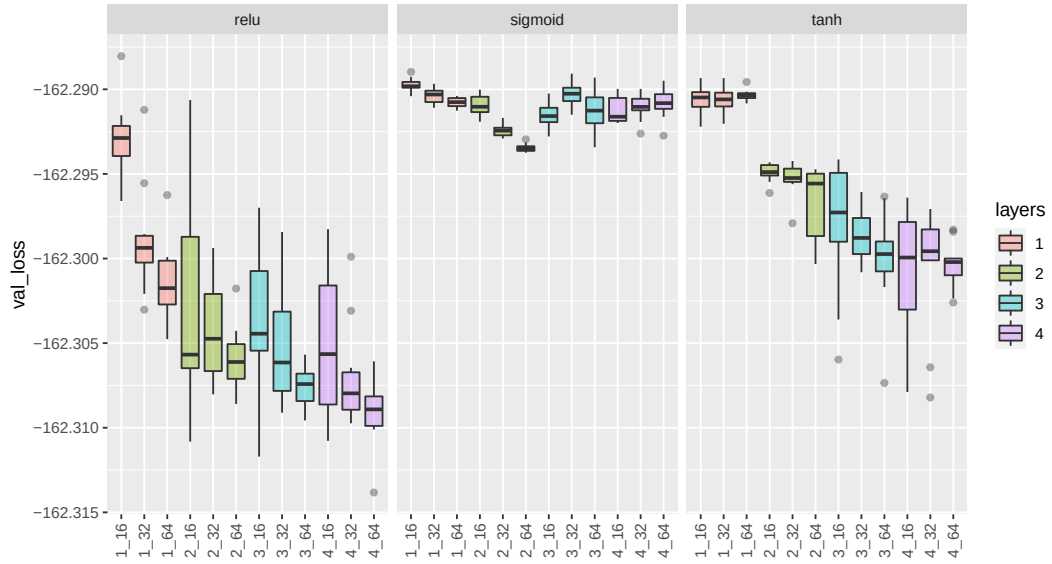

Figure 1. Boxplots of the Poisson loss in the validation set for the different neural network architectures.

We observe that the sigmoid-based architectures present the largest validation loss. This evidence suggests that the sigmoid activation function is less suitable for describing the phenomenon under investigation with respect to the others. Moreover, we observe that considering the relu and tanh architectures, the performance on the validation set improves when the number of layers and units per layer increases. This result indicates that the additional parameters induced by the additional layers and units give more flexibility to the model and improve its out-sample performance. In the experiments shown in the following, we use the network that produces the best performance on the validation set and, therefore, this is expected to be the network with the best out-sample accuracy. In particular, we observe that the best performance validation set can be obtained using a four-layer architecture with 2^6 units per layer and the relu activation function.

4.2. Neural Network-Based Strategy Performance

In this section, we compare the solutions of the portfolio selection problem obtained with Algorithms 2 and 3. This comparison is carried out by considering the values of τ,

the percentage of sparsity in the solution, and the Sharpe Ratio (SR) [39]. The SR is the ratio between the average of the portfolio's expected return and its standard deviation, which measures the risk. Since one would maximize return and minimize risk, great values of Sharpe Ratio are desirable. In the multi-period framework, we compute it in the following way:

$$SR = \frac{\frac{1}{m}\sum_{j=1}^{m} p_j}{\sigma(\mathbf{p})},$$

where $\mathbf{p} = (p_1, ..., p_m)$, and

$$p_j = \mathbf{r}_j^T \frac{\mathbf{u}_j}{\mathbf{u}_j^T \mathbf{1}_n}, \quad j = 1, ..., m.$$

We analyze the out-sample performance of the NN model selected on the test set. First, we investigate the differences between the τ obtained through the I-BIPS strategy and those calculated via NNPS. In Figure 2, the densities of the residual distributions of τ for the testing and training sets are illustrated. The figure shows that both the distributions are centred in zero, thus suggesting that there is not a systematic component in the errors. This behaviour occurs for both the testing and training set.

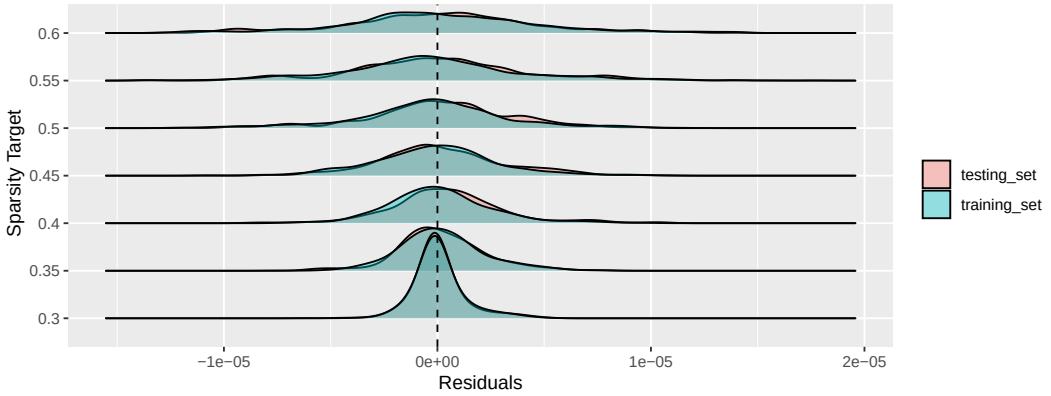

Figure 2. Distributions of the residuals of τ for the testing set (red) and training set (blue).

A graphical comparison of the values of τ is depicted in Figure 3 which compares the value of τ produced by Algorithms 2 and 3 on the testing set. Again, the comparison is carried out for different levels of the target sparsity. According to the theoretical results, the obtained τ increase with the required sparsity for both methods. Moreover, we observe that the distributions have similar positions for all the considered sparsity levels. In this regard, we performed some statistical tests to compare the mean values of the two distributions. We use the paired samples *t*-test, and Table 1 reports the results of the statistics and the respective *p*-values for the different sparsity levels. The null hypothesis is that the true difference between the means of the two distributions is zero, while the alternate hypothesis is that it is different from zero. The *p*-values are rather high in most cases, and we can conclude that the average τ are not significantly different. The only case where the *p*-value appears low is for target sparsity equal to 40%. However, the graphical inspection of the whole τ distributions in Figure 3 and the distributions of the model residuals in Figure 2 do not reveal any particular criticisms also for this level of sparsity.

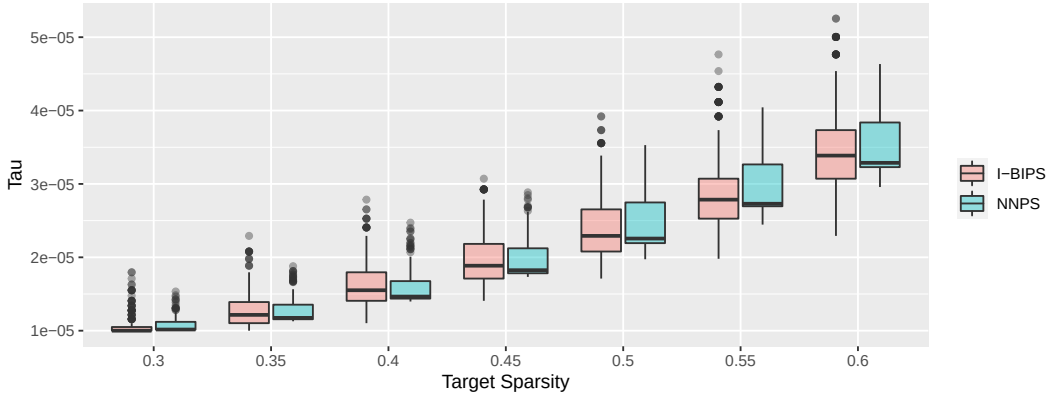

Figure 3. Boxplots of the τ produced by the I-BIPS (red) and the NNPS (blue) methods in the testing set for different sparsity targets.

Table 1. Results of the t-test for the significance of differences between the means of τ distributions for different levels of sparsity.

Target Sparsity	Statistics	p-Value
30%	0.9102	0.3636
35%	1.4477	0.1490
40%	2.8123	0.0053
45%	0.9942	0.3211
50%	0.5310	0.5959
55%	0.2791	0.7804
60%	0.8496	0.3964

In Figure 4, we analyze the percentage of sparsity realized by Algorithm 2 versus the one realized by Algorithm 3. We observe that the boxplots of the two methods have a similar median; however, in some cases, the NNPS method produces a lower level of sparsity with respect to the target sparsity. The boxplots of the two methods show a similar median; however, the variability of the NNPS appears greater than the variability of I-BIPS. We also note that, in some cases, the NNPS method produce sparsities marginally lower than the target one; this happens when the NN underestimates τ_{opt}.

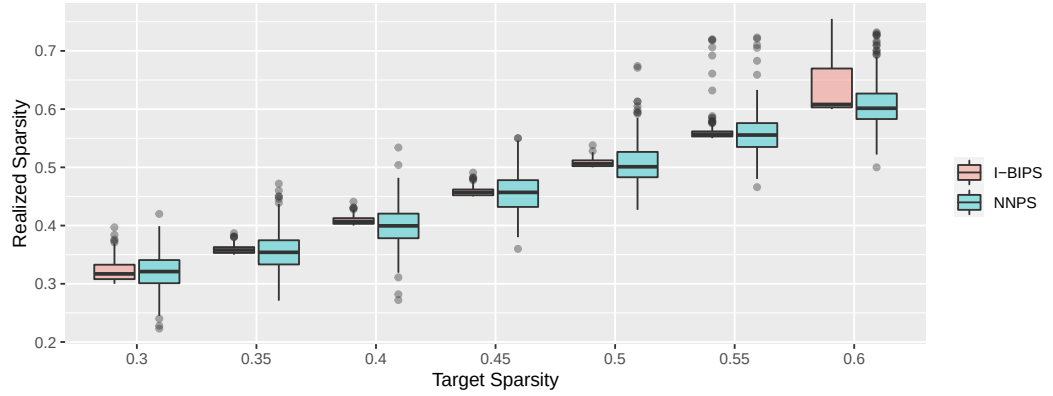

Figure 4. Boxplots of the realized sparsities produced by the I-BIPS (red) and the NNPS (blue) methods in the testing set for different sparsity targets.

Finally, in Figure 5, we compare the Sharpe Ratio of the optimal portfolios produced by the two methods. We observe similar distributions of the Sharpe Ratio for all the values of the target sparsity.

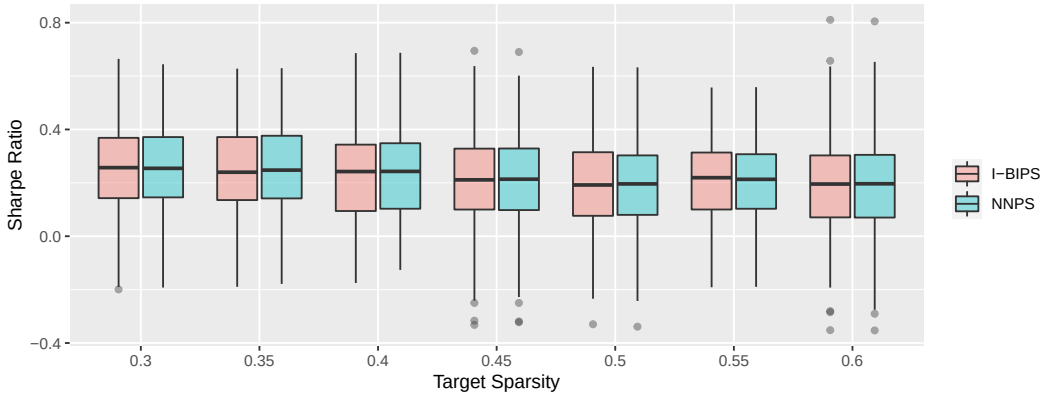

Figure 5. Boxplots of the Sharpe Ratios produced by the I-BIPS (red) and the NNPS (blue) methods in the testing set for different sparsity targets.

4.3. Comparison with the Adaptive Strategy

In this section, we compare the results of the NNPS procedure with those obtained with the A-BIPS one in terms of values of τ, sparsity, Sharpe Ratio and computational cost. Figure 6 shows the boxplots of the τ produced by the two methods for different target sparsity levels in the testing set. The A-BIPS produces some outliers, so we use the logarithmic scale to improve the readability of the Figure.

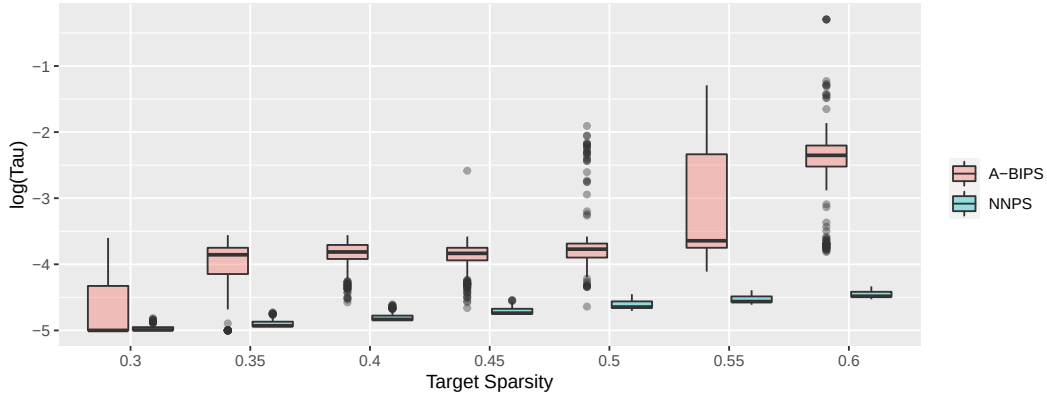

Figure 6. Boxplots of the τ values produced by the A-BIPS (red) and the NNPS (blue) methods in the testing set for different sparsity targets.

Due to the changes of the regularization parameter within the optimization procedure, A-BIPS produces greater values of τ with respect to those obtained by NNPS for the same sparsity target. The difference between the obtained τ seems more significant when required sparsity levels are higher. Larger values of the regularization parameter also induce effects on the realized sparsity. Figure 7 compares the distributions of realized sparsity for different target sparsity levels.

As expected, we observe that the A-BIPS method generally produces more sparse solutions than the NNPS. On the one hand, this effect guarantees the achievement of the required financial target, but, on the other hand, it could penalize the risk diversification. Figure 8, which compares the SR of the two methods, confirms this argument. In particular, we observe that the median SR of the NNPS is always larger than the one resulting from the A-BIPS method for all sparsity degrees. Furthermore, this difference increases when we consider higher target sparsity levels.

This result is further confirmed by Table 2. It lists, for each level of sparsity, the average SR obtained with NNPS (second column) and A-BIPS (third column), and the percentage of cases in which the SR produced by NNPS is greater than the one resulting from A-BIPS. We observe that the average SR of NNPS is always greater than that of A-BIPS for all levels of sparsity. Furthermore, the success rate is above 78%. Table 2 also lists the computational

times required by the two methods to solve the portfolio selection problem. We observe that the NNPS outperforms the A-BIPS method from a computational cost point of view. This result is quite reasonable since NNPS solves the problem by using the τ produced by the neural network, while the A-BIPS approach runs an adaptive approach to identify the optimal τ.

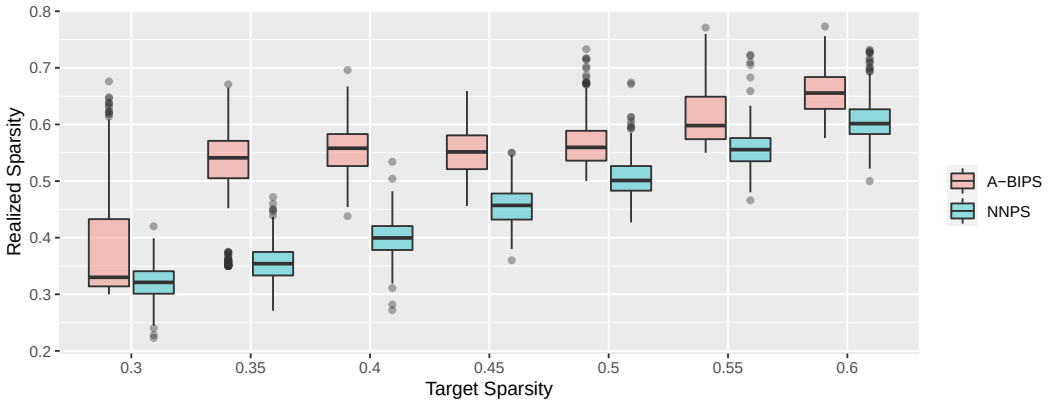

Figure 7. Boxplots of the realized sparsities produced by the A-BIPS (red) and the NNPS (blue) methods in the testing set for different sparsity targets.

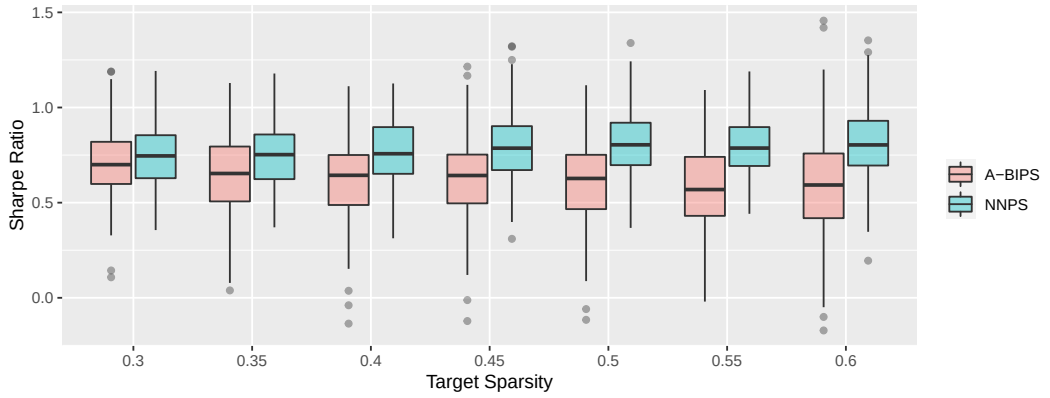

Figure 8. Boxplots of the Sharpe Ratios produced by the A-BIPS (red) and the NNPS (blue) methods in the testing set for different sparsity targets.

Table 2. Comparison in terms of Sharpe ratio and computational time. First column: target sparsity; second column: average SR for NNPS, third column: average SR for A-BIPS, fourth column: percentage of cases in which the NNPS outperforms the A-BIPS in terms of SR; fifth column: average computational time for NNPS; sixth column: average computational time for A-BIPS; seventh column: percentage of cases in which the NNPS outperforms A-BIPS in terms of computational time.

	Sharpe Ratio			**Computational Time (s)**		
Sparsity Target	**NNPS**	**A-BIPS**	**Success**	**NNPS**	**A-BIPS**	**Success**
30%	0.7488	0.7090	78%	1.7532	2.1836	56%
35%	0.7513	0.6428	84%	1.7713	3.2589	90%
40%	0.7672	0.6183	85%	1.7715	3.4991	100%
45%	0.7885	0.6303	82%	1.7726	3.5356	100%
50%	0.8071	0.6053	88%	1.7783	3.6972	100%
55%	0.8027	0.5838	87%	1.7788	4.2314	100%
60%	0.8076	0.5860	86%	1.8772	5.4186	100%

We finally test Algorithm 3 on a different, more recent dataset available on the Kaggle repository (https://www.kaggle.com/ (accessed on 16 January 2022)), which provides the

monthly asset returns of the SP500 assets from January 2017 to December 2021. In this case, $m = 5$ rebalancing dates are considered. We perform the same comparison analysis on Algorithms 2 and 3. The results are reported in Table 3. The two methods are compared in terms of (average) τ, realized sparsity and SR. The results confirm the behaviour observed in the previous tests.

Table 3. Comparison of I-BIPS and NNPS on Kaggle data in terms of τ, realized sparsity and SR.

Target Sparsity	τ		Realized Sparsity		Sharpe Ratio	
	NNPS	I-BIPS	NNPS	I-BIPS	NNPS	I-BIPS
30%	3.2988×10^{-5}	3.3057×10^{-5}	30.5%	30.6%	0.7590	0.7590
35%	4.0874×10^{-5}	4.1454×10^{-5}	35.3%	35.6%	0.7745	0.7747
40%	5.1196×10^{-5}	5.1632×10^{-5}	40.4%	40.6%	0.7699	0.7698
45%	6.3730×10^{-5}	6.3992×10^{-5}	45.5%	45.6%	0.7671	0.7673
50%	8.0339×10^{-5}	7.9435×10^{-5}	50.8%	50.6%	0.7582	0.7581
55%	1.0000×10^{-4}	9.9716×10^{-5}	55.6%	55.5%	0.7670	0.7672
60%	1.2976×10^{-4}	1.3054×10^{-4}	60.4%	60.5%	0.7594	0.7598

5. Conclusions

In this work, we discussed the application of Deep Learning to select the regularization parameter in l_1 regularized portfolio optimization. Preliminary results show the effectiveness of our approach. The NN-based approximation seems to accurately capture the relation between the selected features and the optimal regularization parameter. Optimal portfolios exhibit satisfying financial properties; however, we observed in some cases the underestimate of the optimal regularization parameter, which leads to a sparsity level that is slightly lower than the target one. This issue suggests to explore the use of asymmetric loss function that penalizes more underestimates than overestimates. Moreover, results show that the proposed algorithm often outperforms an existing method for the same problem.

Author Contributions: Conceptualization, S.C., V. De.S., Z.M. and S.S.; Data curation, S.C. and V. De.S.; Investigation, S.C., V. De.S., Z.M. and S.S.; Methodology, S.C., V. De.S., Z.M. and S.S.; Software, S.C., V. De.S., Z.M. and S.S.; Visualization, Z.M. and S.S.; Writing – original draft, S.C., V. De.S., Z.M. and S.S. All the authors contributed to the manuscript equally to all parts of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Università degli Studi di Napoli Parthenope grant number D.R. 831.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data used for the numerical experiments are available on Kaggle and in [35,36].

Acknowledgments: This work was partially supported by INdAM-GNCS Projects, by the VAINHOPES Project, funded by the 2019 V:ALERE (VAnviteLli pEr la RicErca) Program.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ML	Machine Learning
NN	Neural Network
BIPS	Bregman Iteration for Portfolio Selection
I-BIPS	Iterative Bregman Iteration for Portfolio Selection
A-BIPS	Adaptive Bregman Iteration for Portfolio Selection
NNPS	Neural Network for Portfolio Selection
SR	Sharpe Ratio

References

1. Carrasco, M.; Noumon, N. Optimal portfolio selection using regularization. *Citeseer Tech. Rep.* **2011**. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.716.6710&rep=rep1&type=pdf(accessed on 16 January 2022).
2. Brodie, J.; Daubechies, I.; DeMol, C.; Giannone, D.; Loris, I. Sparse and stable Markowitz portfolios. *Proc. Natl. Acad. Sci. USA* **2009**, *30*, 12267–12272.
3. Corsaro, S.; De Simone, V. Adaptive l_1-regularization for short-selling control in portfolio selection. *Comput. Optim. Appl.* **2019**, *72*, 457–478.
4. Corsaro, S.; De Simone, V.; Marino, Z.; Perla, F. Numerical solution of the regularized portfolio selection problem. In *Mathematical and Statistical Methods for Actuarial Sciences and Finance*; Corazza, M., Durbán, M., Grané, A., Perna, C., Sibillo, M., Eds.; Springer International Publishing: New York, NY, USA, 2018; pp. 249–252.
5. Corsaro, S.; De Simone, V.; Marino, Z.; Perla, F. L_1-regularization for multi-period portfolio selection. *Ann. Oper. Res.* **2020**, *294*, 75–86.
6. Culkin, R.; Das, S.R. Machine learning in finance: The case of deep learning for option pricing. *J. Invest. Manag.* **2017**, *15*, 92–100.
7. Dixon, M.F.; Halperin, I.; Bilokon, P. *Machine Learning in Finance*; Springer: Berlin, Germany, 2020.
8. Emerson, S.; Kennedy, R.; O'Shea, L.; O'Brien, J. Trends and applications of machine learning in quantitative finance. In Proceedings of the 8th International Conference on Economics and Finance Research (ICEFR 2019), Lyon, France, 18–21 June 2019.
9. Ghoddusi, H.; Creamer, G.G.; Rafizadeh, N. Machine learning in energy economics and finance: A review. *Energy Econ.* **2019**, *81*, 709–727.
10. Heaton, J.B.; Polson, N.G.; Witte, J.H. Deep learning for finance: Deep portfolios. *Appl. Stoch. Model. Bus. Ind.* **2017**, *33*, 3–12.
11. Chen, W.; Zhang, H.; Mehlawat, M.K.; Jia, L. Mean–variance portfolio optimization using machine learning-based stock price prediction. *Appl. Soft Comput.* **2021**, *100*, 106943.
12. Ma, Y.; Han, R.; Wang, W. Portfolio optimization with return prediction using deep learning and machine learning. *Expert Syst. Appl.* **2021**, *165*, 113973.
13. Paiva, F.D.; Cardoso, R.T.N.; Hanaoka, G.P.; Duarte, W.M. Decision-making for financial trading: A fusion approach of machine learning and portfolio selection. *Expert Syst. Appl.* **2019**, *115*, 635–655.
14. Zhang, Z.; Zohren, S.; Roberts, S. Deep Learning for Portfolio Optimization. *J. Financ. Data Sci.* **2020**, *2*, 8–20.
15. Afkham, B.M.; Chung, J.; Chung, M. Learning Regularization Parameters of Inverse Problems via Deep Neural Networks. *arXiv* **2021**, arXiv:2104.06594.
16. Jiang, Z.; Ji, R.; Chang, K.C. A Machine Learning Integrated Portfolio Rebalance Framework with Risk-Aversion Adjustment. *J. Risk Financ. Manag.* **2020**, *13*, 155.
17. Wang, Q.; Ma, Y.; Zhao, K.; Tian, Y. A Comprehensive Survey of Loss Functions in Machine Learning. *Ann. Data Sci.* **2020**, 1–26, doi:10.1007/s40745-020-00253-5.
18. Goodfellow, I.J.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
19. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed.; Springer: Berlin, Germany, 2009.
20. Cui, X.; Gao, J.; Li, X.; Li, D. Optimal multi-period mean—Variance policy under no-shorting constraint. *Eur. J. Oper. Res.* **2014**, *234*, 459–468.
21. Li, D.; Ng, W. Optimal Dynamic Portfolio Selection: Multiperiod Mean-Variance Formulation. *Math. Financ.* **2000**, *10*, 387–406.
22. Chen, Z.; Li, G.; Guo, J. Optimal investment policy in the time consistent mean–variance formulation. *Insur. Math. Econ.* **2013**, *52*, 145–156.
23. Corsaro, S.; De Simone, V.; Marino, Z. Fused Lasso approach in portfolio selection. *Ann. Oper. Res.* **2021**, *299*, 47–59.
24. Corsaro, S.; De Simone, V.; Marino, Z. Split Bregman iteration for multi-period mean variance portfolio optimization. *Appl. Math. Comput.* **2021**, *392*, 125715.
25. Nocedal, J.; Wright, S.J. *Numerical Optimization*, 2nd ed.; Springer: New York, NY, USA, 2006.
26. Bregman, L. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Comput. Math. Math. Phys.* **1967**, *7*, 200–217.
27. Ho, M.; Sun, Z.; Xin, J. Weighted Elastic Net Penalized Mean-Variance Portfolio Design and Computation. *SIAM J. Financ. Math.* **2015**, *6*, 1220–1244.

28. Goldstein, T.; Osher, S. The split Bregman method for L1-regularized problems. *SIAM J. Imaging Sci.* **2009**, *2*, 323–343, doi:10.1137/080725891.
29. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366.
30. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
32. Montesinos-López, O.A.; Montesinos-López, J.C.; Singh, P.; Lozano-Ramirez, N.; Barrón-López, A.; Montesinos-López, A.; Crossa, J. A multivariate Poisson deep learning model for genomic prediction of count data. *G3 Genes Genomes Genet.* **2020**, *10*, 4177–4190.
33. Gao, G.; Wang, H.; Wüthrich, M.V. Boosting Poisson regression models with telematics car driving data. *Mach. Learn.* **2021**, *111*, 243–272.
34. Fallah, N.; Gu, H.; Mohammad, K.; Seyyedsalehi, S.A.; Nourijelyani, K.; Eshraghian, M.R. Nonlinear Poisson regression using neural networks: A simulation study. *Neural Comput. Appl.* **2009**, *18*, 939–943.
35. Bruni, R.; Cesarone, F.; Scozzari, A.; Tardella, F. Real-world datasets for portfolio selection and solutions of some stochastic dominance portfolio models. *Data Brief* **2016**, *8*, 858–862.
36. Francesco, C.; Luis, M.M.; Alessandra, C. Does ESG Impact Really Enhances Portfolio Profitability? 2022. Available online: Https://ssrn.com/abstract=4007413 (accessed on 16 January 2022).
37. Beck, A.; Guttman-Beck, N. FOM—A MATLAB Toolbox of First Order Methods for Solving Convex Optimization Problems. *Optim. Methods Softw.* **2019**, *34*, 172–193.
38. Chollet, F. Keras: The Python Deep Learning Library; 2018 Available online: https://ui.adsabs.harvard.edu/abs/2018ascl.soft06022C/abstract(accessed on 16 January 2022).
39. Sharpe, W.F. The Sharpe Ratio. *J. Portf. Manag.* **1994**, *21*, 49–58.