國立臺灣大學理學院統計與數據科學研究所

碩士論文

Institute of Statistics and Data Science

College of Science

National Taiwan University

Master's Thesis

使用自助法技術對 LassoNet 模型去偏誤

Bias Reduction in LassoNet Models Using Bootstrap Techniques

劉冠毅

Guan-Yi Liu

指導教授: 楊鈞澔 博士

Advisor: Chun-Hao Yang Ph.D.

中華民國 114 年 2 月

February, 2025

doi:10.6342/NTU202500447

摘要

LassoNet 是一種 Lasso 的擴展模型，其結合了前饋神經網絡以捕捉非線性關係，並同時保留其稀疏解。然而，它跟 Lasso 一樣有偏誤問題，特別是在高維廣義線性模型中。本文中將 LassoNet 擴展到高維廣義線性模型中，使其能夠在維持稀疏性的同時建模複雜的數據結構。然而因為該擴展模型的估計結果仍然與 Lasso 一樣存在偏誤問題，所以我們引入了 van de Geer et al. (2014) 提出的去偏誤架構，並通過基於自助法的校正方法作進一步改進。我們的改進提供了一個具有良好預測性和解釋性的去偏 LassoNet 估計式。我們提出的方法拓展 LassoNet 的應用範圍，並為分析高維數據中預測變量與響應變量之間的非線性和稀疏關係提供了一個可靠的框架。

關鍵字：神經網路、去偏誤、高維度模型、變數選擇、LassoNet

doi:10.6342/NTU202500447

doi:10.6342/NTU202500447

Abstract

LassoNet, an extension of Lasso, incorporates feed-forward neural networks (FFNs) to capture nonlinear relationships while retaining sparse solutions. However, it inherits Lasso's bias issues, particularly in high-dimensional GLMs. In this thesis, we extend LassoNet to GLMs, enabling it to model complex data structures while maintaining sparsity. As the estimations provided by this extended model still exhibit bias similar to Lasso, we address this issue by incorporating the debiasing framework introduced by van de Geer et al. (2014) and further enhancing it with a bootstrap-based correction. These refinements yield a debiased LassoNet estimator that is both predictive and interpretable. The proposed method broadens the applicability of LassoNet, providing a reliable framework for analyzing high-dimensional data with nonlinear and sparse relationships between predictors and response.

Keywords: Neural Networks, Bias Reduction, High-dimensional Models, Variable Selection, LassoNet

doi:10.6342/NTU202500447

doi:10.6342/NTU202500447

Contents

doi:10.6342/NTU202500447

doi:10.6342/NTU202500447

List of Figures

doi:10.6342/NTU202500447

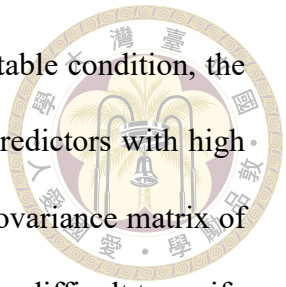
doi:10.6342/NTU202500447

Chapter 1 Introduction

1.1 Background

In recent years, high-dimensional regression models have been widely applied across various domains, particularly in fields where the number of predictors far exceeds the number of observations, such as genomics (Feng and Simon, 2019), finance (Andrews and Lu, 2001), and image analysis (He et al., 2019; Liu et al., 2024). In these scenarios, only a few predictors are informative, while including too many irrelevant predictors can overly complicate the model, leading to overfitting. This has spurred the development of advanced techniques for variable selection, aiming to enhance model interpretability and improve predictive accuracy. Until now, numerous methods for variable selection in linear models have been developed, including stepwise selection, screening, and penalty-based approaches. Among these, the most classical penalty-based method is Lasso (Tibshirani, 1996), which has become a widely recognized approach, particularly in large p smaller n scenarios. The primary advantage of Lasso lies in its ability to efficiently achieve feature sparsity while simultaneously estimating coefficients by incorporating an ℓ_1-regularization term into the loss function. Since the success of variable selection via Lasso, numerous Lasso-based methods and related studies have been developed, such as the *Adaptive Lasso* (Zou, 2006), the *Dantzig Selector* (Candes and Tao, 2007), and the *Elastic Net* (Zou and

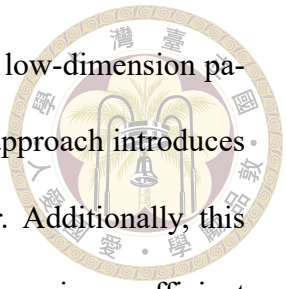
doi:10.6342/NTU202500447



Hastie, 2005), among others. It is well known that, under irrepresentable condition, the nonzero coefficients selected by Lasso are consistent with the true predictors with high probability. However, this condition primarily relies on the sample covariance matrix of the predictors satisfying certain structural assumptions, which are often difficult to verify in practice (Zhao and Yu, 2006). Despite these challenges, Lasso has provided a solid foundation for subsequent research and improvements in variable selection techniques.

In the aforementioned Lasso-based variable selection methods, the focus has been primarily on standard linear models, which assume a linear relationship between the response and predictors. Lemhadri et al. (2021) proposed LassoNet, which additionally incorporates a feed-forward neural network (FFN) to model the nonlinear effects. Although a nonlinear term is introduced into the model, they successfully mimic the Lasso framework to achieve sparse solutions and simultaneous estimation. LassoNet shares the same primary limitation as Lasso. Its inherent estimation bias is caused by the ℓ_1 -regularization term, which shrinks all coefficients, including the true predictors, toward zero. This bias not only reduces the accuracy of prediction performance but also complicates the interpretation of those nonzero coefficients. Luckily, many debiasing methods have been developed for the Lasso estimator. Since the distribution of the Lasso estimator is not tractable, a direct and standard way to debias the regression estimator is bootstrap (Efron, 1992). However, Lahiri (2010) demonstrated that the residual bootstrap approximation for Lasso is inconsistent, making it difficult to draw valid inferences directly from the Lasso estimates. To address this issue, Chatterjee and Lahiri (2011) proposed a modified bootstrap Lasso estimator, which mitigates the inconsistency problem by employing a thresholding technique. On the other hand, in the large p small n scenario, Zhang and Zhang (2014) proposed a low-dimension projection estimator (LDP), also known as ”de-biased Lasso”.

doi:10.6342/NTU202500447



This estimator is an asymptotically Gaussian-distributed estimator of low-dimension parameters in the high-dimensional linear regression model. The LDP approach introduces a correction score designed to address the bias in the Lasso estimator. Additionally, this approach can be used for variable selection, to estimate the entire regression coefficient vector, and to construct confidence intervals to quantify the uncertainty of the estimator.

Following the research line of Zhang and Zhang (2014), van de Geer et al. (2014) approached the problem from a different perspective by starting with the Karush–Kuhn–Tucker (KKT) conditions (Boyd and Vandenberghe, 2004) for the Lasso estimator. They reformulated these conditions into a form of the Lasso estimator plus a bias term. In this framework, approximating the inverse of the sample covariance matrix became necessary, and the authors successfully addressed this using node-wise Lasso regression (Meinshausen and Bühlmann, 2006). Furthermore, they extended their debiasing approach to generalized linear models (GLMs) with convex loss functions, providing a broader framework for bias correction. Building on this work, Xia et al. (2023) further addressed the limitations of the sparsity assumption on the inverse of the information matrix in GLM settings. They proposed a refined debiased Lasso estimator designed for the "large n, diverging p" scenario. On the other hand, Li (2020) took an approach more closely aligned with Zhang and Zhang (2014), proposing a modified correction score to reduce the bias of the modified debiased Lasso estimator through a bootstrap method. A notable advantage of their correction is that it does not rely on the irrepresentable condition, making it particularly useful in high-dimensional settings.

doi:10.6342/NTU202500447

1.2 Proposed method

In the GLM framework, the link function primarily captures the linear relationship between predictor x and response y, neglecting potential nonlinear effects. To address this limitation, this thesis extends LassoNet to GLMs to model more complex structures, while maintaining the ability to obtain sparse solutions in high-dimensional settings, similar to Lasso in GLMs. However, like Lasso in GLMs, this extension also results in a biased solution. To overcome this, we incorporate the debiasing method proposed by van de Geer et al. (2014) into this extension, achieving a debiased LassoNet estimator. Furthermore, we further reduced the bias of this debiased estimator through a bootstrap approach. With these refinements, we obtained a model that is not only predictable but also interpretable and reliable.

1.3 Organization of the thesis

The rest of the thesis is organized as follows. In Chapter 2, we introduce some preliminary concepts necessary for understanding the LassoNet model, including an overview of Lasso, the framework of fully connected neural networks, and the bootstrap procedure for estimation debiasing. In Chapter 3, we provide a detailed description of the LassoNet model, its extension to GLMs, and the exact debiasing procedure. In Chapter 4, we present the empirical results and compare our approach with other existing methods. Finally, we discuss our findings and summarize the main contributions of this work in Chapter 5.

doi:10.6342/NTU202500447

Chapter 2 Preliminary

In this chapter, we introduce some preliminary knowledge relevant to the LassoNet model, including Lasso, fully-connected neural networks, and the framework of estimation debiasing with bootstrap. In Section 2.1, we present the ℓ_1-regularization function, provide intuition to explain why the Lasso optimization leads to sparse solutions, and discuss the associated challenges and limitations of Lasso. In Section 2.2, we describe the framework of neural network models, the process of training these models, and explore some common network-based architectures, such as recurrent neural networks and convolutional neural networks.

2.1 Lasso

As mentioned in the introduction, Lasso is a regularization method designed to address overfitting in models. When considering a linear regression model:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2\mathbb{I}), \tag{2.1}$$

where $\boldsymbol{Y} = (y_1, \cdots, y_n) \in \mathbb{R}^n$ is the response vector, $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_p}] \in \mathbb{R}^{n\times p}$ is the design matrix with column vectors $\boldsymbol{x_i}$, and $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_p) \in \mathbb{R}^p$ represents the unknown regression coefficients. Based on the residual sum of squares criterion, the

doi:10.6342/NTU202500447

Lasso solution is written as:

$$\widehat{\boldsymbol{\beta}}^{Lasso} = \arg\min_{\boldsymbol{\beta}} \{ \frac{1}{2n} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1 \}, \tag{2.2}$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm, and λ is a regularization parameter controlling the sparsity of the Lasso estimator. Specifically, when λ increases, more components in $\widehat{\boldsymbol{\beta}}^{Lasso}$ become zero; conversely, a smaller λ results in fewer zero components. In extreme cases, when $\lambda = 0$, the Lasso estimator reduces to the ordinary least squares (OLS) solution, whereas as $\lambda \to \infty$, it converges to a zero estimator. In fact, (2.2) is equivalent to the following Lagrangian form

$$\begin{aligned} \widehat{\boldsymbol{\beta}}^{Lasso} = \quad & \arg\min_{\boldsymbol{\beta}} \{ \frac{1}{2n} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) \} \\ & \text{subject to} \quad \|\boldsymbol{\beta}\|_1 \leq t \end{aligned} \tag{2.3}$$

where t is a non-negative tuning parameter. An intuitive way to understand why the Lasso can obtain a sparse solution is as follows. When considering the case $p = 2$ the constraint region for the Lasso is a diamond-shaped region defined by $|\beta_1| + |\beta_2| \leq t$, which has four corners. Meanwhile, the residual sum of squares has elliptical contours. When the solution lies at a corner of the diamond, it results in one of the parameters β_j being zero. As p increases, the diamond generalizes to a higher-dimensional rhomboid with more corners, providing more opportunities for additional $\beta_j' s$ to become zero.

2.2 Fully-Connected Neural Network

In this section, we introduce one of the main structures in the LassoNet model: the fully-connected neural network (FCNN) (Hastie et al., 2009). A neural network, in

doi:10.6342/NTU202500447

essence, is a regression model constructed by stacking multiple hidden layers. Assume the input, also called the covariate, is $\boldsymbol{x} = (x_1, \cdots, x_p)^T \in \mathbb{R}^p$, and the response is $y \in \mathbb{R}$. Given an activation function $\sigma_d : \mathbb{R} \to \mathbb{R}$ and the number of neurons in the $(d-1)^{th}$ and d^{th} layers as q_{d-1} and q_d, respectively, the dth layer of a deep FCNN is defined by the mapping $\boldsymbol{z} : \mathbb{R}^{q_{d-1}} \to \mathbb{R}^{q_d}$:

$$\boldsymbol{r} \mapsto \boldsymbol{z}^{(d)} = (z_1^{(d)}(\boldsymbol{r}), \cdots, z_{q_d}^{(d)}(\boldsymbol{r}))^T, \tag{2.4}$$

where $z_j^{(d)}(\boldsymbol{r})$, for $j = 1, \cdots, q_d$, are the neurons in the d^{th} layer, and $\boldsymbol{r} = (r_1, \cdots, r_{q_{d-1}})^T \in \mathbb{R}^{q_{d-1}}$ is the output variable from the $(d-1)^{th}$ layer:

$$z_j^{(d)}(\boldsymbol{r}) = \sigma_d\left(w_{0,j}^{(d)} + \langle \boldsymbol{w}_j^d, \boldsymbol{r}\rangle\right) = \sigma_d\left(w_{0,j}^{(d)} + \sum_{\ell=1}^{q_{d-1}} w_{\ell,j}^{(d)} r_\ell\right), \tag{2.5}$$

where $\boldsymbol{w}_j^{(d)} = (w_{1,j}^{(d)}, \cdots, w_{q_{d-1},j}^{(d)})^T \in \mathbb{R}^{q_{d-1}}$ are the unknown network weights, and $w_{0,j}^{(d)} \in \mathbb{R}$ is the bias term. A depth-D, $D \in \mathbb{N}$, FCNN is obtained by composing D layers as in (2.5). When the input $\boldsymbol{x}$ is given, a depth-D FCNN can be expressed as:

$$g_{\boldsymbol{w}}^{(D)}(\boldsymbol{x}) \equiv \left(\boldsymbol{z}^{(D)} \circ \boldsymbol{z}^{(D-1)} \circ \cdots \circ \boldsymbol{z}^{(1)}\right)(\boldsymbol{x}), \tag{2.6}$$

where the Dth layer is the output layer, this network structure models the underlying relationship between y and $\boldsymbol{x}$. Common nonlinear activation functions include the sigmoid function, $\sigma_{\text{sigmoid}}(u) = \frac{1}{1+e^{-u}}$, and the Rectified Linear Unit (ReLU), $\sigma_{\text{ReLU}}(u) = \max(0, u)$, which enhance model complexity and help mitigate underfitting.

Specifically, if the response variable is continuous, there will be only one neuron in the output layer i.e. $q_d = 1$, and the identity function is typically chosen as the activation function for the output layer. On the other hand, when considering a classification problem

doi:10.6342/NTU202500447

with K classes, there will be $q_d = K$ neurons in the output layer. In this case, the network often employs a softmax activation function at the output layer to normalize the outputs into probabilities:

$$\sigma_{\text{softmax}}(u_k) = \frac{e^{u_k}}{\sum_{j=1}^{K} e^{u_j}}, \quad k = 1, \cdots, K, \tag{2.7}$$

where $u_k = w_{0,k}^{(D)} + \langle \boldsymbol{w}_k^D, \boldsymbol{z}^{D-1}(\boldsymbol{x}) \rangle$.

Typically, training the FCNN involves solving an optimization problem to minimize the loss function. A widely used algorithm for optimizing the loss function is gradient descent. For a training set $\{\boldsymbol{x}_i, y_i\}_{i=1}^n, \boldsymbol{x}_i \in \mathbb{R}^p, y_i \in \mathbb{R}$ following the empirical risk minimization criterion and considering a classification problem as an example, we minimize the empirical loss function:

$$L(\boldsymbol{w}) = -\frac{1}{n} \sum_{i=1}^{n} \left(y_i \log \left(g_{\boldsymbol{w}}^{(D)}(\boldsymbol{x}_i) \right) \right) \tag{2.8}$$

using the gradient descent algorithm outlined in Algorithm 1.

Algorithm 1 Gradient Descent Algorithm

Require: Initial weight $\boldsymbol{w}^0$, learning rate $\eta > 0$

$k \leftarrow 0$

while $\boldsymbol{w}^{(k)}$ not covered **do**

$k \leftarrow k + 1$

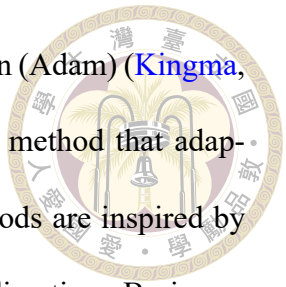
$\Delta \boldsymbol{w}^{(k)} = -\nabla L\left(\boldsymbol{w}^{(k)}\right)$

$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \eta \Delta \boldsymbol{w}^{(k)}$

end while

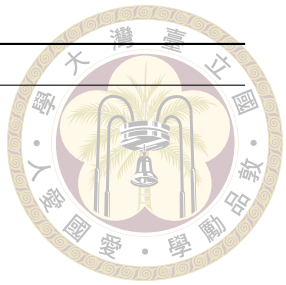
However, due to the high dimensionality of $L(\boldsymbol{w})$, the gradient may converge slowly or even get trapped in suboptimal regions, making optimization inefficient. To address this

doi:10.6342/NTU202500447



issue, more advanced algorithms, such as Adaptive Moment Estimation (Adam) (Kingma, 2014), have been developed. Adam is a momentum-based learning method that adaptively adjusts the learning rate for each parameter. Momentum methods are inspired by physical analogies, where the velocity determines the momentum's direction. By incorporating a momentum term, the algorithm accumulates past gradients' effects, enabling faster convergence when the gradients consistently point in the same direction. Additionally, momentum helps the optimization escape shallow local minima or flat regions by counteracting oscillations when gradients point in opposing directions. Adam further improves on momentum-based methods by introducing adaptive learning rates for each parameter. It computes moving averages of the first and second moments of the gradient, enabling it to scale the updates dynamically. This adaptive behavior makes Adam particularly effective in high-dimensional and sparse settings, where the scale of gradients can vary significantly. For these reasons, Adam has become a standard choice for training neural networks, including LassoNet.

doi:10.6342/NTU202500447



Algorithm 2 Adam Algorithm

Require: Global learning rate $\boldsymbol{\eta}$, decay rates $\rho_1, \rho_2 \in [0, 1)$

Require: Small constant $\boldsymbol{\epsilon}$ for numerical stable

Split the dataset into B batches with batch size m

$k \leftarrow 0$

Initialize $\boldsymbol{w}_B^{(k)}$, $\boldsymbol{s} = 0, \boldsymbol{r} = 0$

while $\boldsymbol{w}_B^{(k)}$ not converged **do**

$k \leftarrow k + 1$

for $b = 1, \cdots, B$ **do**

$\boldsymbol{G} = \frac{1}{|B_b|} \nabla \sum_{j \in B_b} \left(y_j \log \left(g^{(D)}_{\boldsymbol{w}_b^{(k)}}(\boldsymbol{x}_j) \right) \right)$ ▷ Compute the gradient

$\boldsymbol{s} = \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{G}$ ▷ update biased first moment estimate

$\boldsymbol{r} = \rho_2 \boldsymbol{r} + (1 - \rho_1)\boldsymbol{G} \odot \boldsymbol{G}$ ▷ update biased second moment estimate

$\widehat{\boldsymbol{s}} = \frac{\boldsymbol{s}}{1-\rho_1}$ ▷ correct bias in first moment

$\widehat{\boldsymbol{r}} = \frac{\boldsymbol{r}}{1-\rho_2}$ ▷ correct bias in second moment

$\Delta \boldsymbol{w} = -\frac{\boldsymbol{\eta}}{\sqrt{\widehat{\boldsymbol{r}}} + \boldsymbol{\epsilon}} \widehat{\boldsymbol{s}}$

$\boldsymbol{w}_{b+1}^{(k)} + \Delta \boldsymbol{w}$

end for

end while

Note: $\odot$ denotes element-wise multiplication

To conclude, while we have introduced gradient descent and Adam as two key optimization algorithms for training FCNNs, many other algorithms are available depending on the specific needs of the task and the user's familiarity with the methods. Readers are encouraged to select the algorithm that aligns best with their understanding and the requirements of their application. In addition to FCNNs, other neural network architectures, such as recurrent neural networks (RNNs) (Chen and Li, 2021) and convolutional neural networks (CNNs) (Alzubaidi et al., 2021), are widely used for tasks involving sequential data or image data, respectively. These architectures introduce unique structures and train-

doi:10.6342/NTU202500447

ing procedures tailored to their specific applications. However, since our research does not involve RNNs or CNNs, we do not provide further discussion on these architectures here.

doi:10.6342/NTU202500447

doi:10.6342/NTU202500447

Chapter 3 Method

In Section 3.1, we introduce the structure of the LassoNet model and provide an explanation of how it successfully achieves sparse solutions. In Section 3.2 we provide the debiasing procedure van de Geer et al. (2014) proposed for high-dimensional GLMs. In Section 3.3 we provide the procedure of debiasing the LassoNet model.

3.1 Lassonet

First, we consider a residual feed-forward neural network (FFN):

$$\mathcal{H} = \{h : h_{\boldsymbol{\beta},\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{\beta}'\boldsymbol{x} + g_{\boldsymbol{w}}^{(D)}(\boldsymbol{x})\},$$

where $g_{\boldsymbol{w}}^{(D)}$ was defined in Section 2.2, $\boldsymbol{\beta} \in \mathbb{R}^p$ and $\boldsymbol{x} \in \mathbb{R}^p$. This structure models both the linear and non-linear effects between the predictor $\boldsymbol{x}$ and the response $y \in \mathbb{R}$ simultaneously. The linear component typically enhances interpretability, while the non-linear part captures complex relationships. In sparse high-dimensional linear models, Lasso (Tibshirani, 1996) is a commonly used tool for selecting predictive regressors. Similarly, within the FFN framework described above, applying LassoNet (Lemhadri et al., 2021) enables variable selection and estimation simultaneously. Given n training observations and an input design matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_p] \in \mathbb{R}^{n\times p}$, with K neurons in the first hidden

doi:10.6342/NTU202500447

layer, the LassoNet optimization function is defined by

$$\begin{aligned} \min_{\boldsymbol{\beta},\boldsymbol{w}} \quad & L(\boldsymbol{\beta},\boldsymbol{w}) + \lambda\|\boldsymbol{\beta}\|_1 \\ \text{subject to} \quad & \|\boldsymbol{w}_k^{(1)}\|_\infty \le M\|\beta_k\|, \quad k = 1,\cdots,p \end{aligned} \tag{3.1}$$

where λ and $M > 0$ are hyper parameters, $L(\boldsymbol{\beta},\boldsymbol{w})$ is the loss function, and $\boldsymbol{w}_k^{(1)}$ denotes the weights corresponding to the k-th predictor in the first hidden layer. In fact, based on the constraint above, we can observe that traditional Lasso is a special case of this model. When $M = 0$, the k-th regressor does not pass through the network, reducing the model to standard Lasso. Conversely, as M approaches infinity, the model behaves exactly as a FNN. Thus, M serves as a balancing parameter, adjusting the weights between linearity and nonlinearity within the model.

3.2 Debiasing Lasso Estimators

3.2.1 Node-wise Lasso regression

We present the procedure for debiasing the Lasso estimator proposed by van de Geer et al. (2014). Here, we start with the case of linear regression. Considering the linear regression model in (2.1), the Lasso estimator in (2.2) satisfies the KKT stationarity condition:

$$\frac{-1}{n}\boldsymbol{X}^T\left(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}^{Lasso}\right) + \lambda\widehat{\boldsymbol{s}} = 0, \tag{3.2}$$

where $\|\widehat{\boldsymbol{s}}\|_\infty \le 1$ and $\widehat{s}_j = sign\left(\widehat{\beta}_j^{Lasso}\right)$ if $\widehat{\beta}_j^{Lasso} \neq 0$. We can also represent (3.2) with the notation $\widehat{\Sigma} = \boldsymbol{X}^T\boldsymbol{X}/n$ as:

$$\widehat{\Sigma}\left(\widehat{\boldsymbol{\beta}}^{Lasso} - \boldsymbol{\beta}\right) + \lambda\widehat{\boldsymbol{s}} = \boldsymbol{X}^T\boldsymbol{\epsilon}/n \tag{3.3}$$

doi:10.6342/NTU202500447

The main idea in their procedure is to approximate the inverse of $\widehat{\Sigma}$ using node-wise Lasso, denoted as $\widehat{\Omega}$. Incorporating this approximation into (3.3), they derive:

$$\widehat{\boldsymbol{\beta}}^{Lasso} + \widehat{\Omega}\lambda\widehat{\boldsymbol{s}} - \boldsymbol{\beta} = \widehat{\Omega}\boldsymbol{X}^T\boldsymbol{\epsilon}/n - \delta/\sqrt{n}, \tag{3.4}$$

where $\delta = \sqrt{n}\left(\widehat{\Omega}\widehat{\Sigma} - \mathbb{I}\right)\left(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}\right)$. In the same paper, it is proven that δ is asymptotically negligible under certain conditions. Thus, using (3.4) and (3.2) the following debiased estimator is suggested:

$$\widehat{\boldsymbol{\beta}}^{DB-Lasso} = \widehat{\boldsymbol{\beta}}^{Lasso} + \widehat{\Omega}\lambda\widehat{\boldsymbol{s}} = \widehat{\boldsymbol{\beta}}^{Lasso} + \widehat{\Omega}\boldsymbol{X}^T\left(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}\right)/n \tag{3.5}$$

To construct $\widehat{\Omega}$ for the debiasing procedure, we rely on node-wise Lasso. This involves treating each covariate as a response variable and regressing it on the remaining covariates using Lasso below, we outline the detailed steps for calculating $\widehat{\Omega}$: For the design matrix $\boldsymbol{X}$, let $\boldsymbol{x}_j$ denote the j^{th} column and let $\boldsymbol{X}_{-j}$ denote the submatrix excluding the j^{th} column.

- Calculate the Lasso regression coefficients for each covariate:

  $\widehat{\boldsymbol{\alpha}}_j \equiv \underset{\boldsymbol{\alpha}\in\mathbb{R}^{p-1}}{\arg\min}\{\frac{1}{n}\left(\boldsymbol{x}_j - \boldsymbol{X}_{-j}\boldsymbol{\alpha}\right)^T\left(\boldsymbol{x}_j - \boldsymbol{X}_{-j}\boldsymbol{\alpha}\right) + 2\lambda_j\|\boldsymbol{\alpha}\|_1\}$, for $j = 1, \cdots, p$

  with components of $\widehat{\boldsymbol{\alpha}}_j = \{\widehat{\alpha}_{j,k}\ ;\ k = 1, \cdots, p, k \neq j\}$

- $\widehat{\gamma}_j^2 \equiv \left(\boldsymbol{x}_j - \boldsymbol{X}_{-j}\widehat{\boldsymbol{\alpha}}_j\right)^T\left(\boldsymbol{x}_j - \boldsymbol{X}_{-j}\widehat{\boldsymbol{\alpha}}_j\right)/n + \lambda_j\|\widehat{\boldsymbol{\alpha}}_j\|_1$

We then define $\widehat{\Omega}$ as:

$$\widehat{\Omega} = \widehat{\Gamma}^{-2}\widehat{A}$$

where

$$\widehat{\Gamma}^2 \equiv diag\left(\widehat{\gamma}_1^2, \cdots, \widehat{\gamma}_p^2\right) \tag{3.6}$$

doi:10.6342/NTU202500447

and

$$\widehat{A} \equiv \begin{bmatrix} 1 & -\widehat{\alpha}_{1,2} & \cdots & -\widehat{\alpha}_{1,p} \\ -\widehat{\alpha}_{2,1} & 1 & \cdots & -\widehat{\alpha}_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ -\widehat{\alpha}_{p,1} & -\widehat{\alpha}_{p,2} & \cdots & 1 \end{bmatrix} \tag{3.7}$$

Based on these steps, they successfully construct the debiased Lasso estimator in the case of linear model.

3.2.2 Extensions of node-wise Lasso regression

Having established the Node-wise Lasso procedure for linear regression, we now explore its extension to GLMs. Specifically, the procedure is extended to strictly convex loss functions. For the data set $\{\boldsymbol{x}_i, y_i\}_{i=1}^n, \boldsymbol{x}_i \in \mathbb{R}^p, y_i \in \mathbb{R}$ we let $\rho_{\boldsymbol{\beta}}(\boldsymbol{x}_i, y_i) = \rho\left(\boldsymbol{\beta}^T \boldsymbol{x}_i, y_i\right)$ be a convex loss function in $\boldsymbol{\beta} \in \mathbb{R}^p$ and define. The first derivative and the Hessian matrix of $\rho_{\boldsymbol{\beta}}$ are defined as:

$$\rho_{\boldsymbol{\beta}}^{(1)} = \frac{\partial}{\partial \boldsymbol{\beta}} \rho_{\boldsymbol{\beta}}, \quad \rho_{\boldsymbol{\beta}}^{(2)} = \frac{\partial}{\partial \boldsymbol{\beta}\, \partial \boldsymbol{\beta}^T} \rho_{\boldsymbol{\beta}}$$

Moreover, for a given function g, the empirical loss is defined as:

$$L_g(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} g(\boldsymbol{x}_i, y_i)$$

and the corresponding ℓ_1-regularized estimator is written as:

$$\widehat{\boldsymbol{\beta}}^{Lasso} = \arg\min_{\boldsymbol{\beta}} \{L_g(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1\} \tag{3.8}$$

doi:10.6342/NTU202500447

The general debiased estimator for this extended version is then defined as

$$\widehat{\boldsymbol{\beta}}^{DB-Lasso} = \widehat{\boldsymbol{\beta}}^{Lasso} - \widehat{\Omega} L^{(1)}_{\rho_{\widehat{\boldsymbol{\beta}}^{Lasso}}} \left(\widehat{\boldsymbol{\beta}}^{Lasso}\right) \tag{3.9}$$

To compute $\widehat{\Omega}$ we first define the input matrix as

$$\widehat{\Sigma} \equiv L^{(2)}_{\rho_{\widehat{\boldsymbol{\beta}}^{Lasso}}} \left(\widehat{\boldsymbol{\beta}}^{Lasso}\right)$$

Given this input matrix, the node-wise Lasso procedure is applied as follows. For each $j = 1, \cdots, p$ we solve:

$$\widehat{\boldsymbol{\alpha}}_j = \underset{\boldsymbol{\alpha} \in \mathbb{R}^{p-1}}{\arg\min} \{\widehat{\Sigma}_{j,j} - 2\widehat{\Sigma}_{j,\setminus j}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T \widehat{\Sigma}_{\setminus j,\setminus j}\boldsymbol{\alpha} + 2\lambda\|\boldsymbol{\alpha}\|_1\} \tag{3.10}$$

where $\widehat{\Sigma}_{j,\setminus j}$ denotes j^{th} row of $\widehat{\Sigma}$ excluding the j^{th} element, and $\widehat{\Sigma}_{\setminus j,\setminus j}$ is the submatrix of $\widehat{\Sigma}$ obtained by removing the j^{th} row and column. Next, we calculate:

$$\widehat{\gamma}_j = \widehat{\Sigma}_{j,j} - \widehat{\Sigma}_{j,\setminus j}\widehat{\alpha}_j, \quad j = 1, \cdots, p \tag{3.11}$$

to construct $\widehat{\Omega} = \widehat{\Gamma}^{-2}\widehat{A}$, where $\widehat{\Gamma}$ and $\widehat{A}$ are defined as in (3.6) and (3.7) respectively.

3.3 Debiasing the LassoNet

In this section, we provide further details on how we improve the statistical inference of LassoNet. The debiasing methods discussed above primarily consider GLM, where the link function connects only the linear predictor to the response. However, in many cases, the relationship between predictors and responses is more complex.

doi:10.6342/NTU202500447

3.3.1 Merge LassoNet and GLM

Here we extend the LassoNet to GLM. Specifically, for any GLM with a link function link $\ell(\cdot)$ we incorporate a feed-forward network (FFN) $g_{\boldsymbol{w}}$ to model the mean of the response, such that $E(\boldsymbol{y}|\boldsymbol{x}) = \ell^{-1}(\beta'\boldsymbol{x} + g_{\boldsymbol{w}}(\boldsymbol{x}))$. For example, consider logistic regression with the canonical logit link, which can be formulated as:

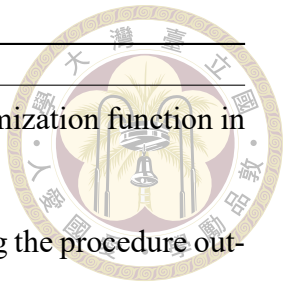
$$y_i \sim Bin(1, logit^{-1}(\beta'\boldsymbol{x}_i + g_{\boldsymbol{w}}(\boldsymbol{x}_i))), \quad i = 1, \cdots, n \tag{3.12}$$

In this setting, we can use the optimization function in (3.1) with the negative log-likelihood of (3.12) as the loss function to obtain a sparse model. This approach allows us to retain the interpretability benefits of traditional GLMs while adapting to more complex data structures.

3.3.2 Bootstraping the Bias of LassoNet

Although a sparse model can be obtained via (3.1), like most regularization methods, suffers from estimation bias. In other words, the estimator produced by LassoNet is not unbiased. Our simulation experiments (4.1) also demonstrate that the LassoNet estimator exhibits a larger bias compared to Lasso. To correct the bias, we adopt a bootstrap strategy inspired by Li (2020), who proposed a method for bootstrapping the bias of a noiseless Lasso estimator. While Li (2020) utilized a noiseless estimator for a standard linear model, we extend this idea to LassoNet. Specifically, we first obtain a noiseless estimator for LassoNet using node-wise regression and then bootstrap the bias of this noiseless estimator. The detailed debiasing procedure is as follows:

doi:10.6342/NTU202500447



Algorithm 3 Procedure for Bootstrap Debiased LassoNet

1: Obtain the initial estimator $\widehat{\boldsymbol{\beta}}$ by training LassoNet using the optimization function in (3.1).

2: Let $\widehat{\boldsymbol{\beta}}^{DB-Net}$ denote the noiseless estimator, obtained by following the procedure outlined in Section 3.2.

3: Define $\boldsymbol{X}$ as the design matrix. Resample the rows of $\boldsymbol{X}$ with replacement B times to create bootstrap design matrices $\boldsymbol{X}_b^*, \quad b = 1, \ldots, B$.

4: Generate the bootstrap response $\boldsymbol{Y}_b^*$ from an exponential family distribution denoted as $f_{\boldsymbol{Y}_b^*|\boldsymbol{X}_b^*}(\boldsymbol{y}_b^*|\boldsymbol{X}_b^*, \widehat{\boldsymbol{\beta}}^{DB-Net})$ with canonical link function $\ell(\cdot)$ satisfying $E(\boldsymbol{Y}_b^*|\boldsymbol{X}_b^*) = \ell^{-1}(\boldsymbol{X}_b^*\widehat{\boldsymbol{\beta}}^{DB-Net})$.

5: Obtain the bootstrap Lasso estimator $\widehat{\boldsymbol{\beta}}_b^*$ by minimizing the negative log-likelihood with ℓ_1 penalty:

$$\widehat{\boldsymbol{\beta}}_b^* = \arg\min_{\boldsymbol{\beta}} \{-\log\{f_{\boldsymbol{Y}_b^*|\boldsymbol{X}_b^*}(\boldsymbol{y}_b^*|\boldsymbol{X}_b^*, \boldsymbol{\beta})\} + \lambda\|\boldsymbol{\beta}\|_1\},$$

where λ is a tuning parameter.

6: Estimate the bias $\widehat{b}$ of $\widehat{\boldsymbol{\beta}}^{DB-Net}$ as

$$\widehat{b} = \frac{1}{B}\sum_{b=1}^{B}(\widehat{\boldsymbol{\beta}}_b^* - \widehat{\boldsymbol{\beta}}^{DB-Net}),$$

and define the bootstrap debiased LassoNet estimator as

$$\widehat{\boldsymbol{\beta}}^{BS-DB-Net} = \widehat{\boldsymbol{\beta}}^{DB-Net} - \widehat{b}.$$

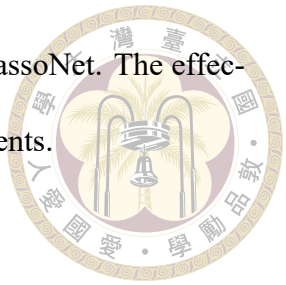
7: Construct a two-sided $100 \times (1-\alpha)\%$ confidence interval for β_j as

$$(q_{\alpha/2}(\widehat{\beta}_{(j)}^*), q_{1-\alpha/2}(\widehat{\beta}_{(j)}^*)),$$

where $q_\alpha(\cdot)$ is the α-quantile, $\widehat{\beta}^* = \{\widehat{\beta}_1^*, \widehat{\beta}_2^*, \ldots, \widehat{\beta}_B^*\}$, and $\widehat{\beta}_{(j)}^*$ represents all values of the j-th component across the elements in $\widehat{\beta}^*$.

doi:10.6342/NTU202500447

Based on this procedure, we successfully corrected the bias in LassoNet. The effectiveness of this correction is demonstrated by our simulation experiments.



doi:10.6342/NTU202500447

Chapter 4 Simulation

In this chapter, we demonstrate the empirical performance of our debiasing procedure for the LassoNet model. For this experiment, we consider a logistic regression model as an example to evaluate the effectiveness of bias correction. The sample size is set to $n = 200$, and the number of covariates $p = 50$. We generate $y_i's$ following the model in 3.12, where each $\boldsymbol{x}_i$ is sampled from $\mathcal{N}(0, 1)$ for $i = 1, \cdots, n$, and $g_{\boldsymbol{w}}$ is a FCNN with 10 layers and 50 nodes per layer. Let $\boldsymbol{\beta}_s$ denote the non-zero coefficients. We set the true coefficient vector as $\boldsymbol{\beta} = (\boldsymbol{B}, \mathbf{0}_{p-6})^T$ where $\boldsymbol{B} = (1.5, 0.5, 1.5, 0.5, 1.5, 0.5)^T$ represents the true signal, and $\mathbf{0}_k$ is a zero vector of length k. This setting demonstrates a weak sparsity level, as well as a smaller contrast between strong signals (1.5) and weak signals (0.5). For this synthetic data, we select the regularization parameter λ by training the Lasso model using 10-fold cross-validation, rather than tuning it in every repetition. This decision is based on findings from Lemhadri et al. (2021), which demonstrated in their experiments that the regularization parameter λ in the LassoNet model exhibits the same property as in Lasso; specifically, as λ increases, the shrinkage effect becomes stronger. However, the effect of M on bias remains unclear. Thus, in this experiment, we focus solely on investigating the effect of different values of M on the debiasing performance. Additionally, to prevent overfitting, we train the model using a single layer with 10 nodes.

To evaluate the validity of the debiasing performance with bootstrapping, we com-

doi:10.6342/NTU202500447

pare the performances of the following estimators: the original Lasso estimator, the LassoNet estimator, the debiased LassoNet estimator (DB-Net), the un-debiased bootstrap LassoNet estimator (BS-Net), where the LassoNet estimator is bootstrapped directly, and the debiased bootstrap LassoNet estimator (BS-DB-Net). The following reports are based on 50 repetitions.

4.1 Evaluation of estimation bias

First, we present the overall loss for all estimators, measured as $\|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2$. As shown in Figure 4.1

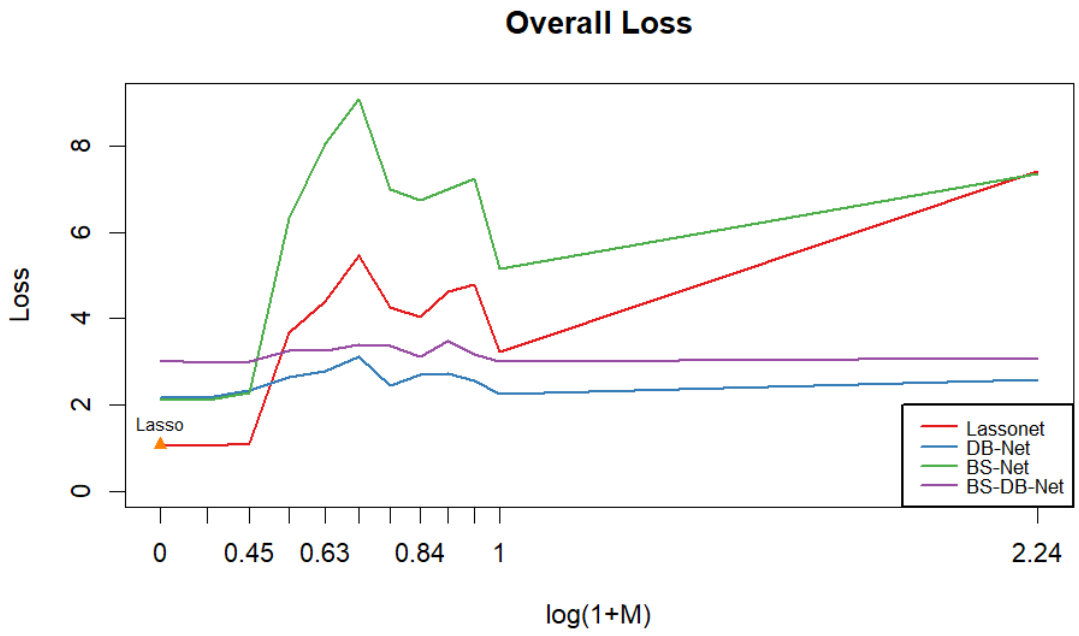


Figure 4.1: Comparison of the overall loss, measured as $\|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2$ across Lasso, the LassoNet, the debiased LassoNet estimator (DB-Net), the un-debiased bootstrap LassoNet (BS-Net), and the debiased bootstrap LassoNet (BS-DB-Net).

it is evident that as M increases, the bias of the LassoNet model becomes larger, especially in the weak non-linearity region ($M < 1$), where the bias increases dramatically.

doi:10.6342/NTU202500447

Meanwhile, the BS-Net also exhibits a large bias, as it is bootstrapped from an estimator with significant bias. This observation highlights that directly bootstrapping the LassoNet estimator cannot achieve good debiasing performance. Instead, our two-step debiasing procedure is necessary to address this issue effectively. Moreover, we observe that BS-Net and BS-DB-Net are competitive in terms of overall loss; however, their bias remains larger than that of the Lasso estimator. Nonetheless, as shown in Figure 4.2

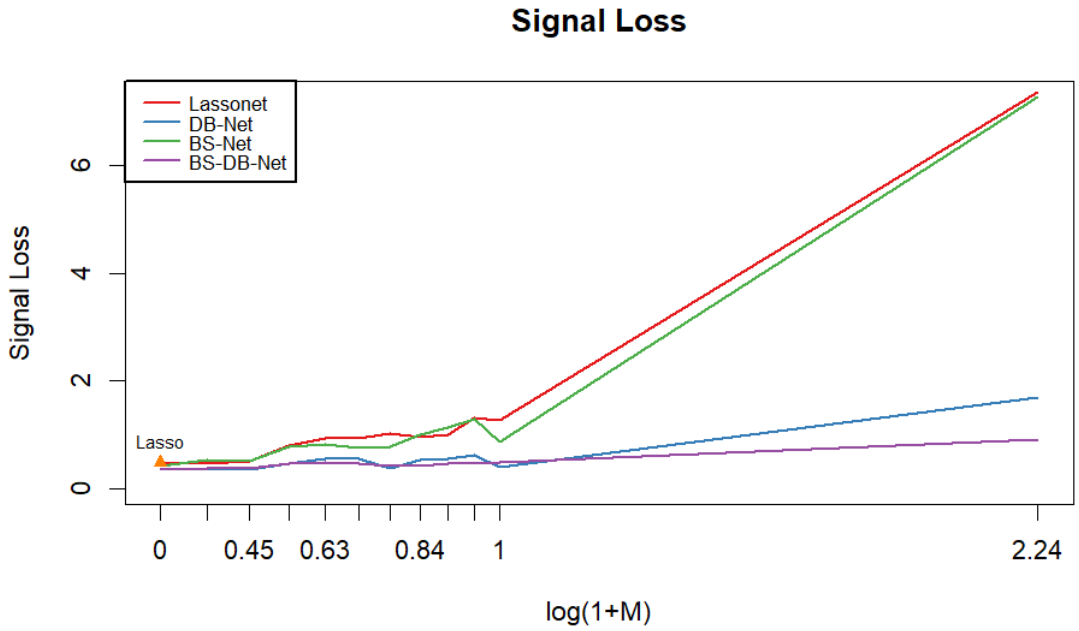


Figure 4.2: Comparison of the signal loss, measured as $\|\widehat{\boldsymbol{B}} - \boldsymbol{B}\|_2$ across Lasso, the LassoNet, the debiased LassoNet estimator (DB-Net), the un-debiased bootstrap LassoNet (BS-Net), and the debiased bootstrap LassoNet (BS-DB-Net).

when focusing solely on the signal, we observe that as the non-linearity becomes stronger, the bias of LassoNet and BS-Net increases at a faster rate. Meanwhile, we also observe that BS-DB-Net outperforms DB-Net in cases of strong non-linearity.

doi:10.6342/NTU202500447

4.2 Evaluating variable selection performance

Next, we evaluate the variable selection performance of all estimators. In Figure 4.3

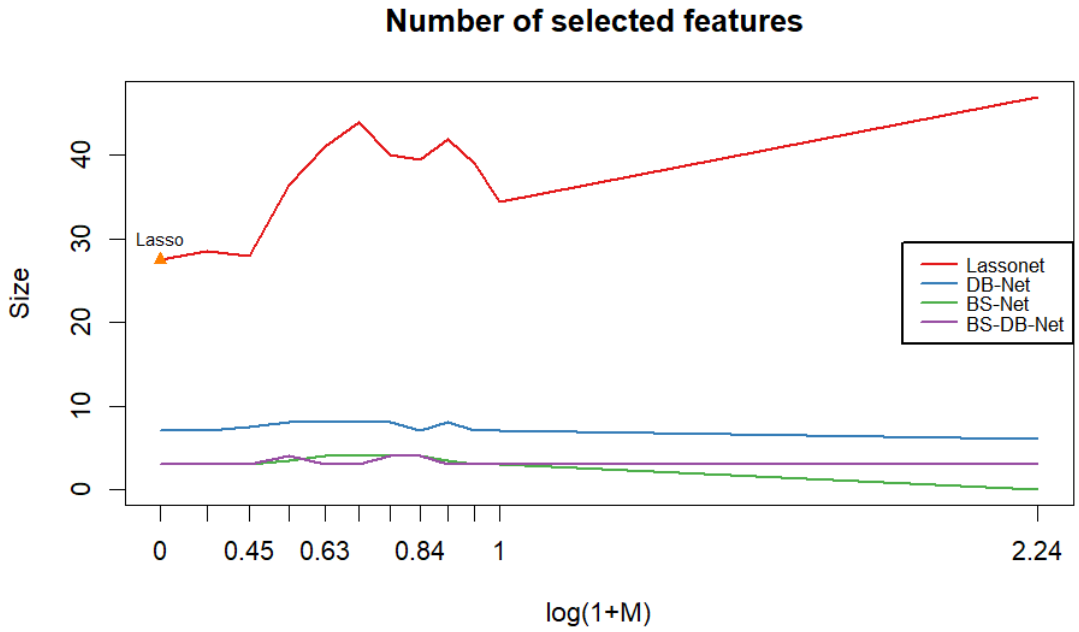


Figure 4.3: Comparison of the number of selections among Lasso, the debiased LassoNet estimator (DB-Net), the un-debiased bootstrap LassoNet estimator (BS-Net), and the debiased bootstrap LassoNet estimator (BS-DB-Net).

we compare the number of selected variables among the four estimators. Note that for DB-Net, BS-Net, and BS-DB-Net, the coefficients are not exactly zero. Variables are selected based on whether 0 falls within the 95% confidence interval defined in step 7 of Algorithm 3. Specifically, if 0 is within the interval, the variable is not selected; otherwise, it is selected. As shown in Figure 4.3, Lasso and LassoNet tend to select too many variables, especially under strong non-linearity, where LassoNet almost selects all variables. In contrast, BS-DB-Net, DB-Net, and BS-Net successfully achieve sparse solutions. However, Figures 4.4 and 4.5

doi:10.6342/NTU202500447

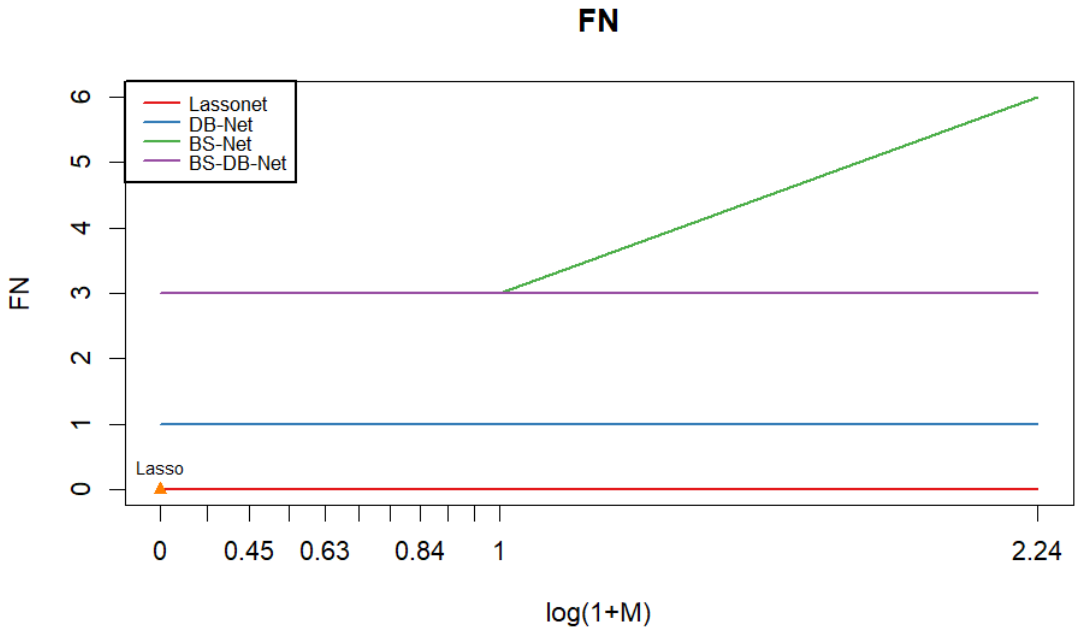


Figure 4.4: False negatives for the debiased LassoNet estimator (DB-Net), the un-debiased bootstrap LassoNet estimator (BS-Net), and the debiased bootstrap LassoNet estimator (BS-DB-Net).

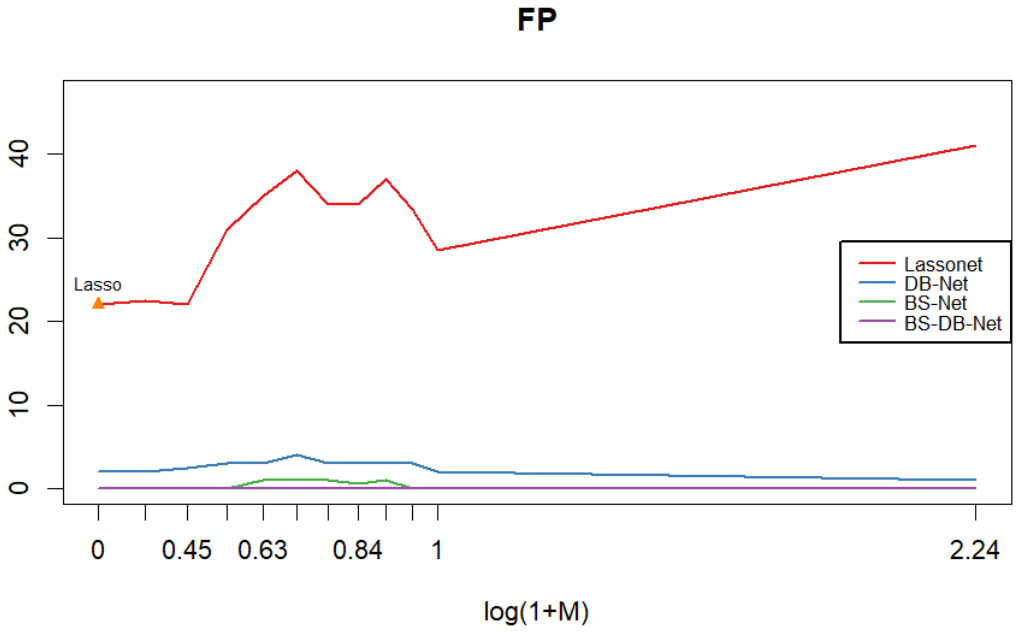


Figure 4.5: False positives for the debiased LassoNet estimator (DB-Net), the un-debiased bootstrap LassoNet estimator (BS-Net), and the debiased bootstrap LassoNet estimator (BS-DB-Net).

reveal that BS-Net exhibits a higher rate of false negatives compared to the other two

doi:10.6342/NTU202500447

estimators, particularly under strong non-linearity. This occurs because BS-Net shrinks all coefficients toward zero, making it unable to detect true signals. Meanwhile, although both BS-DB-Net and DB-Net fail to detect all true signals, BS-DB-Net consistently identifies all strong signals with zero false positives, ensuring that no irrelevant features are selected. In contrast, DB-Net selects both incorrect signals and irrelevant features, compromising the model's interpretability.

doi:10.6342/NTU202500447

Chapter 5 Conclusion

In this thesis, we extend the LassoNet model to GLMs, enabling it to capture both linear and nonlinear relationships between the response and covariates. Additionally, we address the inherent bias in Lasso-type models caused by the ℓ_1-regularization penalty. To mitigate the bias in the extended version, we incorporate node-wise Lasso regression and bootstrap techniques into the debiasing process. This debiasing procedure allows the extended model not only to retain the advantages of Lasso-type models, namely, the ability to select relevant features but also to correct the shrinkage in the estimated coefficients. Our numerical studies demonstrate that the BS-DB-Net procedure achieves less signal loss compared to four alternative estimators when the relationship between the input and output is weakly linear. Furthermore, our procedure exhibits superior performance in variable selection by accurately identifying relatively strong signals and avoiding the selection of irrelevant features. While the current procedure performs effectively on synthetic data, reducing the computational cost of training the neural network and the bootstrap process could open avenues for applying this method to high-dimensional settings, including ultra-high-dimensional scenarios.

doi:10.6342/NTU202500447

doi:10.6342/NTU202500447

References

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. (2021). Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. Journal of big Data, 8:1–74.

Andrews, D. W. and Lu, B. (2001). Consistent model and moment selection procedures for gmm estimation with application to dynamic panel data models. Journal of Econometrics, 101(1):123–164.

Boyd, S. and Vandenberghe, L. (2004). Convex optimization. Cambridge university press.

Candes, E. and Tao, T. (2007). The dantzig selector: Statistical estimation when p is much larger than n. The Annals of Statistics, 35(6):2313–2351.

Chatterjee, A. and Lahiri, S. N. (2011). Bootstrapping lasso estimators. Journal of the American Statistical Association, 106(494):608–625.

Chen, Y. and Li, J. (2021). Recurrent neural networks algorithms and applications. In 2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE), pages 38–43. IEEE.

doi:10.6342/NTU202500447

Efron, B. (1992). Bootstrap methods: another look at the jackknife. In Breakthroughs in statistics: Methodology and distribution, pages 569–593. Springer.

Feng, J. and Simon, N. (2019). Sparse-input neural networks for high-dimensional nonparametric regression and classification.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). The elements of statistical learning: data mining, inference and prediction. Springer, 2 edition.

He, K., Xu, H., and Kang, J. (2019). A selective overview of feature screening methods with applications to neuroimaging data. WIREs Computational Statistics, 11(2):e1454.

Kingma, D. P. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Lahiri, S. (2010). Asymptotic properties of the residual bootstrap for lasso estimators. Proceedings of the American Mathematical Society, 138(12):4497–4509.

Lemhadri, I., Ruan, F., Abraham, L., and Tibshirani, R. (2021). Lassonet: A neural network with feature sparsity. Journal of Machine Learning Research, 22(127):1–29.

Li, S. (2020). Debiasing the debiased lasso with bootstrap. Electronic Journal of Statistics, 14(1):2298–2337.

Liu, B., Zhang, Q., Xue, L., Song, P. X. K., and Kang, J. (2024). Robust high-dimensional regression with coefficient thresholding and its application to imaging data analysis. Journal of the American Statistical Association, 119(545):715–729.

Meinshausen, N. and Bühlmann, P. (2006). High-dimensional graphs and variable selection with the Lasso. The Annals of Statistics, 34(3):1436 – 1462.

doi:10.6342/NTU202500447

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B: Statistical Methodology, 58(1):267–288.

van de Geer, S., Bühlmann, P., Ritov, Y., and Dezeure, R. (2014). On asymptotically optimal confidence regions and tests for high-dimensional models. The Annals of Statistics, 42(3):1166 – 1202.

Xia, L., Nan, B., and Li, Y. (2023). Debiased lasso for generalized linear models with a diverging number of covariates. Biometrics, 79(1):344–357.

Zhang, C.-H. and Zhang, S. S. (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. Journal of the Royal Statistical Society Series B: Statistical Methodology, 76(1):217–242.

Zhao, P. and Yu, B. (2006). On model selection consistency of lasso. The Journal of Machine Learning Research, 7:2541–2563.

Zou, H. (2006). The adaptive lasso and its oracle properties. Journal of the American Statistical Association, 101(476):1418–1429.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society Series B: Statistical Methodology, 67(2):301–320.

doi:10.6342/NTU202500447