STATISTICAL MACHINE LEARNING THEORY AND METHODS FOR HIGH-DIMENSIONAL LOW SAMPLE SIZE (HDLSS) PROBLEMS

By

Kaixu Yang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Statistics—Doctor of Philosophy

2020

ABSTRACT

STATISTICAL MACHINE LEARNING THEORY AND METHODS FOR HIGH-DIMENSIONAL LOW SAMPLE SIZE (HDLSS) PROBLEMS

By

Kaixu Yang

High-dimensional low sample size (HDLSS) data analysis have been popular nowadays in statistical machine learning fields. Such applications involves a huge number of features or variables, but sample size is limited due to reasons such as cost, ethnicity and etc. It is important to find approaches to learn the underlying relationships via a small fraction of data. In this dissertation, we study the statistical properties for some non-parametric machine learning models that deal with these problems and apply these models to various fields for validation.

In Chapter 2, we study the generalized additive model in the high-dimensional set up with a general link function that belong to the exponential family. We apply a two-step approach to do variable selection and estimation: a group lasso step as an initial estimator, then followed by a adaptive group lasso step to obtain final variables and estimations. We show that under certain conditions, the two-step approach consistently selects the truly nonzero variables and derived the estimation rate of convergence. Moreover, we show that the tuning parameter that minimizes the generalized information criterion (GIC) has asymptotically minimum risk. Simulations in variable selection and estimation are given. Real data examples including spam email and prostate cancer genetic data are also used to support the theory. Moreover, we discussed the possibility of using a l_0 norm penalty.

In Chapter 3, we study a shallow neural network model in the high-dimensional classification set up. The sparse group lasso, also known as the $l_{p,1} + l_1$ norm penalty, is applied

to obtain feature sparsity and a sparse network structure. Neural networks can be used to approximate any continuous function with an arbitrary small approximation error given that the number of hidden nodes is large enough, which is known as the universal approximation theorem. Therefore, neural networks are used to model complicated relationships between the response and predictors with huge interactions. We proved that under certain conditions, the sparse group lasso penalized shallow neural network has classification risk tend to the Bayes risk, which is the optimal among all possible models. Real data examples including prostate cancer genetic data, Alzheimer's disease (AD) magnetic resonance imaging (MRI) data and autonomous driving data are used to support the theory. Moreover, we proposed a $l_0 + l_1$ penalty and showed that the solution can be formulated as an mixed integer second order cone optimization (MISOCO) problem.

In Chapter 4, we propose a stage-wise variable selection technique with deep neural networks in the high-dimensional set up, named ensemble neural network selection (ENNS). We apply an ensemble on the stage-wise neural network variable selection method to further the falsely selected variables, which is shown to be able to consistently filter out unwanted variables and selected the truly nonzero variables under certain conditions. Moreover, we proposed a second approach to further simplify the neural network structure by specifying the desired percentage of nonzero parameters in each hidden layer. A type of coordinate descent algorithm is proposed to obtain the solution from the second step. We also show that the two step approach achieves universal consistency for both regression and classification problems. Simulations are studied to support various arguments. Real data examples including the riboflavin production data, the prostate cancer genetic data and a region of interest (ROI) in MRI data are used to validate the method.

I dedicate this dissertation to my parents, Xiangyang Xu and Zhidong Yang, my wife, Jun Liu, and many friends, who all have been supporting me during this hard time.

ACKNOWLEDGMENTS

Here, I would like to express my deepest gratitude to my advisor Dr. Tapabrata Maiti for his guidance towards my studies and researches. Dr. Maiti is extremely kind and knowledgeable. He always provide constructive insights and suggestions that help me make great progress. Without Dr. Maiti's guidance, I wouldn't be able to have such a profound understanding in this gorgeous field.

I would also like to extend my sincere appreciation to my dissertation committee members, Dr. Lyudmila Sakhanenko, Dr. Ping-shou Zhong and Dr. David Zhu. Their comments and suggestions are extremely beneficial for my research.

I am also grateful to the help I obtained from all the professors in the Department of Statistics and Probability, both academically and in other aspects. During my Ph.D. life, I learn a lot from the courses offered by these excellent professors, and I'm able to apply the knowledge I learn to my research. The professors who I have worked with as a teaching assistant also impress me a lot by the way they deal with different kinds of difficulties.

During my seven years at Michigan State University, I made lots of friends and because of them, I never feel lonely in these years. A lot of thanks to my friends at MSU. They either have graduated and become successful faculty members or statisticians in big companies, or are still working hard to pursue their Ph.D. degree. I sincerely wish all of you have a wonderful future.

Last but not least, I would like to express my sincere thanks to my parents for their support and concerns, as well as my wife for her accompany and working hard together towards a glorious future.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

Introduction

In this Chapter, we will state the problems that we study and discuss some existing work that have significant impact to this field.

1.1 Overview

During the past decades, high-dimensional data analysis became increasing more popular. [11] defines that "High-dimensional statistics refers to statistical inference when the number of unknown parameters is of much larger order than sample size." This will be the definition of high-dimensional statistics through the whole thesis. Sometimes, by "high-dimensional", people may refer to a feature space whose dimension is greater than 2 when constructing data visualization, see for example [3], this is not the "high-dimensional" we refer to in this thesis. Let n be the size of sample we have, and let p be the number of unknown parameters involved, thus we have

$$p >> n \tag{1.1}$$

in high-dimensional statistics. Consider a linear regression model

$$y_i = \mu + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i, \ i = 1, ..., n, \tag{1.2}$$

with $p >> n$. It's obvious that the estimation of the unknown parameters $\mu, \beta_1, ..., \beta_p$ can not be estimated without proper assumptions, such as the sparsity assumption, which we will discuss in the next section. For example, the least squared estimator

$$\hat{\mu}, \hat{\beta}_1, ..., \hat{\beta}_p = \underset{\mu, \beta_1, ..., \beta_p}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (y_i - \mu - \beta_j x_{ij})^2$$

is under-determined thus has infinitely many solution.

Various methods have been proposed to solve this issue since [39] brought the problem to people's sight, where the authors considered an orthogonal design in the case $n = p$. In the rest of this Chapter, we will review the work that has been done in high-dimensional statistics field.

On the other hand, non-parametric modeling has been popular for several decades. Non-parametric research happened as early as in 1947, when [133] studied a local averaging approach. Some literature regarding non-parametric research even trace back to the 1930's about partitioning estimation, but this was not fully studied by then. As [61] states, in non-parametric modeling, one does not restrict the class of possible relationship between the response y and the predictors $x_1, ..., x_p$, but assume that y depends on $x_1, ..., x_p$ through a general function $f(x_1, ..., x_p)$. Different approaches have been used to approximate $f(\cdot)$ in a workable way, including averaging, partitioning, basis expansion, neural network approximation, and etc. Further assumptions can also be made on $f(\cdot)$, for example, assuming an additive structure will result in the generalized additive model (GAM), see for example [63], defined as

$$y_i = \mu + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i, \ i = 1, ..., n. \tag{1.3}$$

The relationship is not as general as the $f(x_1, ..., x_p)$ we just defined, but is much more

handy. We will discuss these non-parametric work in the rest of this Chapter.

1.2 Sparsity in High-dimensional Modeling

Assuming sparsity is an effective way of dealing with the high-dimensionality, i.e., we assume that only s of the p variables are involved in predicting the response y. Rich literature have assumed sparsity, for example see [126, 46, 155, 148, 85, 47, 65, 91]. Taking the linear regression model as an example, let $\boldsymbol{\beta} \in \mathbb{R}^p$ be the parameters. A direct method to obtain sparsity is to restrict the number of nonzero parameters, i.e.,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \frac{1}{n} \|\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta}\|_2^2, \quad s.t. \ \|\boldsymbol{\beta}\|_0 \leq K, \tag{1.4}$$

for some positive integer K, where $\boldsymbol{y} \in \mathbb{R}^n$ is the response vector and $\boldsymbol{x} \in \mathbb{R}^{n \times p}$ is the design matrix. The l_0 norm is defined as

$$\|\boldsymbol{\beta}\|_0 = \{\#\ of\ \beta_j : \ \beta_j \neq 0, \ j = 1, ..., p\}. \tag{1.5}$$

This optimization problem can be written into the Lagrangian form as

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \frac{1}{n} \|\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_0 \tag{1.6}$$

for some corresponding λ of K. However, optimizing a loss function with zero norm has been proved to be a non-deterministic polynomial-time hardness (NP-hard) problem, see [99], which requires exponential time to solve.

Fortunately, it has been proved that instead of directly penalizing the number of nonzero

coefficients by l_0 norm, l_1 norm penalty is able to shrink some coefficients to zero, and thus the features corresponding to these coefficients are not included in the model. As a famous piece of work, [126] proposed the method least absolute shrinkage and selection operator (Lasso) in the linear regression set up, which yields sparse estimations for the parameters. The model considers a l_1 norm penalized least square model, i.e.,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{1.7}$$

for some hyper-parameter λ. The Lasso was not intended for high-dimensional modeling, but yields sparsity in the estimators, i.e., with proper choice of λ, the estimated parameters are shrunk towards zero and some of them will be exactly zero. A number of statisticians have studied the properties of the Lasso, for example, [73] derived the asymptotic distributions of the Lasso estimator and showed its estimation consistency. Later people found that this l_1 norm penalty works well in the high-dimensional set up, both practically and theoretically. [60] derived the l_1 norm bound for a persistent sequence of estimator in the Lasso model. [38] considers the high-dimensional case and considered minimizing the l_1 norm of the parameters restricted to zero training error. He shows that this method consistently identify the true subset of nonzero variables given that the true model is sparse. [95] proposed a neighborhood selection scheme that consistently identifies the nonzero entries in the covariance matrix of a multivariate normal distribution. [152] showed the selection consistency of the Lasso under the irrepresentable condition. [149] studied two different sparsity assumptions in the linear model, the strong sparsity

$$\max_{j=s+1,\ldots,p} |\beta_j| = 0 \tag{1.8}$$

and the weak sparsity

$$\sum_{j=s+1}^{p} |\beta_j| \leq \eta_1 \tag{1.9}$$

for some η_1. The authors showed that Lasso consistently selects the true subset of variables with the weak sparsity assumption and the irrepresentable condition.

Later on, Lasso became much more mature and widely applied to obtain a sparse solution in the high-dimensional set up. Variations of the Lasso also emerges rapidly. [46] mentioned that the best tuning parameter λ may not be obtained by minimizing the prediction error, thus Lasso can not reach selection consistency and prediction consistency at the same time. Therefore, Lasso does not have the so-called oracle property. The authors proposed the smoothly clipped absolute deviation penalty (SCAD) and showed its oracle property. [41] argued that Lasso does not work well in correlated predictors, thus [156] proposed the elastic net penalty to overcome this issue. In the next year, [155] proposed a data-driven approach named the adaptive lasso, where a weighted l_1 norm of the parameters is used to replace the l_1 norm, i.e.,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \frac{1}{n} \|\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j|, \tag{1.10}$$

for some $w_j,\ j = 1, ..., p$. A simple convention is to choose $w_j = 1/|\hat{\beta}_j|$ for some initial estimator $\hat{\beta}_j$, and set $w_j = \infty$ if the initial estimator is zero. The author showed that the adaptive Lasso enjoys the oracle property. [20] proposed the Dantzig selector and showed its oracle property.

[148] discussed various grouped variable selection techniques, including the well-known group lasso penalty, see also [135, 65]. Consider that $\boldsymbol{\beta} = (\beta_{1,1}, ..., \beta_{1,g_1}, ..., \beta_{p,1}, ..., \beta_{p,g_p})$, where $\boldsymbol{\beta}$ involves p groups with each group having g_j parameters $j = 1, ..., p$. This case is useful in a lot of real world applications. For example, if a categorical variable include 3

levels, after one-hot encoding, it does not make sense to only include one of the three levels in the model. The group lasso minimizes the loss function plus a grouped penalty

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} \sqrt{g_j}\|\boldsymbol{\beta}_j\|_2, \tag{1.11}$$

where $\boldsymbol{\beta}_j$ denotes parameter vector in the j^{th} group. The sum of l_2 norms encourages group-wise sparsity but in-group non-sparsity. Adaptivity in the group lasso is also discussed by the authors. To be more general, [5] named this type of penalty as $l_{p,1}$ norm penalty, where the l_2 norm in equation 1.11 is replaced by l_p norm with $1 \leq p < \infty$.

A more intuitive penalty emerged later as [114] proposed the sparse group lasso, which is a combination of the group lasso and the lasso

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{j=1}^{p} \sqrt{g_j}\|\boldsymbol{\beta}_j\|_2 + \lambda_2 \sum_{j=1}^{p}\sum_{g=1}^{g_j} |\beta_{j,g}|, \tag{1.12}$$

The penalty encourages both group-wise sparsity and in-group sparsity, and is greatly beneficial in neural network models. A huge number of different types are also widely used in the statistics and machine learning applications, for example, the fused Lasso [127], the graph fused Lasso [122], the tree Lasso [71] and etc.

1.2.1 Choosing the Tuning Parameters

The power of regularization is decided through the hyper parameter, which is also called the tuning parameter. Intuitively, choosing larger tuning parameter λ results in a more sparse model. However, make the model too sparse may lead to losing the true underlying relationship between the response and the predictors. Several criteria are useful in helping

choose a appropriate tuning parameter, for example, BIC [111], EBIC [24], and GIC [151, 52]. Specifically, [52] showed the risk consistency of tuning parameter selection using GIC. Another practical tool is the cross-validations, where the tuning parameter is chosen such that the cross-validated metric is optimized. Conventional theory for cross-validation is given in chapter 8 of [61].

1.2.2 Algorithms for Training Sparse Models

The regularization methods usually involve l_1 norm penalty term, which is not easy to solve using regular gradient descent algorithms, see for example [142]. This issue is general for all models with l_1 penalty. Note that the group lasso penalty does not have explicit l_1 norm penalty but involves sum of l_2 norms, which is equivalent to l_1 norm penalty in terms of optimization theory. Coordinate descent algorithms [143, 54], also known as proximity gradient descent algorithms are useful in dealing with the l_1 norm penalty. For the l_1 norm penalty, not restricted to the linear regression case, a soft-thresholding function $S(\cdot,\cdot) : \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}^p$, where

$$(S(\boldsymbol{z},\lambda))_j = \text{sign}(z_j)(|z_j| - \lambda)_+,\ j = 1, ..., p \tag{1.13}$$

can be applied after the gradient descent for the smooth part to zero some parameters. While for the group lasso penalty, the soft-thresholding function becomes $S(\cdot,\cdot) : \mathbb{R}^{\sum_{j=1}^p g_j} \times \mathbb{R} \to \mathbb{R}^p$, where

$$(S(\boldsymbol{z},\lambda))_j = \left(1 - \frac{\lambda\sqrt{g_j}}{\|\boldsymbol{\theta}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\beta})\|_2}\right)_+ \boldsymbol{\beta}_j,\ j = 1, ..., p, \tag{1.14}$$

where $\boldsymbol{\theta}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\beta})$ is a quantity depending on $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{\beta}$ and the loss function. The coordinate descent algorithm might be slow in updating the groups, thus a block co-ordinate gradient descent method by [132] can be applied, where a second order Taylor approximation is used to simplify the smooth part of the original loss function.

As the development of classical algorithms, the models can also be formulated as classical optimization problems and solved with existing software. The Lasso in the linear regression case can be formulated as a linear programming problem, see for example [30]. The group Lasso in the linear regression set up can be formulated as a second order cone quadratic programming problem, see for example [1]. Mature packages are available for solving these classical optimization problems. As we mentioned the l_0 penalty before, the fast development of optimization software also make this type of penalty easier to apply. The l_0 penalty in the linear regression set up can be formulated as a mixed integer second order cone optimization (MISOCO) problem, see for example [92].

1.2.3 Stage-wise Selection

Various stage-wise algorithms are used to obtain a path selection. The least angle regression [41] provides a forward algorithm to add new features by looking at the correlation. The LARS algorithm with simple modification can be used to obtain the lasso solution path. [128] provides a stage-wise algorithm, which provides very close solution path to the lasso solution path. [102] studied a stage-wise algorithm to incorporate the l_2, l_1 and l_0 norm penalty with the gradients with respect to the input weights. The gradient has implicit connections with the correlation studied in [41].

It worth noting that [125] has shown that there is an equivalence between using the stage-wise algorithm and the group lasso penalty. This built a connection between the

regularization methods and the stage-wise variable selection algorithms. [82] has applied the result on deep artificial neural networks to do feature selection. Compared with optimizing penalized loss functions, stage-wise algorithms starts from the null model and adds variables gradually. This is beneficial to complicated models such as neural networks, since learning a complicated model on all variables involves unpredictable uncertainty, and thus might be sensitive to initialization.

1.3 The Projection Approach

Another popular approach to deal with the high-dimensionality, though not studied too much in this thesis, is worth mentioning, the projection-based methods, also known as dimensionality reduction. Projection methods find a lower dimensional representation for the original feature space. Classical work include [141, 129, 31, 16, 68, 157, 21]. Projection methods have more flavor of unsupervised learning, since the lower dimensional representation should not depend on the response but is purely a property of the design matrix. A main drawback of projection based approaches is that one loses interpretability, because the projected features are no longer the original features, and thus not interpretable.

General dimensional reduction methods do not work in the high-dimensional set up. For example, the principal component analysis (PCA) uses a linear projection on $\boldsymbol{x}$. It uses a matrix $\boldsymbol{A} \in \mathbb{R}^{d\times p}$ to project $\boldsymbol{x} \in \mathbb{R}^p$ to $\boldsymbol{A}\boldsymbol{x} \in \mathbb{R}^d$, where the matrix $\boldsymbol{A}$ is chosen to maintain the greatest variance based on the training data. During the computation, the sample covariance matrix $\hat{\Sigma}_n = \boldsymbol{X}^T\boldsymbol{X}/n$ is used to estimate the population covariance matrix $\Sigma = Cov(\boldsymbol{x})$, and the principal components can be obtained from the eigenvectors of the sample covariance matrix. However, in the high-dimensional set up, the sample covariance

matrix is no longer a consistent estimator of the population covariance matrix, i.e.

$$\hat{\Sigma}_n \not\to \Sigma \quad as\ n \to \infty,$$

if we have $p > n$. [69] gave the conditions when PCA works in the high dimensional case: the largest eigenvalues have to be large enough.

Variations of the PCA include the Simplified Component Technique LASSO (SCoTLASS) by [68] and the sparse PCA by [157]. The former considers a L_1 penalty on the projection matrix $\boldsymbol{A}$ that yields some sparse solution in $\boldsymbol{A}$. However, this is not feasible in practice, since there is no guideline for choosing the tuning parameter and a high computational cost due to the non-convexity. The latter obtained the connection between PCA and ridge regression, and use the ridge or elastic net to obtain a sparse approximation to the original PCA. In both methods, their projection matrix $\boldsymbol{A}$ is sparse in the original features, i.e., some columns of $\boldsymbol{A}$ are exactly zeros. These build a bridge connecting with the sparse assumption to some extent.

After the classic work of manifold learning (non-linear dimensionality reduction) such as [130, 107, 124], current manifold learning focuses on two aspects: image processing and data visualization. In image processing, manifold learning can be used to reduce the dimensionality to its intrinsic dimension, see for example [89, 154, 84]. These applications are not generalized to a broader set up. Manifold learning applied to data visualization usually reduces the dimensionality to two or three, see for example [139]. These applications are not useful in building models. Another concern is the out-of-sample performance of manifold learning, which is recently studied by [123]. According to the paper, most current manifold learning algorithms fail to map new data to the previously learnt embedding.

Autoencoder [75] uses neural network as a non-linear projection to encode the original feature space to a lower dimensional space, which is chosen to be such that the original feature space can be recovered with another neural network. Training a high-dimension neural network is still challenging, sparse autoencoder [100, 90] eliminates the insignificant connections and thus kills those parameters.

Another useful tool is the random projection, which relies on the Johnson-Lindenstrauss lemma

Lemma 1.1 ([67])**.** Given $0 < \epsilon < 1$, a set $\boldsymbol{X}$ of m points in $\mathbb{R}^N$, and a number $n > 8\log(m)/\epsilon^2$, there is a linear map $f : \mathbb{R}^N \to \mathbb{R}^n$ such that

$$(1-\epsilon)\|\boldsymbol{u}-\boldsymbol{v}\|^2 \leq \|f(\boldsymbol{u})-f(\boldsymbol{v})\|^2 \leq (1+\epsilon)\|\boldsymbol{u}-\boldsymbol{v}\|^2$$

for all $u, v \in \boldsymbol{X}$.

The lemma states that there exist a lower dimensional projection that well maintains the distance in the original feature space, however, there isn't a guideline of how to find such a lower dimensional space. [21] used random projection over classical classifiers as an ensemble classifier, and provided an efficient algorithm due to the fact that the random projection matrices are drawn from distributions rather than computed from data. They also showed that the best random projection can be found in linear time when the random projection matrices are sampled according to Haar measure. One possible drawback of random projection is that it does not provide us with any lower-dimensional representation information.

1.4 Non-parametric Modeling

Non-parametric modeling assumes an arbitrary relationship between the response and the variables. We would like to find a function $f(\cdot) : \mathbb{R}^p \to \mathbb{R}$ such that $f(\boldsymbol{x})$ is a good approximation of y or a representation of y, see [61]. It's common to assume that $f(\cdot)$ is continuous, but this is not necessary. Complicated machine learning models such as tree models, see [108, 81, 56], neural networks, see [116] and etc. can be considered as non-parametric models that approximate a complicated relationship. An arbitrary function with great flexibility can be wiggly, i.e., it may perfectly fit the training data. Therefore, people usually need to restrict the smoothness of a function by restricting

$$\int \left(f''(x)\right)^2 dx, \tag{1.15}$$

see for example [11]. Obviously, a linear function has the quantity equal to zero and thus is the smoothest, but the approximation power of a linear function is sub-optimal. Therefore, a balance between the approximation power and the smoothness needs to be considered.

1.4.1 Basis Expansion

Directly working with the arbitrary function f is not realistic, since the candidate space is infinitely dimensional. Therefore, parametric approximations are need. Consider the univariate case, a great number of approaches can be used to approximate a continuous function $f(\cdot) : \mathbb{R} \to \mathbb{R}$. The most popular approach is basis expansion

$$f(x) = \sum_{k=1}^{\infty} \beta_k \phi_k(x), \tag{1.16}$$

where $\{\phi_k(\cdot),\ k \in \mathbb{N}^+\}$ is a set of basis functions. A finite number of bases is usually used to approximate the infinite sum. As mentioned above, a set of basis with maximal differentiability is a good property, therefore B-spline is among the most popular basis functions, see [110]. According to [120], a spline with degree l consists of piece-wise polynomials up to degree l on K pre-specified partitions, with connection knots l' times continuously differentiable, where $0 \leq l' \leq l-2$. According to [65], there exists a normalized B-spline basis $\{\phi_k\}$ such that

$$f_n(x) = \sum_{k=1}^{K+l} \beta_k \phi_k(x). \tag{1.17}$$

[65] showed that using the above B-spline basis, we have the following approximation error

$$\|f - f_n\|_2^2 = \int_a^b (f(x) - f_n(x))^2 dx = O((K+l)^{-2d}), \tag{1.18}$$

where the parameter $d = k + \alpha$ is such that the k^{th} derivative of function $f(\cdot)$ satisfies the Lipschitz condition of order α

$$|f^{(k)}(s) - f^{(k)}(t)| \leq C|s-t|^{\alpha},$$

for some constant C. This result shows that as we increase the number of partitions in a basis expansion, the approximation error can be arbitrarily small under certain conditions.

1.4.2 Neural Networks

To approximate an arbitrary multi-variate function $f(\cdot) : \mathbb{R}^p \to \mathbb{R}$, neural network is a powerful tool. Neural network is used to mimic human's brain: the input does not lead to the output directly, but a few intermediate nodes are needed. Originated from a multi-layer

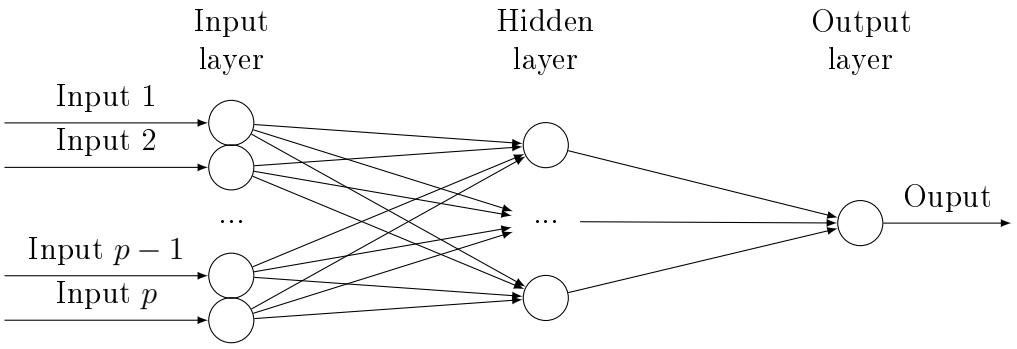

Figure 1.1: A diagram for the single-layer neural network model

perceptron in [106], which is a multi-layer version of the perceptron in [105], neural network has been getting deeper and more powerful in approximating a continuous function. figure 1.1 shows a diagram of a single hidden layer neural network, which is also known as a shallow neural network. The hidden layer consists of linear combinations of the first layer's inputs plus a non-polynomial activation function.

Mathematically, let $\boldsymbol{x} \in \mathbb{R}^p$ be the input vector, the output of a shallow neural network is

$$\eta(\boldsymbol{x}) = \sum_{k=1}^{K} \beta_k \cdot \sigma\left(\boldsymbol{\theta}\boldsymbol{x} + \boldsymbol{t}\right) + b \in \mathbb{R}, \tag{1.19}$$

where $\beta_k \in \mathbb{R},\ k = 1, ..., K$ are the output layer coefficients, $\boldsymbol{\theta} \in \mathbb{R}^{K \times p}$ is the hidden layer coefficient matrix, $\boldsymbol{t}$ is the hidden layer intercept vector, b is the output layer intercept, and $\sigma(\cdot)$ is an activation function. A notable theorem on the approximation power of a shallow neural network is given by [33] as follow

Theorem 1.1 (Universal approximation theorem, Cybenko 1989)**.** Let $\sigma(\cdot)$ be such that $\sigma(t) \to 0$ as $t \to -\infty$ and $\sigma(t) \to 1$ as $t \to \infty$. For a continuous function f on $[0, 1]^n$ and an

arbitrary $\epsilon > 0$, there exist a K and parameters $\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{t}, b$ such that

$$|f(\boldsymbol{x}) - \eta(\boldsymbol{x})| < \epsilon, \quad \forall x \in [0,1]^n$$

This theorem guarantees the universal approximation ability of shallow neural networks and is the reason that neural networks are useful in many cases. Typical activation functions includes

- Hyperbolic Tangent: $\sigma(x) = tanh(x)$.
- Sigmoid: $\sigma(x) = \frac{e^x}{1+e^x}$.
- Rectified Linear Unit (ReLU): $\sigma(x) = x_+$.
- Leaky ReLU: $\sigma(x) = \max \epsilon x, x$ for some small $\epsilon > 0$.

Actually, as long as that the activation function is not polynomial, a shallow neural network has the universal approximation ability, see [113]. Rectified Linear Unit (ReLU) is one of the most popular activation functions nowadays, though it may suffer from the dead ReLU problems, see [87]. It's computationally efficient and converges fast.

1.4.3 Deep Neural Networks

Deep neural networks refer to neural networks with more than one hidden layers. As the development of deep neural networks, people found the limitations of shallow neural network such that the number of neurons needed to achieve a desire error increases as fast as exponentially, see [29, 28]. After that people found that "the two hidden layer model may be significantly more promising than the single hidden layer model", see [103]. Sum neural networks, or equivalently, polynomial neural networks have been studied [36, 86], and universal

approximation property has been established recently by [43] that a continuous function in $\mathcal{F}_d^n$ can be approximated with error ϵ by a quadratic network who have depth

$$O\left(\log(\log(\frac{1}{\epsilon})) + \log(\frac{1}{\epsilon})\right)$$

and number of weights

$$O\left(\log(\log(\frac{1}{\epsilon}))(\frac{1}{\epsilon})^{d/n} + \log(\frac{1}{\epsilon})(\frac{1}{\epsilon})^{d/n}\right)$$

where d is the dimension of domain. The approximation theory for regular deep neural networks have also been established recently. [104] showed that a deep network need

$$O\left((n-1)\left(\frac{\epsilon}{L}\right)^{-2}\right)$$

model complexity to approximate a L-Lipshitz continuous function of n variables instead of

$$O\left(\left(\frac{\epsilon}{L}\right)^{-n}\right)$$

in a shallow neural network. [112, 113] provides more detailed results for the deep neural network approximation power.

Therefore, it's promising that a deeper neural network works better in learning complicated relationships. Through this thesis, we will start from the generalized additive model (GAM), study its properties and tuning parameter selection in Chapter 2, then move to a more complicated shallow neural network model and talk about its asymptotic properties in chapter 3, and finally dive into deep neural networks in chapter 4, where we proposed an en-

semble variable selection algorithm and estimation method. In chapter 5, we will summarize the work we have done and the future work to be done.

Chapter 2

High-dimensional Generalized Additive Modeling

In this chapter, we will study the generalized additive model (GAM), see [63]. We consider a non-linear link function that belong to the exponential family. A two-step approach is used for variable selection and estimation, with a group lasso followed by an adaptive group lasso applied to the basis expansion of the non-parametric functions. We will show that the variable selection result in the second step is consistent, and we also derive the convergence rate of the second step estimator. Moreover, we discuss the tuning parameter selection and showed that we reach risk consistency by using the generalized information criterion (GIC). Simulations and read data examples on various aspects are given to support the arguments.

2.1 Introduction

The main objective of this work is to establish theory driven high dimensional generalized additive modeling method with nonlinear links. The methodology includes consistently selecting the tuning parameter. Additive models play important roles in non-parametric statistical modeling and in machine learning. Although this important statistical learning tool has been used in many important applications and there are free software available for implementing these models along with their variations, to our surprise, there is no procedure with nonlinear

link available which has been studied systematically with theoretical foundation. generalized additive modeling allows the nonlinear relationship between a response variable and a set of covariates. This includes the special case, namely, the generalized linear models, by letting each additive component be a linear function. In general, let $(y_i, \boldsymbol{X}_i), i = 1, ..., n$ be independent observations, where y_i's are response variables whose corresponding p-dimensional covariate vectors are $\boldsymbol{X}_i$'s. A generalized additive model [62] is defined as

$$\mu_i = E(y_i|\boldsymbol{X}_i) = g^{-1}\left(\sum_{j=1}^{p_n} f_j(X_{ij})\right), \tag{2.1}$$

where $g(\cdot)$ is a link function, f_j's are unspecified smooth functions and X_{ij} is the jth component of vector $\boldsymbol{X}_i$. One of the functions could be a constant, which is the intercept term, but this is not necessary. The number of additive components is written as p_n, since it sometimes (usually in high dimensional set up) increases as n increases. A simple case that many people studied is $p_n = p$, where the number of additive components is fixed and less than the sample size n. The choice of link functions is as simple as in generalized linear models, where people prefer to choose link functions that make the distribution of the response variables belong to popular exponential family. A widely used generalized additive model has the identity link function $g(\mu) = \mu$, which gives the classical additive model

$$y_i = \sum_{j=1}^{p_n} f_j(X_{ij}) + \epsilon_i, \tag{2.2}$$

where ϵ_i's are i.i.d random variables with mean 0 and finite variance σ^2.

On the other hand, high dimensional data has increasingly become a part of many modern days scientific applications. Often the number of covariates p_n is much larger than the

number of observations n, which is usually written as $p_n \gg n$. A most interesting scale is p_n increases exponentially as n increases, i.e. $\log p_n = O(n^\rho)$ for some constant $\rho > 0$. [48] called this as non-polynomial dimensionality or ultra high-dimensionality.

In this chapter, we consider the generalized additive model in a high-dimensional set up. To avoid identification problems, the functions are assumed to be sparse, i.e. only a small proportion of the functions are non-zero and all others are exactly zero. A more generalized set up is that the number of nonzero functions, denoted s_n, also diverges as n increases. This case is also considered in this paper.

Many others have worked on generalized additive models. Common approaches use basis expansion to deal with the nonparametric functions, and perform variable selection and estimation methods on the bases. [94] considered a simpler case, as in 2.2, with a new sparsity-smoothness penalty and proved it's oracle property. They also performed a simulation study under logit link with their new penalty, however, no theoretical support was provided. [45] proposed the nonparametric independence screening (NIS) method in screening the model as in 2.2. However, the selection consistency and the generalized link functions were not discussed. [91] discussed the practical variable selection in additive models, but no theory was given. [83] considered a two-step oracally efficient approach in generalized additive models in the low dimensional set up, but no variable selection in the high dimensional set up was done. [65] focused on the variable selection of 2.2 with fixed number of nonzero functions using a two step approach: First group lasso [7, 148] on the bases to select the nonzero covariates and then use adaptive group lasso to estimate the bases coefficients. They then established the selection consistency and provided the rate of convergence of the estimation. [2] reviewed several existing algorithms highlighting the connections between them, including the non-negative garrote, COSSO and adaptive shrinkage, and presented

some computationally efficient algorithms for fitting the additive models. [98] extended the consistency and rate of convergence of [65] to spatial additive models. [51] studied the GAM with identity link under the endogeneity setting. It worth mentioning that alternative methods to penalization have also been studied, for example, [134] studied fitting GAM and perform variable seleciton implicitly through likelihood based boosting.

However, though widely used, no systematic theory about selection and estimation consistency and rate of convergence has been established for generalized additive models with non-identity link functions.

In this chapter, we establish the theory part for generalized additive models with non-identity link functions in high dimensional set up. We develop a two-step selection approach, where in the first step we use group lasso to perform a screening, which, under mild assumptions, is able to select all nonzero functions and not over-select too much. In the second step, the adaptive group lasso procedure is used and is proved to select the true predictors consistently.

Another important practical issue in variable selection and penalized optimization problems is tuning parameter selection. Various cross validation (CV) techniques have been used in practice for a long time. Information criteria such as Akaike information criterion (AIC), AICc, Bayesian information criterion (BIC), Mallow's C_p and etc. have been used to select 'the best' model as well. Many equivalences among the tuning parameter selection methods have been shown in the Gaussian linear regression case. However, the consistency of these selection methods were not established. Later some variations of the information criteria such as modified BIC [150, 137] extended BIC [24] and generalized information criterion (GIC) [52] were proposed and shown to have good asymptotic properties in penalized linear models and penalized likelihoods. However, the results are not useful for grouped variables in

additive models, for which basis expansion technique is usually used and thus brings grouped selection.

In this chapter, we generalize the result of generalized information criterion (GIC) by [52] to group-penalized likelihood problems and show that under some common conditions and with a good choice of the parameter in GIC, we are able to select the tuning parameter that corresponds to the true model.

In section 2.2, the model is specified and basic approach is discussed. Notations and basic assumptions are also introduced in this section. Section 2.3 gives the main results of the two steps selection and estimation procedure. Section 2.4 develops the tuning parameter selection. Variation of penalty functions is discussed in section 2.5. Extensive simulation study and real data example are presented in section 2.6 followed by a short discussion in section 2.7. The proofs of all theorems are deferred to Appendix A.

2.2 Model

We consider the generalized additive model (2.1) where the link function corresponds to an exponential family distribution of the response. For each of the n independent observations, the density function is given as

$$f_{y_i}(y) = c(y) \exp\left[\frac{y\theta_i - b(\theta_i)}{\phi}\right], \ 1 \leq i \leq n, \ \theta_i \in \mathbb{R}. \tag{2.3}$$

Without loss of generality, we assume that the dispersion parameter $0 < \phi < \infty$ is assumed to be a known constant. Specifically we assume $\phi = 1$. We consider a fixed-design through this paper, i.e., the design matrix $\boldsymbol{X}$ is assumed to be fixed. However, we have shown in

appendix A that the same theory works for a random design under simple assumptions on the distribution of $\boldsymbol{X}$. The additive relationship assumes that the densities of y_i's depend on $\boldsymbol{X}_i$'s through the additive structure $\theta_i = \sum_{j=1}^{p_n} f_j(X_{ij})$. This is the canonical link. If we use other link functions, for example, $A(\cdot)$, the theory also works as long as the functions $A(\cdot)$ satisfies the Lipschitz conditions for some order. Let $b^{(k)}(\cdot)$ be the k-th derivative of $b(\cdot)$, then by property of the exponential family, the expectation and variance matrix of $\boldsymbol{y} = (y_1, ..., y_n)^T$, under mild assumptions of $b(\cdot)$, is given by $\boldsymbol{\mu}(\boldsymbol{\theta})$ and $\phi\Sigma(\boldsymbol{\theta})$, where

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = (b^{(1)}(\theta_1), ..., b^{(1)}(\theta_n))^T \quad \text{and} \quad \Sigma(\boldsymbol{\theta}) = \text{diag}\{b^{(2)}(\theta_1), ..., b^{(2)}(\theta_n)\}. \tag{2.4}$$

The log-likelihood (ignoring the term $c(y)$ which is not interesting to us in parameter estimation) can be written as

$$l = \sum_{i=1}^{n} \left[y_i \left(\sum_{j=1}^{p_n} f_j(X_{ij}) \right) - b \left(\sum_{j=1}^{p_n} f_j(X_{ij}) \right) \right]. \tag{2.5}$$

Assume that the additive components belong to the Sobolev space $W_2^d([a, b])$. According to [110], see pages 268-270, there exists B-spline approximation

$$f_{nj}(x) = \sum_{k=1}^{m_n} \beta_{jk}\phi_k(x), \qquad 1 \leq j \leq p. \tag{2.6}$$

with $m_n = K_n + l$, where K_n is the number of internal knots and $l \geq d$ is the degree of the splines. Generally, it is chosed that $d = 2$ and $l = 4$, i.e., cubic splines.

Using the approximation above, [65] proved that f_{nj} well approximates f_j in the sense

of rate of convergence that

$$\|f_j - f_{nj}\|_2^2 = \int_a^b (f_j(x) - f_{nj}(x))^2 dx = O(m_n^{-2d}). \tag{2.7}$$

Therefore, using the basis approximation, the log-likelihood (ignoring the term $c(y)$ which is not related to the parameters) can be written as

$$\begin{aligned} l &= \sum_{i=1}^{n} \left[y_i \left(\sum_{j=1}^{p_n} \sum_{k=1}^{m_n} \beta_{jk}^0 \Phi_k(x_{ij}) \right) - b \left(\sum_{j=1}^{p_n} \sum_{k=1}^{m_n} \beta_{jk}^0 \Phi_k(x_{ij}) \right) \right] \\ &= \sum_{i=1}^{n} \left[y_i \left(\boldsymbol{\beta}^{0T} \Phi_i \right) - b \left(\boldsymbol{\beta}^{0T} \Phi_i \right) \right], \end{aligned} \tag{2.8}$$

where $\boldsymbol{\beta}^0$ and Φ_i are the vector basis coefficients and bases defined below.

It's also worth noting that the number of bases m_n increases as n increases. This is necessary since [110] mentioned that one need to have sufficient partitions to well approximate f_j by f_{nj}. If we fix m_n, i.e. let $m_n = m_0$, though in the later part we will show the approach to estimate the basis coefficients can have better rate of convergence, the approximation error between the additive components and the spline functions $\|f_j(x) - f_{nj}(x)\|_2 = [\int_a^b (f_j(x) - f_{nj}(x))^2 dx]^{1/2} = O(1)$ will increase and lead to inconsistent estimations. Therefore, m_n, or more precisely, K_n, need to increase with n.

Our selection and estimation approach will be based on the bases approximated log likelihood 2.8. Before starting the methodology, we list the notations and state the assumptions we need in this paper.

Notations

The design matrix is $\boldsymbol{X}_{(n \times p_n)} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)^T$.

The basis matrix is $\Phi_{(n\times m_n p_n)} = (\Phi_1, ..., \Phi_n)^T$, where

$$\Phi_i = (\phi_1(x_{i1}), ..., \phi_{m_n}(x_{i1}), ..., \phi_1(x_{ip_n}), ..., \phi_{m_n}(x_{ip_n}))^T.$$

The true basis parameters

$$\boldsymbol{\beta}^0 = (\beta^0_{11}, ..., \beta^0_{1m_n}, ..., \beta^0_{p_n 1}, ..., \beta^0_{p_n m_n})^T \in \mathbb{R}^{m_n p_n}$$

We assume the functions $f_1, ..., f_{p_n}$ are sparse, then $\boldsymbol{\beta}^0$ is block-wise sparse, i.e. the blocks ${\boldsymbol{\beta}^0}_1 = (\beta^0_{11}, ..., \beta^0_{1m_n})^T, ..., {\boldsymbol{\beta}^0}_{p_n} = (\beta^0_{p_n 1}, ..., \beta^0_{p_n m_n})^T$ are sparse.

Let $\boldsymbol{\mu}_y$ be the expectation of $\boldsymbol{y}$ based on the true basis parameters and $\boldsymbol{\varepsilon} = \boldsymbol{y} - \boldsymbol{\mu}_y$.

Define the relationship $a_n \preceq b_n$ as there exists a finite constant c such that $a_n \leq c b_n$.

For any function f define $\|f\|_2 = [\int_a^b f^2(x)dx]^{1/2}$, whenever the integral exists.

For any two collections of indices $S, \tilde{S} \subseteq \{1, ..., p_n\}$, the difference set is denoted $S - \tilde{S}$. The cardinality of S is denoted $\text{card}(S)$. For any $\boldsymbol{\delta} \in \mathbb{R}^{m_n p_n}$, define $\boldsymbol{\delta}_1, ..., \boldsymbol{\delta}_{p_n}$ as its sub-blocks, where $\boldsymbol{\delta}_i \in \mathbb{R}^{m_n}$, and define the block-wise support

$$\text{supp}_B(\boldsymbol{\delta}) = \{j \in \{1, ..., p_n\}; \boldsymbol{\delta}_j \neq \mathbf{0}\}.$$

Define the block-wise cardinality

$$\text{card}_B(\boldsymbol{\delta}) = \text{card}(\text{supp}_B(\boldsymbol{\delta})).$$

For $S = \{s_1, ..., s_q\} \subseteq \{1, ..., p_n\}$, define sub-block vector

$$\boldsymbol{\delta}_S = (\boldsymbol{\delta}_{s_1}^T, ..., \boldsymbol{\delta}_{s_q}^T)^T$$

The number of additive components is denoted p_n, which is possible to grow faster than the sample size n. Let $T = \text{supp}_B(\boldsymbol{\beta}^0)$ and T^c be the compliment set. Let $\text{card}(T) = s_n$, where s_n is allowed to diverge slower than n.

For each $U \subseteq \{1, ..., p_n\}$ with $\text{card}(U - T) \leq m$ for some m, define

$$\mathcal{B}(U) = \{\boldsymbol{\delta} \in \mathbb{R}^{m_n p_n}; \text{supp}_B(\boldsymbol{\delta}) \subseteq U\},$$

$$\mathcal{B}(m) = \{\mathcal{B}(U); \text{for any } U \subseteq \{1, ..., p_n\}; \text{Card}(U - T) \leq m\}.$$

Let q be an integer such that $q > s_n$ and $q = o(n)$. Define

$$\mathcal{B}_1 = \{\boldsymbol{\beta} \in \mathcal{B} : \text{card}_B(\boldsymbol{\beta}) \leq q\},$$

where $\mathcal{B}$ is a sufficiently large, convex and compact set in $\mathbb{R}^d$.

Assumptions

Assumption 2.1 (On design matrix)**.** Using the normalized B-spline bases, the basis matrix Φ has each covariate vector $\Phi_j, j = 1, ..., p_n$ bounded, i.e., $\exists\ c_\Phi$ such that $\|\Phi_j\|_2 \leq \sqrt{n} c_\Phi, \forall j = 1, ..., m_n \times p_n$.

Assumption 2.2 (Restricted Eigenvalues **RE**)**.** For a given sequence N_n, there exist γ_0 and

γ_1 such that

$$\gamma_0 \gamma_2^{2s_n} m_n^{-1} \leq \frac{\boldsymbol{\delta}^T \Phi^T \Phi \boldsymbol{\delta}}{n \|\boldsymbol{\delta}\|_2^2} \leq \gamma_1 m_n^{-1}, \tag{2.9}$$

where γ_2 is a positive constant such that $0 < \gamma_2 < 0.5$, for all $\boldsymbol{\delta} \in \mathcal{C}$, where $\boldsymbol{\delta}^T = (\boldsymbol{\delta}_1^T, ..., \boldsymbol{\delta}_{p_n}^T)$ and

$$\mathcal{C} = \{\boldsymbol{\delta} \in \mathbb{R}^{p_n m_n} : \|\boldsymbol{\delta}\|_2 \neq 0,\ \|\boldsymbol{\delta}\|_2 \leq N_n \text{ and } \text{card}_B(\boldsymbol{\delta}) = o(s_n)\}. \tag{2.10}$$

Assumption 2.3 (On the exponential family distribution)**.** The function $b(\theta)$ is three times differentiable with $c_1 \leq b''(\theta) \leq c_1^{-1}$ and $|b'''(\theta)| \leq c_1^{-1}$ in its domain for some constant $c_1 > 0$. For unbounded and non-Gaussian distributed Y_i, there exists a diverging sequence $M_n = o(\sqrt{n})$ such that

$$\sup_{\boldsymbol{\beta} \in \mathcal{B}_1} \max_{1 \leq i \leq n} \left| b' \left(\left| \Phi_i^T \boldsymbol{\beta} \right| \right) \right| \leq M_n. \tag{2.11}$$

Additionally the error term $\epsilon_i = y_i - \mu_{y_i}$'s follow the uniform sub-Gaussian distribution, i.e., there exist constants $c_2 > 0$ such that uniformly for all $i = 1, .., n$, we have

$$P(|\epsilon_i| \geq t) \leq 2 \exp(-c_2 t^2) \text{ for any } t > 0. \tag{2.12}$$

Assumption 2.4 (On nonzero function coefficients)**.** There exist a sequence $c_{f,n}$ that may tend to zero as $n \to \infty$ such that for all $j \in T$, the true nonzero functions f_j satisfy

$$\min_{j \in T} \|f_j\|_2 \geq c_{f,n}.$$

We note that assumption 2.1 is a standard assumption in high dimensional models, where the design matrix needs to be bounded from above. assumption 2.2 is a well-known condition in high-dimension set up on the empirical Gram matrix [15]. It is different than the regular

eigenvalue condition, since when $n < p$, the $p \times p$ Gram matrix has rank less than p, thus it must have zero eigenvalues. Therefore, it is not realistic to bound the eigenvalues away from zero for all $\boldsymbol{\nu} \in \mathbb{R}^{p_n m_n}$, but we need to restrict to some space $\mathcal{C}$. In our set up, $\mathcal{C}$ is the restricted sub-block eigenvalue condition on sub-blocks of the Gram matrix studied by [9]. Though the lower bound and upper bound are imposed on the fixed design matrix, we gave a derivation in Appendix A that this condition holds when $\boldsymbol{X}$ is drawn from a continuously differentiable density function which is bounded away from 0 and infinity on the domain of $\boldsymbol{X}$. This result is similar to the results in [65].

Assumption 2.3 is a standard assumption to generalized models. 2.11 and 2.12 together controls the tail behavior of the responses, and as mentioned by [52], ensures a general and broad applicability of the method. Analogous assumptions to 2.11 can also be seen in [50] and [11]. Specifically, for example, we have $b(\boldsymbol{\theta}) = \log(1 + \exp(\boldsymbol{\theta}))$. It's easy to verify that both its second and third derivatives have their absolute values all bounded from above by 1. For equation (2.11), observe that the first derivative is the mean of Bernoulli distribution, and thus it is also bounded. The error term is also bounded by 1, therefore, taking $c_2 = \log(2)$ will make equation (2.12) satisfy all logistic regression cases. Moreover, bounded second moment in logistic regression ensure that there exist ϵ such that the probability p_i of each observation satisfies $\epsilon < p < 1 - \epsilon$.

Assumption 2.4 appears often in variable selection methodologies, because intuitively a nonzero function or covariate has to contribute enough to the response in order to be considered nonzero.

Remark 2.1. In assumption 2.2, $\boldsymbol{\delta} = \boldsymbol{\beta} - \boldsymbol{\beta}_0$ is the difference vector between a $\boldsymbol{\beta}$ and the

true coefficients $\boldsymbol{\beta}_0$, thus we can view $\mathcal{C}$ as a restricted neighborhood of $\boldsymbol{\beta}_0$, i.e.,

$$\mathcal{N}_{\boldsymbol{\beta}_0}^{RE} = \{\boldsymbol{\beta} : \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2 \leq N_n,\ m_n \times \mathrm{card}_B(\boldsymbol{\delta}) \leq n^* = o(n)\}.$$

If $\boldsymbol{\beta} \in \mathcal{N}_{\boldsymbol{\beta}_0}^{RE}$, then by assumption 2.2 we have

$$\frac{(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^T \Phi^T \Phi (\boldsymbol{\beta} - \boldsymbol{\beta}_0)}{n\|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2^2} \geq \gamma_0 \gamma_2^{2s_n} m_n^{-1}.$$

This, together with the bounded variance assumption in assumption 2.3, ensures the restricted strong convexity of the target function, i.e., for a $\boldsymbol{\beta}^* \in \mathcal{N}_{\boldsymbol{\beta}_0}^{RE}$, we have

$$\frac{(\boldsymbol{\beta}^* - \boldsymbol{\beta}_0)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}) \Phi (\boldsymbol{\beta}^* - \boldsymbol{\beta}_0)}{n\|\boldsymbol{\beta}^* - \boldsymbol{\beta}_0\|_2^2} \geq \gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1},\ \forall\ \boldsymbol{\beta} \in \mathcal{N}_{\boldsymbol{\beta}_0}^{RE}. \tag{2.13}$$

2.3 Methodology & Theoretical Properties

We propose a two step procedure for selecting high dimensional additive models with generalized link that has improved convergence rates compared to single stage selection.

2.3.1 First Step: Model Screening

The objective of this step is to recover the true support T of the additive components. Let $\hat{T}$ be a random support given by a model selection procedure and $|\hat{T}|$ be the number of variables selected. A good model selection procedure, especially our second stage estimation with good convergence rates, should satisfy the common screening consistency conditions

$$T \subset \hat{T},\ |\hat{T}| = O(s_n),\ \text{w.p. converging to 1.} \tag{2.14}$$

There have been many variable selection penalization [49, 135, 50, 48] in generalized linear models and [65] in linear additive models this condition holds. Specifically, [50] satisfies the requirements in 2.14 in generalized linear models and [65] also satisfies 2.14 in additive models with identity link function. In this paper, we show that under mild conditions, by maximizing the log-likelihood with group lasso-like penalization, we can select a model that satisfies 2.14. We also provide a rate of convergence of this first step selection.

Define the objective function to be

$$L(\boldsymbol{\beta}; \lambda_{n1}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^T\Phi_i\right) - b\left(\boldsymbol{\beta}^T\Phi_i\right)\right] + \lambda_{n1}\sum_{j=1}^{p_n}\|\boldsymbol{\beta}_j\|_2. \tag{2.15}$$

Let $\hat{\boldsymbol{\beta}}$ be the optimizer for 2.15, i.e.

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{p_n m_n}}{\arg\min}\ L(\boldsymbol{\beta}; \lambda_{n1}).$$

Let $\hat{T} = \text{supp}_B(\hat{\boldsymbol{\beta}})$.

The objective function is negative log-likelihood plus the group lasso penalization term, and the parameters are estimated as the minimizers of the objective function. Here the negative log likelihood function is averaged among the n observations to ensure that it is under the same scale as the penalization function. An analogy is to omit the $1/n$ and consider the re-scaling in the choice of λ. Similarly, since we have same group sizes, instead of multiplying the group size to each penalty term, we consider it in the choice of λ. Two analogies of the objective functions are

$$L(\boldsymbol{\beta}; \lambda_{n1}) = -\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^T\Phi_i\right) - b\left(\boldsymbol{\beta}^T\Phi_i\right)\right] + \lambda_{n1}\sum_{j=1}^{p_n}\|\boldsymbol{\beta}_j\|_2$$

and

$$L(\boldsymbol{\beta};\lambda_{n1}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^T\Phi_i\right) - b\left(\boldsymbol{\beta}^T\Phi_i\right)\right] + \lambda_{n1}\sum_{j=1}^{p_n}\sqrt{m_n}\|\boldsymbol{\beta}_j\|_2$$

[148, 93]. Since they only differ in a multiplier, we can suppress these terms into the selection of λ, which gives the target function we used.

With this group lasso type penalized log-likelihood, the selected model has the following properties.

Theorem 2.1. Consider the model $\hat{T}$ obtained by minimizing 2.15. Under assumptions 1-4, for some constant C and any diverging sequence $\gamma_n > 0$, choose the regularization parameter

$$\lambda_{n1}^b = C\sqrt{m_n}\sqrt{\frac{\gamma_n + \log(p_n m_n)}{n}}$$

for bounded response (i.e., $|y_i| < c$), and the regularization parameter

$$\lambda_{n1}^{ub} = \sqrt{m_n}\gamma_n\sqrt{\frac{\log(p_n m_n)}{n}}$$

for unbounded sub-Gaussian response, as the sample size increases,

(i) With probability tending to 1,

$$|\hat{T}| = O(s_n)$$

(ii) With probability tending to 1,

$$\sum_{j=1}^{p_n}\left\|\boldsymbol{\beta}_j^0 - \hat{\boldsymbol{\beta}}_j\right\|_2^2 = O_P\left(s_n\gamma_2^{-2s_n}\frac{m_n^2\log(p_n m_n)}{n}\right) + O({\lambda_{n1}^b}^2 m_n^2 s_n\gamma_2^{-2s_n})$$

$$+ O(s_n^2 m_n^{1-2d}\gamma_2^{-2s_n})$$

for the bounded response and

$$\sum_{j=1}^{p_n} \left\| \boldsymbol{\beta}_j^0 - \hat{\boldsymbol{\beta}}_j \right\|_2^2 = O_P\left(s_n \gamma_2^{-2s_n} \gamma_n \frac{m_n^2 \log(p_n m_n)}{n} \right) + O(\lambda_{n1}^{ub\,2} m_n^2 s_n \gamma_2^{-2s_n}) + O(s_n^2 m_n^{1-2d} \gamma_2^{-2s_n})$$

for any diverging sequence γ_n and unbounded sub-Gaussian response.

(iii) If $s_n \gamma_2^{-2s_n} m_n^2 \log(p_n m_n)/n << c_{f,n}$ ($s_n \gamma_2^{-2s_n} m_n^2 \gamma_n \log(p_n m_n)/n << c_{f,n}$ in the unbounded case), $\gamma_2^{-2s_n} m_n^2 \lambda_{n1}^2 s_n/m_n << c_{f,n}$ and $s_n^2 m_n^{1-2d} \gamma_2^{-2s_n} << c_{f,n}$, with probability tending to 1, all nonzero coefficients are selected.

The proof of this theorem is given in Appendix A.

Remark 2.2. To avoid estimability issues, here the constants C are selected to be large enough such that the number of parameters to be estimated, i.e., the number of selected nonzero functions $|\hat{T}|$ multiplied by the number of basis function m_n should be less than or equal to n. Moreover, considering the multicollinearity in the design matrix, the constants are chosen such that $m_n \times |\hat{T}| = o(n)$.

Remark 2.3. The additional term γ_n in the convergence rate is due to unboundedness nature of the response variable rather due to non-linear link function.

Remark 2.4. For the special case, linear (Gaussian) additive model our results coincides with the results in [65]. The difference is that we study a fixed design with assumptions on the eigenvalues of the design matrix and they studied a random design with assumption on the distribution of the design matrix. We have put further assumption on the eigenvalue due to the divergence of s_n, the number of nonzero variables. In the special case that s_n

is fixed, our assumptions coincides with the assumptions in [65]. Another difference is that we include a diverging term γ_n that establishes the rate of convergence with probability converging to one.

There are three terms in the convergence rate: the first term comes from the regression it self, the second term comes from shrinkage, and the third term comes from the spline approximation error.

Also noting that in generalized linear models, each additive component is a simple multiplication, i.e. $m_n = 1$. Moreover, we don't have the spline approximation error. So the following corollary is a direct result from theorem 2.1.

Corollary 2.1. In generalized linear models, under the assumptions in theorem 2.1, consider the model $\hat{T}$ obtained by minimizing 2.15, for some constant C and any diverging sequence $\gamma_n > 0$, choose the regularization parameter

$$\lambda_{n1}^{b} = C\sqrt{\frac{\gamma_n + \log(p_n)}{n}}$$

for bounded response (i.e., $|y_i| < c$), and the regularization parameter

$$\lambda_{n1}^{ub} = \gamma_n\sqrt{\frac{\log(p_n)}{n}}$$

for unbounded sub-Gaussian response, we have

(i) With probability tending to 1, we have

$$|\hat{T}| = O(s_n)$$

(ii) With probability tending to 1, we have

$$\left\|\boldsymbol{\beta}^0 - \hat{\boldsymbol{\beta}}\right\|_2^2 = O_P\left(s_n \frac{\log(p_n)}{n}\right) + O(\lambda_{n1}^{b\,2} s_n)$$

for the bounded response case and

$$\left\|\boldsymbol{\beta}^0 - \hat{\boldsymbol{\beta}}\right\|_2^2 = O_P\left(s_n \gamma_n \frac{\log(p_n)}{n}\right) + O(\lambda_{n1}^{ub\,2} s_n)$$

(iii) If $s_n \log(p_n)/n << c_{f,n}$ ($s_n \gamma_n \log(p_n)/n << c_{f,n}$ in the unbounded case), and $\lambda_{n1}^2 s_n << c_{f,n}$, with probability tending to 1, all nonzero coefficients are selected.

Let $\hat{f}_{nj}(x) = \sum_{k=1}^{m_n} \hat{\beta}_{jk}\phi_k(x)$. We can also state the results of the first selection step in terms of functions, which is a direct consequence of theorem 2.1.

Theorem 2.2. Consider the model $\hat{T}$ obtained by minimizing 2.15. Under Assumptions 1-4, for some constant C and any diverging sequence $\gamma_n > 0$, choose the regularization parameter

$$\lambda_{n1}^b = C\sqrt{m_n}\sqrt{\frac{\gamma_n + \log(p_n m_n)}{n}}$$

for bounded response (i.e., $|y_i| < c$), and the regularization parameter

$$\lambda_{n1}^{ub} = \sqrt{m_n}\gamma_n\sqrt{\frac{\log(p_n m_n)}{n}}$$

for unbounded sub-Gaussian response, we have

(i) With probability tending to 1, we have

$$|\hat{T}| = O(s_n)$$

(ii) With probability tending to 1, we have

$$\sum_{j=1}^{p_n} \left\| f_j - \hat{f}_{nj} \right\|_2^2 = O_P\left(s_n \gamma_2^{-2s_n} \frac{m_n \log(p_n m_n)}{n} \right) + O(\lambda_{n1}^{b\,2} m_n s_n \gamma_2^{-2s_n}) + O(s_n^2 m_n^{-2d} \gamma_2^{-2s_n})$$

for the bounded response case and

$$\sum_{j=1}^{p_n} \left\| f_j - \hat{f}_{nj} \right\|_2^2 = O_P\left(s_n \gamma_2^{-2s_n} \gamma_n \frac{m_n \log(p_n m_n)}{n} \right) + O(\lambda_{n1}^{ub\,2} m_n s_n \gamma_2^{-2s_n}) + O(s_n^2 m_n^{-2d} \gamma_2^{-2s_n})$$

for any diverging sequence γ_n.

(iii) If $s_n m_n \gamma_2^{-2s_n} \log(p_n m_n)/n << c_{f,n}$ ($s_n m_n \gamma_2^{-2s_n} \gamma_n \log(p_n m_n)/n << c_{f,n}$ in the unbounded case), $\lambda_{n1}^2 s_n m_n \gamma_2^{-2s_n} << c_{f,n}$ and $s_n^2 m_n^{-2d} \gamma_2^{-2s_n} << c_{f,n}$, with probability tending to 1, all nonzero coefficients are selected.

The proof of this theorem is given in Appendix A.

Remark 2.5. The two theorems together tell us under Assumptions 1-4, by choosing proper γ_n, the functions selected by minimizing the first target function satisfy

$$T \subset \hat{T} \quad \text{and} \quad |\hat{T}| = O(s_n)$$

with probability converging to 1, i.e. we obtained screening consistency.

2.3.2 Second Step: Post Selection

After we have a "good" initial estimator, we use the adaptive group lasso to recover the true model [65] and we are able to achieve selection consistency in probability with some mild assumptions. Adaptive group lasso idea is similar to adaptive lasso [155] which enjoys better theoretical properties than simple lasso. [22] and [34] studied rate of convergence and other asymptotic properties of the adaptive lasso estimator. Define the objective function to be

$$L_a(\boldsymbol{\beta}; \lambda_{n2}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\boldsymbol{\beta}^T\Phi_i\right) - b\left(\boldsymbol{\beta}^T\Phi_i\right)\right] + \lambda_{n2}\sum_{j=1}^{p_n} w_{nj}\|\boldsymbol{\beta}_j\|_2, \tag{2.16}$$

where the weights depend on the screening stage group lasso estimator

$$w_{nj} = + \begin{cases} \|\hat{\boldsymbol{\beta}}_j\|_2^{-1}, & \text{if } \|\hat{\boldsymbol{\beta}}_j\|_2 > 0 \\ \infty, & \text{if } \|\hat{\boldsymbol{\beta}}_j\|_2 = 0 \end{cases} . \tag{2.17}$$

Let $\hat{\boldsymbol{\beta}}_{AGL}$ be the optimizer for 2.16, i.e.

$$\hat{\boldsymbol{\beta}}_{AGL} = \underset{\boldsymbol{\beta}\in\mathbb{R}^{m_n p}}{\arg\min}\, L_a(\boldsymbol{\beta}; \lambda_{n2}).$$

For the choice of weights, the first stage estimators need not to be necessarily the solution of group lasso, rather more general that satisfy following assumptions.

Assumption 2.5. The initial estimator $\hat{\boldsymbol{\beta}}$ is r_n consistent at zero, i.e.,

$$r_n \max_{j\in T^c} \|\hat{\boldsymbol{\beta}}_j - \boldsymbol{\beta}_j^0\|_2 = O_P(1), \tag{2.18}$$

and there exists a constant c_3 such that

$$\mathbb{P}\left(\min_{j\in T}\|\hat{\boldsymbol{\beta}}\|_2 \geq c_3 b_{n1}\right) \to 1, \tag{2.19}$$

where $b_{n1} = \min_{j\in T}\|\boldsymbol{\beta}_j^0\|_2$.

Assumption 2.6. Let $s_n^* = p_n - s_n$ be the number of nonzero components. The tuning parameter λ_{n2} satisfies

$$\frac{\sqrt{\log(s_n^* m_n)}}{n^{1/2}\lambda_{n2} r_n} + \frac{s_n}{\lambda_{n2} r_n m_n^{d+1/2}} + \frac{\lambda_{n2} r_n}{\gamma_n \sqrt{s_n/n}} = o(1) \tag{2.20}$$

for any diverging sequence γ_n.

assumption 2.5 gives the restrictions on the initial estimator. We don't require our initial estimator to be the group lasso estimator, but any initial estimator satisfying assumption 2.5 will be able to make the adaptive group lasso estimator consistently selects and estimates the true nonzero components. However, the rate of convergence of the adaptive group lasso estimator depends on the rate of convergence of the initial estimator, which is assumed to be r_n in assumption 2.5. Moreover, the initial mustn't have a 0 estimation for the nonzero components, otherwise it will mislead the results in the proceeding step. assumption 2.6 put restrictions on the tuning parameter λ_{n2} in the adaptive group lasso step. The first two terms gives the upper bound for λ_{n2} and the third term gives the lower bound. Only with "appropriate" choice of λ_{n2} we can have the selection consistency and estimation consistency.

It worth noting that if we take the group lasso estimator as our initial estimator, assumptions 5 and 6 are automatically satisfied. Specifically, a trivial choice of r_n would

be

$$r_n = O_P^{-1}\left(\sqrt{s_n}\gamma_2^{-s_n}\frac{m_n\sqrt{\log(p_n m_n)}}{\sqrt{n}}\right) + O^{-1}(\lambda_{n1}^b m_n\sqrt{s_n}\gamma_2^{-s_n}) + O^{-1}(s_n m_n^{0.5-d}\gamma_2^{-s_n})$$

for the bounded response and

$$r_n = O_P^{-1}\left(\sqrt{s_n}\gamma_2^{-s_n}\sqrt{\gamma_n}\frac{m_n\sqrt{\log(p_n m_n)}}{\sqrt{n}}\right) + O^{-1}(\lambda_{n1}^{ub} m_n\sqrt{s_n}\gamma_2^{-s_n}) + O^{-1}(s_n m_n^{0.5-d}\gamma_2^{-s_n})$$

for the unbounded case and any diverging sequence γ_n, since we observe that for $j \in T^c$, $\hat{\boldsymbol{\beta}}_j$ is either estimated as zero, or has a rate of convergence to $\boldsymbol{\beta}_j$ bounded by the rate of convergence in theorem (2.1). For equation (2.19), observe that the rate of convergence of the group lasso estimator is higher order infinitesimal of the minimal signal strength of nonzero coefficients, thus taking $c_3 = 0.5$ is sufficient. In assumption 6, with our trivial choice of r_n, we are able to find a range of tuning parameters that satisfy equation (2.20). Therefore, it's reasonable to take the group lasso estimator as an initial estimator for the adaptive group lasso.

Let the notation $\hat{\boldsymbol{\beta}}_n \overset{0}{=} \boldsymbol{\beta}^0$ denote that the sign of each $\hat{\boldsymbol{\beta}}_j$ and $\boldsymbol{\beta}_j^0$ are either both zero or both nonzero. Then we have the following asymptotic properties for the adaptive group lasso estimator.

Theorem 2.3. Assume assumptions 1-6 hold, consider the estimator $\hat{\boldsymbol{\beta}}_{AGL}$ by minimizing 2.16, we have

(i) If $f_{c,n} >> \sqrt{s_n/n}$, the adaptive group lasso consistently selects the true active predic-

tors with probability converging to 1, i.e.,

$$\mathbb{P}\left(\hat{\boldsymbol{\beta}}_{AGL} \overset{0}{=} \boldsymbol{\beta}^0\right) \to 1. \tag{2.21}$$

(ii) The rate of convergence of the adaptive group lasso estimator is given by

$$\sum_{j\in T} \|\hat{\boldsymbol{\beta}}_{AGLj} - \boldsymbol{\beta}_j^0\|_2^2 = O_p\left(s_n \gamma_2^{-2s_n} m_n^2 \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{1-2d}) + O(\lambda_{n2}^2 m_n^2 s_n \gamma_2^{-2s_n})$$

for the bounded response case and

$$\sum_{j\in T} \|\hat{\boldsymbol{\beta}}_{AGLj} - \boldsymbol{\beta}_j^0\|_2^2 = O_p\left(\gamma_n s_n \gamma_2^{-2s_n} m_n^2 \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{1-2d}) + O(\lambda_{n2}^2 m_n^2 s_n \gamma_2^{-2s_n})$$

for the unbounded response case, where γ_n is any diverging sequence.

The proof of this theorem is given in Appendix A. It's interesting to compare the adaptive group lasso results with [138], who studied the asymptotic properties of the adaptive group lasso for generalized linear models. It worth noting that we considered a more general case by allowing the group size to diverge with n, and the eigenvalue to be bounded by sequences that depending on n on a broader domain. In the special case that corresponds to their assumptions, our results (Theorem 4.2) coincides with their results. These results are also true in the special case: generalized linear models. The special case is given in the following corollary.

Corollary 2.2. Consider the generalized linear models with the assumptions same as in theorem 2.3, but with λ_{n2} satisfies

$$\frac{\sqrt{\log(s_n^*)}}{n^{1/2}\lambda_{n2}r_n} + \frac{\lambda_{n2}r_n}{\gamma_n\sqrt{s_n/n}} = o(1),$$

we have

(i) If $f_{c,n} >> \sqrt{s_n/n}$, the adaptive lasso consistently selects the true active predictors with probability converging to 1, i.e.,

$$\mathbb{P}\left(\hat{\boldsymbol{\beta}}_{AL} \overset{0}{=} \boldsymbol{\beta}^0\right) \to 1, \tag{2.22}$$

where $\hat{\boldsymbol{\beta}}_{AL}$ is the adaptive lasso estimator corresponding to the adaptive group lasso estimator in the nonparametric set up.

(ii) The rate of convergence of the adaptive lasso estimator is given by

$$\|\hat{\boldsymbol{\beta}}_{AL} - \boldsymbol{\beta}^0\|_2^2 = O_P\left(s_n\frac{\log(s_n)}{n}\right) + O(\lambda_{n2}^2 s_n) \tag{2.23}$$

for the bounded response case and

$$\|\hat{\boldsymbol{\beta}}_{AL} - \boldsymbol{\beta}^0\|_2^2 = O_P\left(\gamma_n s_n\frac{\log(s_n)}{n}\right) + O(\lambda_{n2}^2 s_n) \tag{2.24}$$

for the unbounded response case, where γ_n is any diverging sequence.

Similar to the group lasso estimator, we also derive results for the non-parametric function estimation.

Let $\hat{f}_{AGLj}(x) = \Phi_j(x)\hat{\boldsymbol{\beta}}_{AGLj}$, then we have the following results.

Theorem 2.4. Assume assumptions 1-6 hold, consider the estimator $\hat{\boldsymbol{\beta}}_{AGL}$ by minimizing 2.16, we have

(i) The adaptive group lasso consistently selects the true active predictors in probability, i.e.,

$$\mathbb{P}\left(\|\hat{f}_{AGLj}(x)\|_2 > 0,\ j \in T \text{ and } \|\hat{f}_{AGLj}(x)\|_2 = 0,\ j \in T^c\right) \to 1. \tag{2.25}$$

(ii) The rate of convergence of the adaptive group lasso estimator is given by

$$\sum_{j\in T} \|\hat{f}_{AGLj} - f_j\|_2^2 = O_p\left(s_n \gamma_2^{-2s_n} m_n \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{-2d}) + O(\lambda_{n2}^2 m_n s_n \gamma_2^{-2s_n})$$

for the bounded response case and

$$\sum_{j\in T} \|\hat{f}_{AGLj} - f_j\|_2^2 = O_p\left(\gamma_n s_n \gamma_2^{-2s_n} m_n \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{-2d}) + O(\lambda_{n2}^2 m_n s_n \gamma_2^{-2s_n})$$

for the unbounded response case, where γ_n is any diverging sequence.

The proof of this theorem is given in Appendix A. The theorems in this section ensures that under mild assumptions, we are able to recover the true model with probability tending to 1 and achieves a rate of convergence better than the initial estimator. Particularly, if the restrictions of n, p_n, m_n and s_n in the previous section satisfy, the group lasso esti-

mator is actually a good initial estimator. Therefore, this two step procedure actually is a complete procedure that gives us a way to do this model selection and estimation on any high-dimensional generalized additive model. However, the procedure is not practically complete without proper selection of the tuning parameter λ. Therefor, we propose a theoretically validated tuning parameter selection in the next section.

The convergence rate for the group lasso estimator is

$$\sum_{j=1}^{p_n} \left\| f_j - \hat{f}_{nj} \right\|_2^2 = O_P\left(s_n \gamma_2^{-2s_n} \frac{m_n \log(p_n m_n)}{n} \right) + O(\lambda_{n1}^b{}^2 m_n s_n \gamma_2^{-2s_n}) + O(s_n^2 m_n^{-2d} \gamma_2^{-2s_n}),$$

while for the adaptive group lasso estimator is

$$\sum_{j \in T} \|\hat{f}_{AGLj} - f_j\|_2^2 = O_p\left(s_n \gamma_2^{-2s_n} m_n \frac{\log(s_n m_n)}{n} \right) + O(\lambda_{n2}^2 m_n s_n \gamma_2^{-2s_n}) + O(s_n^2 \gamma_2^{-2s_n} m_n^{-2d})$$

The regression term differs by the size of candidate set. The price we pay by not knowing the true set is $\log(pm_n)$ in the group lasso step, and becomes $\log(s_n m_n)$ in the adaptive group lasso step, since the initial estimator have recovered a super set of the true set with cardinality $O(s_n)$. The penalty term's difference appears on the tuning parameter, where λ_{n2} is of a smaller order than λ_{n1} with a multiplier of r_n^{-1}. According to our choice of λ_{n2}, it has a trivial upper bound which is of order $O(\lambda_{n1}^2)$. Therefore, the tuning parameter part in the penalty convergence rate term becomes quadratic. The approximation error term is not affected by the adaptive group lasso step.

The adaptive group lasso is important in two reasons: first, with probability tending to 1, this is enable to select the true nonzero components accurately, which is not always the case in group lasso; second, the rate of convergence of the adaptive group lasso estimator is

faster than the rate of convergence of the group lasso estimator. The difference in the leading terms are in the order of r_n^{-1}. This makes the adaptive group lasso estimator to achieve a better error with the same sample size, or the same error with a smaller sample size.

2.4 Tuning Parameter Selection

One important issue in penalized methods is choosing a proper tuning parameter. It is known that the selection results are sensitive to the choice of tuning parameter. The theoretical results only provide the order of the tuning parameter, which is not very useful in practice. The reason is that the order of a sequence describes the limit properties when n goes to infinity. In reality, our n is a fixed number, so we must have a practical instruction of selecting the tuning parameter.

Despite it's importance, there isn't much development for tuning parameter selection in the high dimensional literature. The conventional tuning parameter selection criteria tend to select too many predictors, thus is hard to reach selection consistency. Another reason, especially in group lasso problems, is that the solution path of group lasso is piece-wise nonlinear, which makes the testing procedure even harder. Here, we propose the generalized information criterion (GIC) [151, 52] that supports consistent model selection.

Let $\hat{\boldsymbol{\beta}}^{\lambda}$ be the adaptive group lasso solution with tuning parameter λ. The generalized information criterion is defined as

$$GIC(\lambda) = \frac{1}{n}\{D(\hat{\mu}_{\lambda}; \boldsymbol{Y}) + a_n|\hat{T}_{\lambda}|\}, \tag{2.26}$$

where $D(\hat{\mu}_{\lambda}; \boldsymbol{Y}) = 2\{l(\boldsymbol{Y}; \boldsymbol{Y}) - l(\hat{\mu}_{\lambda}; \boldsymbol{Y})\}$. Here the $l(\boldsymbol{\mu}; \boldsymbol{Y})$ is the log-likelihood function

in equation (2.3) expressed as a function of the expectation $\boldsymbol{\mu}$ and $\boldsymbol{Y}$. $l(\boldsymbol{Y};\boldsymbol{Y})$ represents the saturated model with $\boldsymbol{\mu} = \boldsymbol{Y}$, and $\hat{\mu}_\lambda = b'(\sum_{i=1}^{pn} \hat{f}_j^\lambda(x_{ij})) = b'(\phi\hat{\boldsymbol{\beta}}^\lambda)$ is our estimated expectation when the tuning parameter is λ. The hyper-parameter a_n here is chosen to penalize the size of the model. Using GIC, under proper choice of a_n, we are able to select all active predictors consistently. The result is given in the following theorem.

The importance of the following consistency theorem is that the result in the previous section guarantees that with probability converging to 1, there exists a λ_{n0} that will be able to identify the true model. Therefore, a good choice of a_n will be able to identify the true model with probability converging to 1. For a support $A \subset \{1, ..., p\}$ such that $|A| \leq q$, where $q \geq s_n$ and $q = o(n)$, let

$$I(\boldsymbol{\beta}(A)) = E\left[\log(f^*/g_A)\right] = \sum_{i=1}^{n} \left[b'(\Phi_i\boldsymbol{\beta}^0)\Phi_i^T(\boldsymbol{\beta}^0 - \boldsymbol{\beta}(A)) - b(\Phi_i^T\boldsymbol{\beta}^0) + b(\Phi_i^T\boldsymbol{\beta}(A))\right] \tag{2.27}$$

be the Kullback-Leibler (KL) divergence between the true model and the selected model, where f^* is the density of the true model, and g_A is the density of the model with population parameter $\boldsymbol{\beta}(A)$. Let $\boldsymbol{\beta}^*(A)$ be the model with the smallest KL divergence over all models with support A, and let

$$\delta_n = \inf_{\substack{A \not\supset T \\ |A| \leq q}} \frac{1}{n} I(\boldsymbol{\beta}^*(A))$$

Here we note that if $T \subset A$, the minimizer is automatically $\boldsymbol{\beta}^0$ and thus the KL-divergence is zero. For an underfitted models $T \not\subset A$, δ_n describes how easily one can distinguish the models from the true model by measuring the minimum distance from the true model and the "best estimated models". Later in the theorems we will need to assume lower bounds on δ_n so that we will be able to reach our consistency results. The following theorem proves

that GIC works under mild conditions.

Theorem 2.5. Under assumptions 1-6, suppose that $\delta_n K^{-1} R_n^{-1} \to \infty$, $n\delta_n s_n^{-1} a_n^{-1} \to \infty$ and $a_n \psi^{-1} \to \infty$, where R_n and ψ_n are defined in lemma A.3 and lemma A.4, we have, as $n \to \infty$,

$$\mathbb{P}\{\inf_{\lambda \in \Omega_- \cup \Omega_+} GIC_{a_n}(\lambda) > GIC_{a_n}(\lambda_{n0})\} \to 1, \tag{2.28}$$

where

$$\Omega_- = \{\lambda \in [\lambda_{min}, \lambda_{max}] : T_\lambda \not\supset T\},$$

$$\Omega_+ = \{\lambda \in [\lambda_{min}, \lambda_{max}] : T_\lambda \supset T \text{ and } T_\lambda \neq T\},$$

where T_λ is the set of predictors selected by tuning parameter λ. λ_{min} can be chosen as the smallest λ such that the selected model has size q that satisfies the theorem assumption, and λ_{max} simply corresponds to a model with no variables.

The proof of this theorem is given in Appendix A. In practice, a choice of a_n is proposed to be $m_n \log(\log(n)) \log(p_n)$. We have

Corollary 2.3. Under assumptions 1-6, with choice of $a_n = m_n \log(\log(n)) \log(p_n)$, we have

$$\mathbb{P}\{\inf_{\lambda \in \Omega_- \cup \Omega_+} GIC_{a_n}(\lambda) > GIC_{a_n}(\lambda_{n0})\} \to 1.$$

In our two step procedure, there are two tuning parameters to be selected: λ_{n1} in the group lasso step and λ_{n2} in the adaptive group lasso step. The choice of λ_{n2} is of more importance, since λ_{n1} only serve as the parameter in screening. As long as we have a screening step that satisfies 2.14, we are ready for the adaptive group lasso step. To be simple, we propose to use GIC for selecting both λ_{n1} and λ_{n2}. As a result of the previous

theorem, we are able to reach selection consistency.

2.5 Other Possible Penalty

In this section we briefly discussed other possible penalty functions in the context of high dimensional generalized additive models. In particular we consider the best subset selection, which corresponds to a L_0 penalty, and the best subset with shrinkage which corresponds to a combination of L_0 and L_q, $0 < q \leq 2$ penalty. The procedures have many practical advantages such as dealing with correlated variables, predictive power, better estimation errors etc, these are not well studied neither numerically nor theoretically due to various reasons. The main drawback is the non-convexity of the loss function, thus an algorithm is difficult to build. [85] studied the $L_0 + L_1$ penalty for linear model and support vector machine. They also proposed to use the mixed-integer programming (MIP) method to solve the problem. Recently a few other works on this have brought important development. [92] studied the linear model with these penalties. They gave the algorithm and discussed the advantages under the low signal-to-noise ratio. In this section, we discusses these penalties under the additive set up and compare with our proposed methods.

2.5.1 The L0 Norm Penalty

The L_0 penalty instead of L_1 penalty provides an unbiased estimator, thus we don't have the error (bias) from shrinkage. Consider the problem of optimizing

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} \mathbf{1}_{\|\boldsymbol{\beta}_j\|_2 \neq 0}, \tag{2.29}$$

where $\boldsymbol{y}$ is $n \times 1$ continuous response, $\boldsymbol{\phi}$ is $n \times (p_n m_n)$ basis matrix, and $\boldsymbol{\beta}$ is $(p_n m_n) \times 1$ coefficient vector.

The L_0 penalty penalizes the number of predictors being selected. This formulation is equivalent to

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}\|_2^2, \ s.t. \ \sum_{j=1}^{p} \mathbf{1}_{\|\boldsymbol{\beta}_j\|_2 \neq 0} \leq k \tag{2.30}$$

for some k, which is related to the selection of λ.

Theorem 2.6. Let $\hat{T} = \{j : \|\hat{\boldsymbol{\beta}}_j\|_2 \neq 0\}$, i.e. the indices of predictors that are nonzero in the estimated model, and $\hat{k} = |\hat{T}| = \sum_{j=1}^{p} \mathbf{1}_{\|\hat{\boldsymbol{\beta}}_j\|_2} \neq 0$, i.e., the number of nonzero predictors in the estimated model. If the λ is chosen such that $\hat{k} \geq s_n$ or k is chosen such that $s_n \leq k = O(s_n)$, where s_n is the number of nonzero predictors in the true parameter coefficients, we have

i) The rate of convergence

$$\sum_{j=1}^{p} \|\hat{\boldsymbol{\beta}}_j - \boldsymbol{\beta}_j^0\|_2^2 = O_p\left(s_n \gamma_2^{-2s_n} m_n^2 \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{1-2d}).$$

ii) If furthermore $s_n m_n \log(p_n m_n)/\lambda \to 0$ and $n s_n^2 m_n^{-2d}/\lambda \to 0$, then with probability tending to 1, we have

$$|\hat{k} - s_n| \to 0 \ as \ n \to 0.$$

The proof of this theorem is given in Appendix A. Our group lasso type penalty has rate of convergence

$$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 = O_p\left(s_n \gamma_2^{-2s_n} m_n^2 \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{1-2d}) + O(\lambda_{n2}^2 m_n^2 s_n \gamma_2^{-2s_n}). \tag{2.31}$$

The difference is because there is no shrinkage from the penalty in the L_0 norm, thus we don't have the error term from shrinkage.

2.5.2 The L0 and L1 Norm Penalty

Consider the problem of optimizing

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{j=1}^{p} \|\boldsymbol{\beta}_j\|_2 + \lambda_2 \sum_{j=1}^{p} \mathbf{1}_{\|\boldsymbol{\beta}_j\|_2 \neq 0}, \tag{2.32}$$

where $\boldsymbol{y}$ is $n \times 1$ continuous response, $\boldsymbol{\phi}$ is $n \times (p_n m_n)$ basis matrix, and $\boldsymbol{\beta}$ is $(p_n m_n) \times 1$ coefficient vector.

The L_0 penalty penalizes the number of predictors being selected. This formulation is equivalent to

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}\|_2^2 + \lambda_1 \sum_{j=1}^{p} \|\boldsymbol{\beta}_j\|_2, \ s.t. \ \sum_{j=1}^{p} \mathbf{1}_{\|\boldsymbol{\beta}_j\|_2 \neq 0} \leq k \tag{2.33}$$

for some k, which is related to the selection of λ_2.

In this target function, the selection is done by the L_0 penalty, and we choose λ_1 to be such that the L_1 penalty is dominated by the L_0 penalty, i.e., the L_1 penalty does not do any penalization on top of k predictors, but just provides a shrinkage estimator. Therefore, the estimator will perform well in both selection and estimator, since it possess the advantages of shrinkage estimators as well. Similar to theorem 2.6, we have the following theorem.

Theorem 2.7. Let $\hat{T} = \{j : \|\hat{\boldsymbol{\beta}}_j\|_2 \neq 0\}$, i.e. the indices of predictors that are nonzero in the estimated model, and $\hat{k} = |\hat{T}| = \sum_{j=1}^{p} \mathbf{1}_{\|\hat{\boldsymbol{\beta}}_j\|_2} \neq 0$, i.e., the number of nonzero predictors in the estimated model. If the λ_2 is chosen such that $\hat{k} \geq s_n$ or k is chosen such

that $s_n \leq k = O(s_n)$, where s_n is the number of nonzero predictors in the true parameter coefficients, and λ_1 is chosen such that no more coefficients are shrunk to zero on top of the L_0 penalty, we have

i) The rate of convergence

$$\sum_{j=1}^{p} \|\hat{\boldsymbol{\beta}}_j - \boldsymbol{\beta}_j^0\|_2^2 = O_p\left(s_n \gamma_2^{-2s_n} m_n^2 \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{1-2d}) + O_P(\gamma_2^{-2s_n} m_n^2 \lambda_{n1}^2 s_n).$$

ii) If furthermore $s_n m_n \log(p_n m_n)/\lambda_2 \to 0$, $n s_n^2 m_n^{-2d}/\lambda_2 \to 0$ and $\lambda_1 s_n^{3/2} m_n \log(p_n m_n)/\lambda_2 \to 0$, then with probability tending to 1, we have

$$|\hat{k} - s_n| \to 0 \; as \; n \to 0.$$

The proof of this theorem is given in Appendix A.

Remark 2.6. So far, there hasn't been any method of selecting the tuning parameters that has a theoretical base to ensure the selection consistency. The most common way is to use cross-validation and generalized cross-validation type of methods [92]. Selecting a consistent tuning parameter with some criterion in this case can be a future research interest.

2.6 Numerical Properties

In this section we conduct various empirical exercises to illustrate our theoretically guided method in practice. To optimize the group lasso problems, we apply the algorithm named groupwise-majorization-descent (GMD) by [147], which approximates the convex log-likelihood

part with second order Taylor expansion and solve it with a quadratic function's closed form solution, wrapped in a block coordinate descent algorithm. We made the algorithm in GAM available as a python class, which is available at `https://github.com/KaixuYang/PenalizedGAM`.

As smoothness is a heavy concern in practical GAM computations, we bring the P-spline [42] penalty into the model while implementing the model numerically. The P-spline penalty controls the difference between coefficients of consecutive basis functions, and thus yields smoother spline functions.

Specifically, let $l(\boldsymbol{\beta}; \boldsymbol{X}, \boldsymbol{y})$ be the loss function in section 2.3, either the group lasso loss function or the adaptive group lasso loss function. The loss function with smoothness penalty is defined as

$$l_s(\boldsymbol{\beta}; \boldsymbol{X}, \boldsymbol{y}) = l(\boldsymbol{\beta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda_s \sum_{j=1}^{p} \boldsymbol{\beta}_j^T \boldsymbol{D} \boldsymbol{\beta}_j, \tag{2.34}$$

where

$$\boldsymbol{D} = \begin{bmatrix} 1 & -1 & 0 & . \\ -1 & 2 & -1 & . \\ 0 & -1 & 2 & . \\ . & . & . & . \end{bmatrix}$$

A slightly modified soft-thresholding function is used to handle the combination of group lasso penalty and the smoothness penalty.

2.6.1 Simulated Examples

Here we undertake extensive simulation study to see the performance of our proposed two step selection and estimation approach. We investigate the performance of both uncorre-

lated and correlated covariates and we consider different sample sizes and varying number of predictors in each case. In this section, we will consider three different types of generalized models: the logistic regression (Bernoulli distribution), the Poisson regression (Poisson distribution) and the Gamma regression (Gamma distribution). Through the whole subsection, we choose $l = 4$ which implies a cubic B-spline. We choose $m_n = 9$ for most cases unless stated otherwise. The choice of l and m_n implies that there are $m_n - l = 5$ inner knots, which are evenly placed over the empirical percentiles of the training data. In this subsection, we will compare the performance of the two-step approach with the Lasso [126], the GAMBoost [134] and the GAMSEL [27]. We implement our two-step approach with our own package mentioned above. The Lasso is implemented with the scikit-learn package in python. The GAMBoost and GAMSEL methods are implemented using their packages in R. In the group lasso step, we choose the tuning parameter corresponding to n_g variables, where n_g is the largest number such that $n_g \times m_n <= n$. This choice prevents estimation issues when we have too many parameters. The GIC procedure is applied in the adaptive group lasso step to select tuning parameters. In the GIC procedure, the tuning parameter selection criterion is defined as

$$GIC(\lambda) = \frac{1}{n}\{D(\hat{\mu}_\lambda; \boldsymbol{Y}) + a_n|\hat{T}|\}. \tag{2.35}$$

From our results in the previous section, we choose $a_n = (\log\log n)(\log p)m_n$.

2.6.1.1 Logistic Regression

First, we consider the logistic regression

$$y_i \sim Bernoulli(\theta_i),\ i = 1, ..., n, \tag{2.36}$$

where $\theta_i = logit^{-1}[\alpha + \sum_{j=1}^{p} f_j(x_{ij})]$ and x_{ij} is the $(i,j) - th$ element of the design matrix X.

Example 2.1. We first consider the logistic additive model on an independent design matrix case, where each predictor in X is independent of other predictors. Each element of the design matrix is generated from a $Unif(-1,1)$ distribution. We consider 3 different cases with all n, p and s increasing, which coincides with our theory in section 2.3. Specifically, the three cases are: $n = 100$, $p = 200$ and $s = 3$; $n = 200$, $p = 500$ and $s = 4$; $n = 300$, $p = 3000$ and $s = 5$. A testing sample of size 1000 is generated independently to measure the performance. For all three cases, we have nonzero functions $f_1(x) = 5\sin(3x)$, $f_2(x) = -4x^4 + 9.33x^3 + 5x^2 - 8.33x$ and $f_3(x) = x(1-x^2)\exp(3x) - 4$. These three general terms include a periodic term, a polynomial term and an exponential term. The last two cases have one more function of $f_4(x) = 4x$, a linear term. Finally, the last case has an addition $f_5(x) = 4\sin(-5\log(\sqrt{x+3})$, a complicated composite function. Without loss of generality, the first s functions are set to be nonzero. The constants in the functions are to ensure similar signal strength and smoothness. The other functions $f_{s+1}(x) = ... = f_p(x) = 0$.

Our results focus on NV, the average number of variables being selected; TPR, the true positive rate (what percent of the truly nonzero variables are selected); FPR, the false positive rate (where percent of the zero variables are selected); and PE, the prediction error. In the logistic regression problem, our metric to measure the prediction error will be the misclassification rate, which is also the measurement in [27]. The simulation results are averaged over 100 repetitions.

The simulation results are summarised in table 2.1 on page 54. Compared with the classical method Lasso and the existing GAM methods GAMSEL and GAMBoost, the two-step approach performs the best in terms of both variable selection and estimation in the

high-dimensional set up. The two-step approach performs significantly better in prediction errors. In variable selection, the two-step approach selects the closest number of variables to the ground truth, while keeping the TPR high and FPR low. The existing GAM algorithms have similar TPR but includes too many false positives. The existing GAM algorithms were not intended for very high-dimensional, and thus fails to handle the variable selection and prediction at the same time. As mentioned in [46], the tuning parameter in the Lasso for consistent variable selection is not the same as the tuning parameter for best prediction. We can see this may also be true for the group lasso case, since the estimated nonzero coefficients in the group lasso step are much over-penalized. This also proves that an adaptive group lasso step is important, in terms of both variable selection and prediction.

	n=100 p=200 s=3				n=200 p=500 s=4				n=300 p=3000 s=5			
	NV	TPR	FPR	PE	NV	TPR	FPR	PE	NV	TPR	FPR	PE
Two-step	3.56	.920	.004	.148	4.82	.989	.002	.128	4.92	.968	.000	.122
	(1.19)	(.146)	(.005)	(.027)	(1.02)	(.057)	(.002)	(.018)	(0.535)	(.086)	(.000)	(.018)
Lasso	30.0	.920	.138	.249	64.7	.978	.122	.229	85.2	.816	.027	.211
	(17.9)	(.144)	(.090)	(.041)	(19.2)	(.452)	(.039)	(.024)	(68.3)	(.243)	(.022)	(.024)
GAMSEL	10.1	.820	.039	.241	14.0	.943	.021	.214	33.9	.986	.010	.208
	(11.1)	(.209)	(.055)	(.035)	(12.6)	(.112)	(.025)	(.023)	(27.9)	(.065)	(.009)	(.016)
GAMBoost	44.7	.738	.213	.231	85.4	1.00	.164	.196	138	.996	.044	.186
	(4.84)	(.055)	(.025)	(.027)	(6.88)	(.000)	(.014)	(.018)	(9.64)	(.028)	(.003)	(.015)

Table 2.1: Simulation results for the two-step approach compared with the Lasso, GAMSEL and GAMBoost in the three cases of example 2.1. NV, average number of the variables being selected; TPR, the true positive rate; FPR, the false positive rate; and PE, prediction error (here is the misclassification rate). Results are averaged over 100 repetitions. Enclosed in parentheses are the corresponding standard errors.

In practice, the predictors are sometimes correlated to each other. It's interesting to see how well the procedure performs in correlated predictor cases. Therefore, we also perform the same procedure on correlated predictors.

Example 2.2. In this example, we study the case where the design matrix contains correlated predictors. We generate the data in the following way. First we generate each element of $X_{n\times p}$ independently from $Unif(-1,1)$. Then we generate u from $Unif(-1,1)$, independently from $X_{n\times p}$. Then all columns of X are transformed using $X_j = (X_j + tu)/\sqrt{1+t^2}$. This procedure controls the correlation among predictors through t such that $corr(x_{ik}, x_{ij}) = t^2/(1+t^2)$. Here the simulation is run on $n = 100$, $p = 200$ and $s = 3$. All other set-ups are kept the same as example 2.1. In our example, we choose $t = \sqrt{3/7}$, where the correlation is 0.3 and $t = \sqrt{7/3}$, where the correlation is 0.7.

The results are summarised in table 2.2 on page 56. In the correlated cases, all four methods are influenced, more or less. In terms of variable selection, the two-step approach still has the closest number of selected variables. The methods behave differently in TPR and FPR. GAMBoost tends to have greater numbers in both TPR and FPR, while GAMSEL tends to have both lower numbers. The two-step approach balances between those two methods, while maintaining the smallest FPR among all methods. In terms of the prediction error, the two-step approach significantly beats the other methods. The results show the good performance of the two-step approach, and again emphasize that the adaptive group lasso step is necessary for better selection and estimation.

This underselection for correlated predictors has been an issue for the lasso and adaptive lasso methods. For nonparametric additive models, [65] found the same issue when dealing with correlated predictors. Also the NIS proposed by [45] did not perform well in correlated predictors compared to uncorrelated case. Our two-step approach is not affected too much

	Cor=0.3				Cor=0.7			
	NV	TPR	FPR	PE	NV	TPR	FPR	PE
Two-step	2.82	.753	.003	.171	2.05	.557	.002	.174
	(.994)	(.229)	(.004)	(.033)	(.829)	(.170)	(.003)	(.022)
Lasso	37.0	.690	.176	.312	21.9	.327	.103	.288
	(38.2)	(.259)	(.194)	(.069)	(37.9)	(.291)	(.193)	(.047)
GAMSEL	15.4	.573	.069	.342	12.5	.397	.057	.264
	(16.0)	(.285)	(.079)	(.065)	(9.15)	(.271)	(.044)	(.033)
GAMBoost	44.2	.977	.209	.268	33.7	.860	.158	.203
	(5.21)	(.085)	(.026)	(.033)	(4.52)	(.178)	(.014)	(.026)

Table 2.2: Simulation results for the two-step approach compared with the Lasso, GAMSEL and GAMBoost in example 2.2 with correlation 0.3 and 0.7 for $n = 100$, $p = 200$ and $s = 3$. NV, average number of the variables being selected; TPR, the true positive rate; FPR, the false positive rate; and PE, prediction error (here is the misclassification rate). Results are averaged over 100 repetitions. Enclosed in parentheses are the corresponding standard errors.

with the correlation, in terms of both variable selection and prediction.

It also happens in the real world that the signal strength is low. Therefore, it is interesting to consider a cases where we have lower signal strength than in example 2.1.

Example 2.3. In the next example, we reduce the signal strength of example 2.1 by a factor of 2, while all other assumptions are kept the same. The results are shown in 2.3 on page 57. From the table we see that minimal signal strength is an important factor to the performance of variable selection in the generalized models. The performance is impacted by the signal strength for all models. The two-step approach still have the closest number of nonzero variables to the ground truth. Though the true positive rate is lower than that of the Lasso or the GAMBoost, the latter two methods have too many false positives. The Lasso or GAMBoost selects too many variables and should not be considered as good variable selection methods. Moreover, the prediction error of the two-step approach remain the best among all four methods.

	NV	TPR	FPR	PE
Two-step	3.91	.703	.009	.218
	(2.05)	(.240)	(.009)	(.033)
Lasso	30.0	.770	.142	.258
	(30.5)	(.304)	(.154)	(.036)
GAMSEL	15.3	.510	.070	.377
	(18.0)	(.266)	(.090)	(.054)
GAMBoost	50.3	.980	.240	.308
	(5.11)	(.079)	(.026)	(.028)

Table 2.3: Simulation results for the two-step approach compared with the Lasso, GAMSEL and GAMBoost in example 2.3, with $n = 100$, $p = 200$, $s = 3$ and signal strength reduced. NV, average number of the variables being selected; TPR, the true positive rate; FPR, the false positive rate; and PE, prediction error (here is the misclassification rate). Results are averaged over 100 repetitions. Enclosed in parentheses are the corresponding standard errors.

2.6.1.2 Other link functions

In this subsection, we study the performance of the two-step approach numerically on the Poisson regression and Gamma regression. In the Poisson regression, we have

$$y_i \sim Poisson(\theta_i),\ i = 1, ..., n, \tag{2.37}$$

where $\theta_i = \exp[\alpha + \sum_{j=1}^{p} f_j(x_{ij})]$ and x_{ij} is the $(i, j) - th$ element of the design matrix X. In the Gamma regression, we have

$$y_i \sim Gamma(\theta_i, \phi),\ i = 1, ..., n, \tag{2.38}$$

where $\theta_i = \exp[\alpha + \sum_{j=1}^{p} f_j(x_{ij})]$ and x_{ij} is the $(i, j) - th$ element of the design matrix X. The dispersion parameter ϕ is assumed to be known. Without loss of generality, we take $\phi = 1$.

Example 2.4. In this example, we keep the same set up as in example 2.1 to generate

the design matrix, and use the Poisson distribution/Gamma distribution above to generate response variables. All other parameters are kept the same as in example 2.1, but the signal strength is set to 1/4 of the original signal strength, and we have $n = 100$, $p = 200$ and $s = 3$. We compare the two-step approach with generalized linear models (GLM) and the GAMBoost. Note that the GAMSEL only support Gaussian and Binomial link, thus is not used as a comparison here. The GAMBoost only supports generalized models with canonical link. The canonical link for Gamma regression suffers from the risk that the mean might fall outside of its range, thus the canonical link is not useful in practice. Therefore, we only use GAMBoost in Poisson regression as a comparison. Our algorithm works for both Gamma regression and Poisson regression, and to the best of our knowledge, is the only public algorithm that supports both in the high-dimensional case. The GLMs are run with the scikit-learn package in python.

The results are showed in table 2.4. We see the two-step approach works significantly better than the linear model, and than the GAMBoost in the Poisson regression case, except for the true positive rate. The GAMBoost has a perfect true positive rate, which is slightly better than that of our two-step approach. However, the same issue as before is that it selected too many variables and make the false positive rate much higher than tolerable. Moreover, the prediction performance on the two-step approach is also in the first place in both cases.

2.6.2 Real Data examples

In this section, we use three real data examples to illustrate our procedure. In the first example, we consider the case $n > p$ in the classification set up, in the second example, we consider the high-dimensional set up $n < p$ in the classification set up, and in the third

	Poisson Regression				Gamma Regression			
	NV	TPR	FPR	PE	NV	TPR	FPR	PE
Two-step	4.30	.930	.008	2.34	3.57	.997	.003	14.4
	(1.51)	(.172)	(.009)	(.703)	(0.98)	(.033)	(.005)	(19.5)
Lasso	13.4	.867	.054	3.51	12.5	.887	.048	42.3
	(9.79)	(.189)	(.050)	(.403)	(7.72)	(.196)	(.039)	(11.5)
GAMBoost	82.1	1.00	.401	15.4	NA	NA	NA	NA
	(4.27)	(.000)	(.022)	(2.12)				

Table 2.4: Simulation results for the two-step approach compared with the Lasso, GAMSEL and GAMBoost in example 2.4 for Poisson regression and Gamma regression with $n = 100$, $p = 200$ and $s = 3$. NV, average number of the variables being selected; TPR, the true positive rate; FPR, the false positive rate; and PE, prediction error (here is the misclassification rate). Results are averaged over 100 repetitions. Enclosed in parentheses are the corresponding standard errors. The GAMBoost method does not support Gamma regression with non-canonical link function, while the canonical link falls outside of range, therefore it does not support Gamma regression.

example, we consider a Gamma regression.

Example 2.5. In this example, we use the data set in example 1 of [55], the spam data as an example of the case $n > p$. The data set is available at `https://web.stanford.edu/~hastie/ElemStatLearn/data.html`. This data set has been studied in many different contexts with the objective being to predict whether an email is a spam or not based on a few features of the emails. There are $n = 4601$ observations, among which 1813 (39.4%) are spams. There are $p = 57$ predictors, including 48 continuous real $[0, 100]$ attributes of the relative frequency of 48 'spam' words out of the total number of words in the email, 6 continuous real $[0, 100]$ attributes of the relative frequency of 6 'spam' characters out of the total number of characters in the email, 1 continuous real attribute of average length of uninterrupted sequences of capital letters, 1 continuous integer attribute of length of longest uninterrupted sequence of capital letters, and 1 continuous integer attribute of total number of capital letters in the e-mail. The data was first log transformed, since most of the predictors have long-tailed distribution, as mentioned in [55]. They were then centered and

standardised.

The data was split into a training data set with 3067 observations and a testing data set with 1534 observations. We choose order $l = 4$ which implies a cubis B-spline. We choose $m_n = 15$, which implies there are 11 inner knots, evenly placed over the empirical percentiles of the data. We compare the result with the logistic regression with Lasso penalty, the support vector machine (SVM) with Lasso penalty, and the sparse group lasso neural network (SGLNN, [53], see also [146]). The Lasso and SMV are implemented with the skikit-learn module in python, and the SGLNN is implemented with the algorithm in the paper in python. By changing the tuning parameter or stopping criterion, we get estimations with different sparsity levels. All results are averaged over 50 repetitions. The classification error with different level of sparsity is shown in figure 2.1 on page 61. The two-step approach and the neural network perform better than the linear methods, which indicates a non-linear relationship. The two-step approach has best accuracy 0.944, while the neural network has best accuracy 0.946. The neural network performs a little better than the two-step approach due to its ability to model the interactions among predictors, but this difference is not significant. However, neural network has no interpretation and takes longer to train. All four methods have performance increase as more predictors are included, which indicates that all predictors contributes to some effect to the prediction. However, we are able to reach more than 0.9 accuracy with only 15 predictors included. With the GIC criterion, the two-step approach selects 14.6 ± 1.52 predictors, with an average accuracy of 0.914 ± 0.015. The most frequently selected functions are shown in figure 2.2 on page 62, which also shows that these functions are truly non-linear. The plots are of the original functions, i.e., before the logarithm transformation. The estimated functions are close to the results in [55], Chapter 9, with slight scale difference due to different penalization. The results show that the additive

model by the adaptive group lasso is more suitable for this data than linear models.

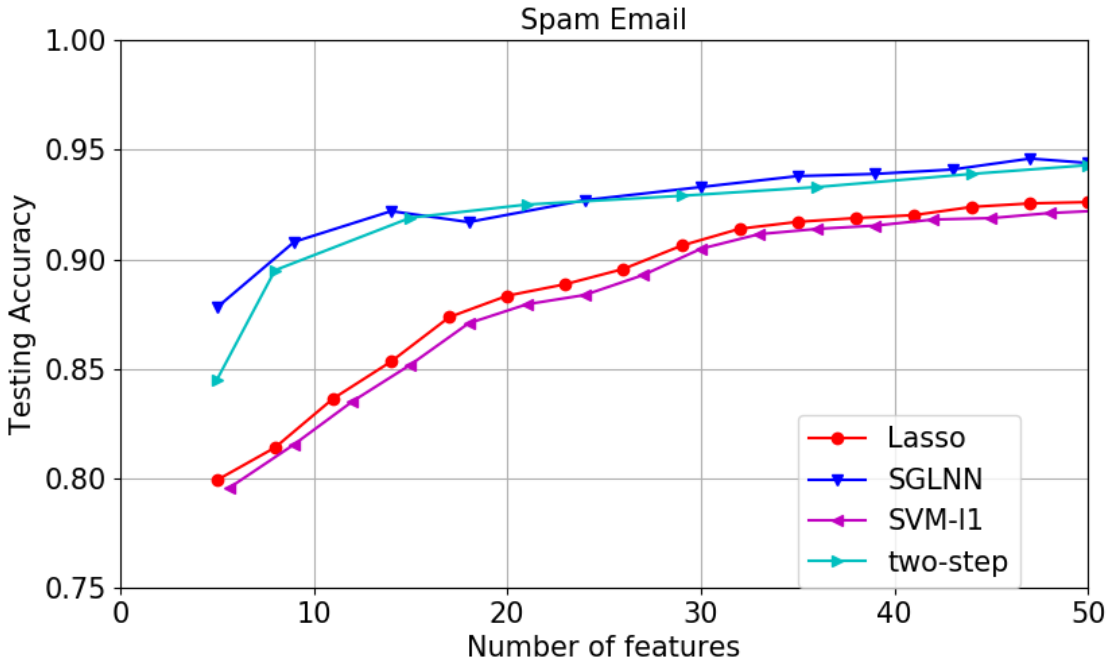

Figure 2.1: The classification accuracy against the number of nonzero variables measured on a testing set for example 2.5 over 50 repetitions. The two-step approach, the logistic regression with Lasso, the l_1 norm penalized SVM and the sparse group lasso neural network are included in comparison.

Example 2.6. For high-dimensional classification example, we use the prostate cancer gene expression data described in http://featureselection.asu.edu/datasets.php. The data set has a binary response. 102 observations were studied on 5966 predictor variables, which indicates that the data set is really a high dimensional data set. The responses have values 1 (50 sample points) and 2 (52 sample points), where 1 indicates normal and 2 indicates tumor. All predictors are continuous predictors, with positive values.

To see the performance of our procedure, we ran 100 replications. In each replication, we randomly choose 76 of the observations as training data set and the rest 26 observations as testing data set. We choose order $l = 4$ which implies a cubis B-spline. We choose $m_n = 9$, which implies there are 5 inner knots, evenly placed over the empirical percentiles of the data.

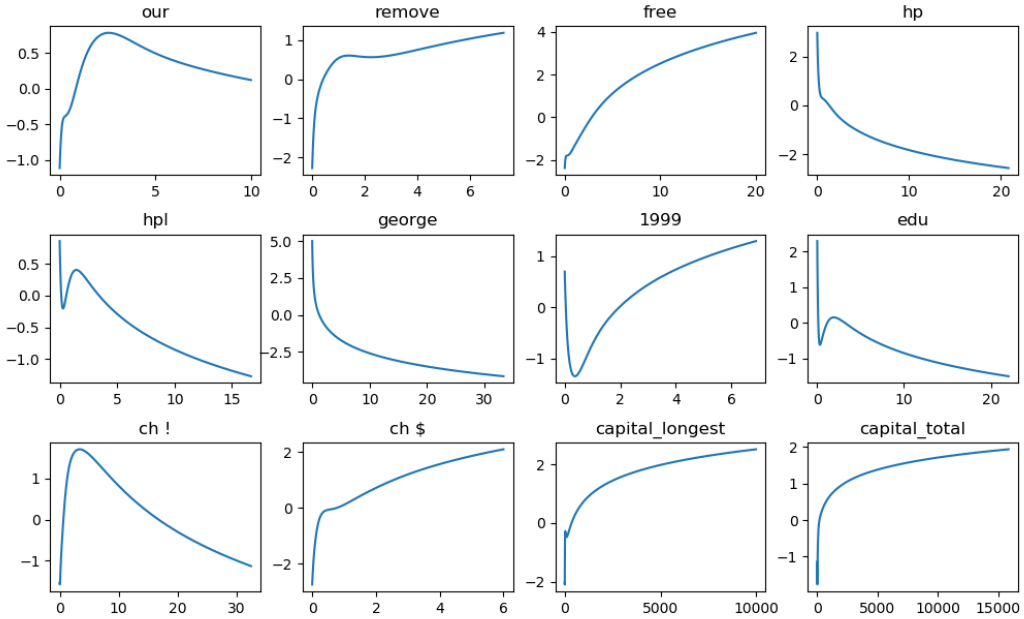

Figure 2.2: The estimated functions for the most frequently selected functions for example 2.5.

Similar to the last example, we compare the result with the logistic regression with Lasso penalty, the SVM with Lasso penalty, and SGLNN. The classification error with different level of sparsity is shown in figure 2.3 on page 63. From the figure we see that compared with linear methods such as the logistic regression or support vector machine, the non-parametric approaches converges faster. The two-step approach reaches a testing accuracy of 0.945 when around 15 variables are included in the model, while the linear methods need over 30 variables to reach competitive results. Compared with neural network, the two-step approach is easier to implement and performs stabler. A drawback of the non-parametric methods is to easily overfit small sample, and that's the reason the performance drops as too many variables entered the model. With the GIC criterion, the two-step approach selects 3.25 ± 1.67 predictors, with an average accuracy of 0.914 ± 0.016. To show the non-linear relationship, figure 2.4 on page 64 shows the estimated functions for the 6 most frequently

selected variables.

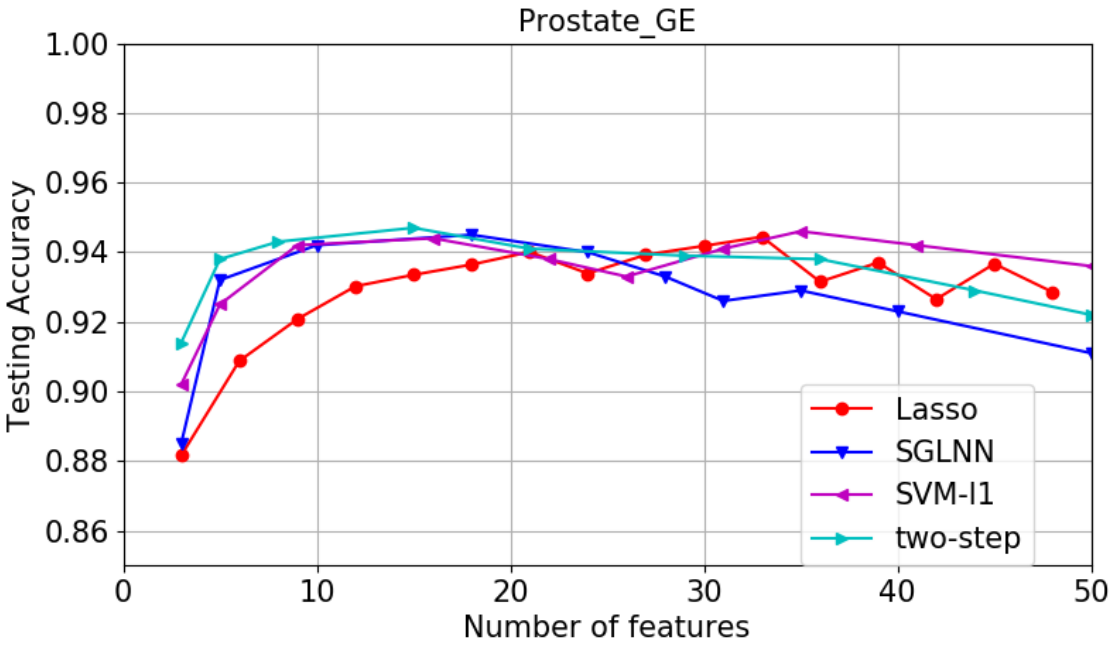

Figure 2.3: The classification accuracy against the number of nonzero variables measured on a testing set for example 2.6 over 500 repetitions. The two-step approach, the logistic regression with Lasso, the l_1 norm penalized SVM and the sparse group lasso neural network are included in comparison.

Example 2.7. In this example, we investigate the performance of the two-step approach on Gamma regression. The data set is from National Oceanic and Atmospheric Administration (NOAA). We use the storm data, which includes the occurrence of storms in the United States with the time, location, property damage, a narrative description and etc. Here we only take the data in Michigan from 2010 to 2018 and keep the narrative description as our predictor variable and the property damage as our response variable. The description is in text, therefore we applied wording embedding algorithm Word2vec [96] to transform each description into a numeric representation vector of length $p = 701$, similar word embedding preprocessing can be found in [77]. The response variable property damage has a long tail distribution, thus we use a Gamma regression here. After removing outliers, the data

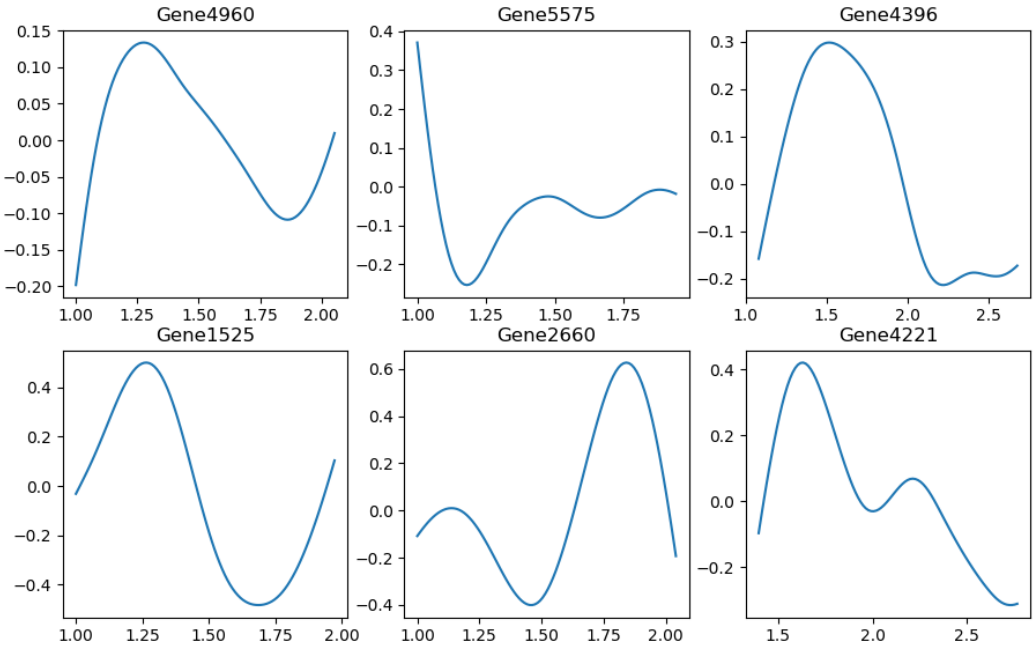

Figure 2.4: The estimated functions for the most frequently selected functions ordered by descending in frequency for example 2.6.

set contains 3085 observations. In order to study the high-dimensional case, we randomly sample 10% of the observations as our training data ($n = 309$) and the rest are used for validation. Moreover, the response is normalized with the location and scale parameters of gamma distribution.

To see the performance of our procedure, we ran 50 replications. We choose order $l = 4$ which implies a cubis B-spline. We choose $m_n = 9$, which implies there are 5 inner knots, evenly placed over the empirical percentiles of the data. Since there's limited libraries available in variable selection under high-dimensional set up, we compare the two-step approach with the linear regression with Lasso on a logarithm transformation on the response variable. The prediction error with different level of sparsity is shown in figure 2.5 on page 65. With the GIC criterion, the two-step approach selects 34.45 ± 3.52 predictors, with an average

MSE of 0.004334 ± 0.000115.

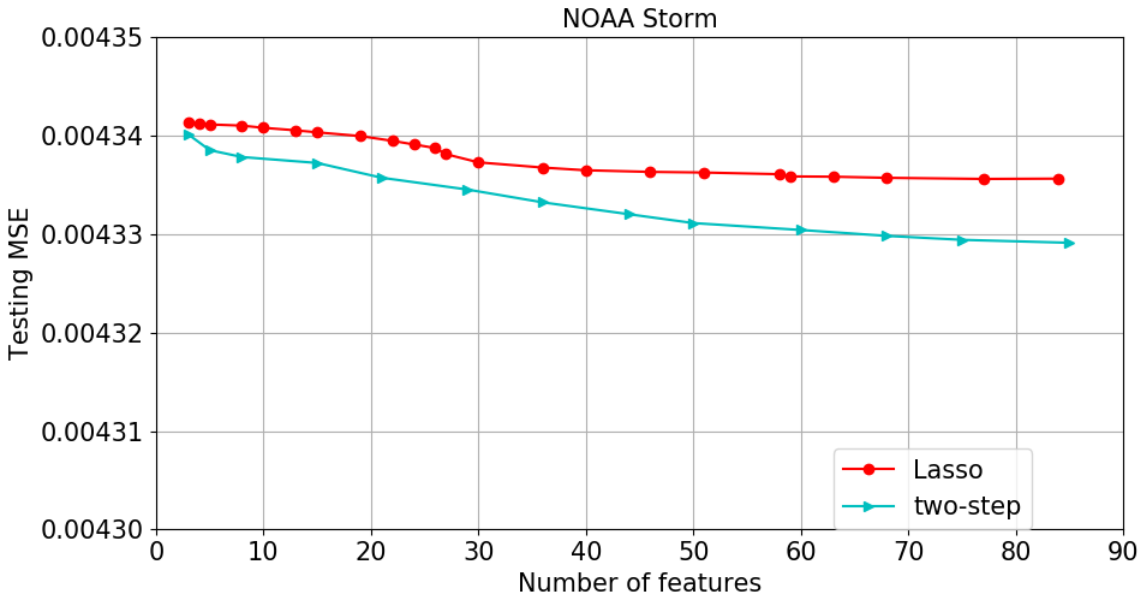

Figure 2.5: The testing MSE against the number of nonzero variables measured on a testing set for example 2.7 over 50 repetitions. The two-step approach and logarithm transformation with the Lasso are included in comparison.

2.7 Discussion

In this chapter, we considered ultra high-dimensional ($\log p_n = O(n^\rho)$) generalized additive model with a diverging number of nonzero functions ($s_n \to 0$ as $n \to \infty$). After using basis expansion on the nonparametric functions, we used two step procedures—group lasso and adaptive group lasso to select the true model. We have proved the screening consistency of the group lasso estimator and the selection consistency of the adaptive group lasso estimator. The rates of convergence of both estimators were also derived, which proved that the adaptive group lasso does have an improvement on the estimator. The whole paper provides a solid foundation for the existing methods. Finally we proved that under this nonparametric set up, the generalized information criterion (GIC) is a good way to select the tuning parameter

that consistently selects the true model.

In this paper, we used a fixed design on the data matrix $\boldsymbol{X}$. A random design on $\boldsymbol{X}$ could be considered, i.e., X has a continuous distribution function $f_X(X)$ on its interval $[a, b]$, however, extra assumptions such as the boundedness of the density function are needed to reach the same result. Also we proved the selection consistency of the GIC procedure on the adaptive group lasso estimator, conditioning that the initial estimator satisfies 2.14, which is possessed by the group lasso procedure with probability tending to 1. However, the theory of screening consistency for the group lasso estimator is still to be established. This is a challenging problem, since there doesn't have to exist a tuning parameter that gives selection consistency in the group lasso procedure, but this is an interesting problem that deserves further investigation. We also discussed the subset selection and subset selection with shrinkage under our set up. The theoretical investigation suggests the other penalty functions may not have clear advantages over the proposed procedure.

Moreover, the heteroskedastic error case is also attracting in high-dimensional GAM. The square root Lasso [10] has been proved to overcome this issue, however, it hasn't been extended to the non-parametric set up. It could be interesting to apply square root Lasso on the GAM to incorporate this case. This is an demanding topic that deserves further investigation as well.

Chapter 3

Sparse Neural network

As we mentioned in the introduction chapter, the generalized additive model is appropriate in the additive case. If the variables interact with each other, it's better to use a multi-variate approximation, the neural network. In this chapter, we will study the sparse group lasso penalized shallow neural network. We will show that under certain assumptions, this model have classification risk tending to the optimal risk, the Bayes risk. Simulations and real data examples are used to support the arguments.

3.1 Introduction

High-dimensional models with correlated predictors are commonly seen in practice. Most statistical models work well either in low-dimensional correlated case, or in high-dimensional independent case. There are few methods that deal with high-dimensional correlated predictors, which usually have limited theoretical and practical capacity. Neural networks have been applied in practice for years, which have a good performance under correlated predictors. The reason is that the non-linearity and interactions are brought in by the activation functions and nodes in the hidden layers. A universal approximation theorem guarantees that a single-layer artificial neural network is able to approximate any continuous function with an arbitrarily small approximation error, provided that there is a large enough number of hidden nodes in the architecture. Thus the ANN handles the correlation and interactions

automatically and implicitly. A popular machine learning application that deals with this type of dependency is the spatio-temporal data, where the traditional statistical methods model the spatial covariance matrix of the predictors. However, by artificial neural networks, working with this big covariance matrix can be avoided. Moreover, artificial neural networks also have good performance in computer vision tasks in practice.

A main drawback of neural networks is that it requires a huge number of training sample due to large number of inherent parameters. In some application fields, such as clinical trials, brain imaging data analysis and some computer vision applications, it's usually hard to obtain such a large number of observations in the training sample. Thus there is a need for developing high-dimensional neural networks with regularization or dimension reduction techniques. It is known that l_1 norm regularization [126] shrinks insignificant parameters to zero. Commonly used regularization includes l_p norm regularization, for example, see the keras package [26]. l_p norm regularization with $p \geq 2$ controls the model sensitivity [74]. On the other hand l_p norm regularization with $p < 2$, where people usually take $p = 1$ for computation efficiency, does not encourage group information. The group lasso regularization [148] yields group-wise sparseness while keeping parameters dense within the groups. A common regularization used in high-dimensional artificial neural network is the sparse group lasso by [114], see for example [53], which is a weighted combination of the lasso regularization and the group lasso regularization. The group lasso regularization part penalizes the input features' weights group-wisely: a feature is either selected or dropped, and it is connected to all nodes in the hidden layer if selected. The lasso part further shrinks some weights of the selected inputs features to zero: a feature does not need to be connected to all nodes in the hidden layer when selected. This penalization encourages as many zero weights as possible. Another common way to overcome the high-dimensionality is to add

dropout layers [117]. Randomly setting parameters in the later layers to zero keeps less non-zero estimations and reduces the variance. Dropout layer is proved to work well in practice, but no theoretical guarantee is available. [82] considers a deep network with the combination of regularization in the first layer and dropout in other layers. With a deep representation, neural networks have more approximation power which works well in practice. They propose a fast and stable algorithm to train the deep network. However, no theoretical guarantee is given for the proposed method other than practical examples.

On the other hand, though widely used, high-dimensional artificial neural networks still do not have a solid theoretical foundation for statistical validation, especially in the case of classification. Typical theory for low-dimensional ANNs traces back to the 1990s, including [33, 8, 119, 4]. The existing results include the universal approximation capabilities of single layer neural networks, the estimation and classification consistency under the Gaussian assumption and 0-1 loss in the low dimensional case. These theory assumes the 0-1 loss which is not used nowadays and are not sufficient for high-dimensional case as considered here. Current works focus more on developing new computing algorithms rather building theoretical foundations or only have limited theoretical foundations. [53] derived a convergence rate of the log-likelihood function in the neural network model, but this does not guarantee the universal classification consistency or the convergence of the classification risk. The convergence of the log-likelihood function is necessary but not sufficient for the classification risk to converge. In this paper, we obtained consistency results of classification risk for high-dimensional artificial neural networks. We derived the convergence rate for the prediction error, and proved that under mild conditions, the classification risk of high-dimensional artificial neural network classifier actually tends to the optimal Bayes classifier's risk. This type of property has been established on other classifiers such as KNN [23], which derived the

result that the classification risk of KNN tends to the Bayes risk, LDA [79], which derives the classification error rate under Gaussian assumptions and etc. Popular tasks like analyzing MRI data and computer vision models were also included in these research papers, and we applied the high-dimensional neural network to these demanding tasks as well.

In section 3.2, we will formulate the problem and the high-dimensional neural network formally. In section 3.3, we state the assumptions and the main consistency result. In section 3.4, we compared the high-dimemsional neural network with other neural network variable selection approaches. In section 3.5, we apply the high-dimensional neural network in three different aspects of examples: the gene data, the MRI data and the computer vision data. In section 3.6, further ideas will be discussed.

3.2 The Binary Classification Problem

Consider the binary classification problem

$$P(Y = 1|X = x) = f(\boldsymbol{x}),\ P(Y = 0|X = x) = 1 - f(\boldsymbol{x}),$$

where $\boldsymbol{x} \in \mathbb{R}^p$ is the feature vector drawn from the feature space according to some distribution $P_{\boldsymbol{X}}$, and $f(\cdot) : \mathbb{R}^p \to \mathbb{R}$ is some continuous function. Note here that, in the function $f(\boldsymbol{x})$, there can be any interactions among the predictors in $\boldsymbol{x}$, which ensures the possibility to handle correlated predictors. Let $P_{\boldsymbol{X},Y}$ be the joint distribution on $(\boldsymbol{X}, Y)$, where $\boldsymbol{X} \in \mathbb{R}^p$ and $Y \in \{0, 1\}$. Here p could be large, and may be even much larger than the training sample size n. To study the theory, we assume p has some relationship with n, for example, $p = O(\exp(n))$. Therefore, p should be written as p_n, which indicates the

dependency. However, for simplicity, we suppress the notation p_n and denote it with p.

For a new observation $\boldsymbol{x}_0 \in \mathbb{R}^p$, the Bayes classifier, denoted $C^*(\boldsymbol{X})$, predicts 1 if $f(\boldsymbol{x}) \geq p_s$ and 0 otherwise, where $p_s \in (0,1)$ is a probability threshold, which is usually chosen as $1/2$ in practice. The Bayes classifier is proved to minimize the risk

$$R(C) = \int_{\mathbb{R}^p \times \{0,1\}} \mathbb{1}_{\{C(\boldsymbol{X}) \neq Y\}} dP_{\boldsymbol{X},Y}. \tag{3.1}$$

However, the Bayes classifier is not useful in practice, since $f(\boldsymbol{x})$ is unknown. Thus a classifier is to be found based on the observations $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$, which are drawn from $P_{\boldsymbol{X},Y}$. A good classifier based on the sample should have its risk tend to the Bayes risk as the number of observations tends to infinity, without any requirement for its probability distribution. This is the so-called universal consistency.

Multiple methods have been adopted to estimate $f(\boldsymbol{x})$, including the logistic regression (a linear approximation), generalized additive models (GAM, a non-parametric nonlinear approximation which do not take interactions into account), neural networks (a complicated structure which is dense in continuous functions space) and etc. The first two methods usually work in practice with good theoretical foundation, however, they sometimes fail to catch the complicated dependency among the feature vector $\boldsymbol{x}$ in a wide range of applications (brain images, computer visions, spatial data analysis). The neural network structure is proved to be able to capture this dependency implicitly without explicitly specifying the dependency hyper-parameters. Consider a single layer neural network model with p predictor variables. The hidden layer has m_n nodes, where m_n may be a diverging sequence depending on n. Similar to p_n, we suppress m_n as m.

For an input vector $\boldsymbol{x} \in \mathbb{R}^p$, its weight matrix $\boldsymbol{\theta} \in \mathbb{R}^{p \times m}$ and its hidden layer intercept

vector $\boldsymbol{t} \in \mathbb{R}^m$, let the vector $\boldsymbol{\xi} \in \mathbb{R}^m$ be the corresponding values in the hidden nodes, which is defined as

$$\xi_k = t_k + \boldsymbol{\theta}_k^T \boldsymbol{x},\ k = 1, ..., m.$$

Let $\psi(\cdot)$ be an activation function, then the output for a given set of weight $\boldsymbol{\beta}$ is calculated by

$$b + \boldsymbol{\beta}^T \boldsymbol{\psi}(\boldsymbol{\xi}),$$

where the function $\boldsymbol{\psi}(\cdot)$ is the function $\psi(\cdot)$ being applied element-wisely. We have a wide range of choices for the activation function. [78] proved that as long as the activation is not algebraic polynomials, the single layer neural network is dense in the continuous function space, thus can be used to approximate any continuous function. This structure can be considered as a model which, for a given activation function $\psi(\cdot)$, maps a $p \times 1$ input vector to an real-valued output

$$\eta_{(\boldsymbol{\theta},\boldsymbol{t},\boldsymbol{\beta},b)}(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{\psi}(\boldsymbol{t} + \boldsymbol{\theta}^T \boldsymbol{x}) + b = \sum_{k=1}^{m} \beta_k \psi(t_k + \boldsymbol{\theta}_k^T \boldsymbol{x}) + b,$$

where $\eta_{(\boldsymbol{\theta},\boldsymbol{t},\boldsymbol{\beta},b)}(\boldsymbol{x}) \in \mathbb{R}$ is the output of the single hidden layer neural network with parameter $(\boldsymbol{\theta}, \boldsymbol{t}, \boldsymbol{\beta}, b)$. Applying the logistic function $\sigma(\cdot)$, $\sigma(\eta_{(\boldsymbol{\theta},\boldsymbol{t},\boldsymbol{\beta},b)}(\boldsymbol{x})) \in (0, 1)$ as an approximation of $f(\boldsymbol{x})$ with parameters $(\boldsymbol{\theta}, \boldsymbol{t}, \boldsymbol{\beta}, b)$

$$P(Y = 1|X = x) \approx \sigma(\eta_{(\boldsymbol{\theta},\boldsymbol{t},\boldsymbol{\beta},b)}(\boldsymbol{x})),\ P(Y = 0|X = x) \approx 1 - \sigma(\eta_{(\boldsymbol{\theta},\boldsymbol{t},\boldsymbol{\beta},b)}(\boldsymbol{x})),$$

where $\sigma(\cdot) = \exp(\cdot)/[1+\exp(\cdot)]$. According to the universal approximation theorem, see [33], with a big enough m, the single layer neural network is able to approximate any continuous

function with a quite small approximation error.

By [8], assuming that there is a Fourier representation of $f(\boldsymbol{x})$ of the form $f(\boldsymbol{x}) = \int_{\mathbb{R}^p} e^{i\boldsymbol{\omega}^T\boldsymbol{x}}\tilde{F}(d\boldsymbol{\omega})$, let $\Gamma_{B,C} = \{f(\cdot) : \int_B \|\boldsymbol{\omega}\|_2|\tilde{f}(d\boldsymbol{\omega}) < C\}$ for some bounded subset of $\mathbb{R}^p$ containing zero and for some constant $C > 0$. Then for all functions $f \in \Gamma_{B,C}$, there exists a single layer neural network output $\eta(\boldsymbol{x})$ such that $\|\eta(\boldsymbol{x}) - f(\boldsymbol{x})\|_2 = O(1/\sqrt{m})$ on B. Later [113] generalizes the result by relaxing the assumptions on the activation function and improved the rate of approximation by a logarithmic factor. They showed that on a bounded domain $\Omega \subset \mathbb{R}^p$ with Lipschitz boundary, assuming $f \in H^r(\Omega)$ satisfies $\gamma(f) = \int_{\mathbb{R}p}(1 + |\omega|)^{m+1}|\hat{f}_e(\omega)|d\omega < \infty$ for some extension $f_e \in H^r(\mathbb{R}^p)$ with $f_{e|\Omega}$, if the activation function $\sigma \in W^{r,\infty}(\mathbb{R})$ is non-zero and satisfies the polynomial decay condition $|(D^k\sigma(t))| \leq C_r(1 + |t|)^{-s}$ for some $0 \leq k \leq r$ and some $s > 1$, we have

$$\inf_{f_m \in NeuNet_m} \|f - f_m\|_{H^r(\Omega)} \leq C(s, r, \Omega, \sigma)\gamma(f)m^{-1/2},$$

where the norm is in Sobolev space of order r, and $C(s, r, \Omega, \sigma)$ is a function of s, r, Ω and σ only. Both results ensure the good approximation property of single layer neural network, and the convergence rate is independent of the dimension of $\boldsymbol{x}$, p, as long as f has a Fourier transform which decays sufficiently fast.

Towards building the high-dimensional ANN, we start by formalizing the model. Let $\boldsymbol{X}$ be a $n \times p$ design or input matrix,

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_{(1)} & \cdots \boldsymbol{x}_{(p)} \end{bmatrix},$$

let $\boldsymbol{y}$ be a $n \times 1$ response or outcome vector,

$$\boldsymbol{y} = \begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix}^T,$$

let $\boldsymbol{\theta}$ be a $p \times m$ parameter or input weight matrix,

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_{11} & \cdots & \theta_{1m} \\ \cdots & \cdots & \cdots \\ \theta_{p1} & \cdots & \theta_{pm} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_{(1)}^T \\ \vdots \\ \boldsymbol{\theta}_{(p)}^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_1 & \cdots \boldsymbol{\theta}_m \end{bmatrix},$$

let $\boldsymbol{t}$ be a $p \times 1$ parameter vector,

$$\boldsymbol{t} = \begin{bmatrix} t_1 & \cdots & t_m \end{bmatrix}^T,$$

let $\boldsymbol{\beta}$ be a $m \times 1$ parameter vector representing node weights,

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 & \cdots & \beta_m \end{bmatrix}^T,$$

and let b be a scalar parameter.

When one tries to bring neural network into the high-dimension set up, or equivalently, the small sample size scenario, it usually does not work well. The estimability issue [70] arise from the fact that even a single layer neural network may have too many parameters. This issue might already exist in the low dimensional case ($n < p$), let alone the high dimension case. A single layer neural network usually includes $mp + 2m + 1$ parameters, which is possible to be much more than the training sample size n. In practice, a neural network may work well with one of the local optimal solutions although this is not guaranteed by

theory. Regularization methods can be applied to help obtain a sparse solution. On one hand, proper choice of regularization shrinks partial parameters to zero, which addresses the statistical estimability issue. On the other hand, regularization makes the model more robust.

Assuming sparsity is usually the most efficient way of dealing with the high dimensionality. A lasso type regularization on the parameters has been shown numerically to have poor performance on neural network models. On one hand, lasso does not drop a feature entirely but only disconnect it with some hidden nodes. On the other hand, lasso does not select dependent predictor variables in a good manner [41]. Consider the sparse group lasso proposed by [114], which penalizes the predictors group-wise and individually simultaneously. It is a combination of the group lasso and the lasso, see for example [53]. The group lasso regularization part penalizes the input features' weights group-wisely: a feature is either selected or dropped, and it is connected to all nodes in the hidden layer if selected. The lasso part further shrinks some weights of the selected inputs features to zero: a feature does not need to be connected to all nodes in the hidden layer when selected.

Define the loss function as the log-likelihood function

$$
\begin{aligned}
& l(\boldsymbol{\theta}, \boldsymbol{t}, \boldsymbol{\beta}, b) \\
=& \sum_{i=1}^{n} \left[y_i (\sum_{k=1}^{m} \beta_k \psi(t_k + \boldsymbol{\theta}_k^T \boldsymbol{x}_i) + b) - \log \left(1 + \exp \left(\sum_{k=1}^{m} \beta_k \psi(t_k + \boldsymbol{\theta}_k^T \boldsymbol{x}_i) + b \right) \right) \right]. \quad (3.2)
\end{aligned}
$$

Besides the sparse group lasso regularization, we consider a l_2 regularization on other pa-

rameters. Then we have

$$\hat{\boldsymbol{\theta}}_{sgl}, \hat{\boldsymbol{t}}_{sgl}, \hat{\boldsymbol{\beta}}_{sgl}, \hat{b}_{sgl} = \underset{\boldsymbol{\theta},\boldsymbol{t},\boldsymbol{\beta},b\in\mathbb{R}^{pm\times m\times m\times 1}}{\arg\min} -\frac{1}{n} l(\boldsymbol{\theta}, \boldsymbol{t}, \boldsymbol{\beta}, b) + \alpha\lambda \sum_{j=1}^{p} \|\boldsymbol{\theta}_{(j)}\|_2 + (1-\alpha)\lambda\|\boldsymbol{\theta}\|_1, \tag{3.3}$$

such that

$$\|\boldsymbol{\beta}\|_2^2 + \|\boldsymbol{t}\|_2^2 + b^2 \leq K.$$

The sparse group lasso penalty [114, 53] includes a group lasso part and a lasso part, which are balanced using the hyper-parameter $\alpha \in (0, 1)$. The group lasso part treats each input as group of m variables, including the weights for the m hidden nodes connected to each input. This regularization will be able to include or drop an input variable's m hidden nodes group-wisely [148]. The lasso regularization is used to further make the weights sparse within each group, i.e., each input selected by the group lasso regularization does not have to connect to all hidden nodes. This combination of the two regularization makes the estimation even easier for small sample problems. The l_2 norm regularization on the other parameters is more about practical concerns, since it further reduces the risk of over-fitting.

Though with slight difference on the regularization, [53] proposed a fast coordinate gradient descent algorithm for the estimation, which cycles the gradient descent for the differentiable part of the loss function, the threshold function for the group lasso part and the threshold function for the lasso part. Three tuning parameters, α, λ and K can be selected with cross-validations on a grid search.

3.3 The Consistency of Neural Network Classification Risk

In this section, we conduct the theoretical investigation of classification accuracy of the neural network model. Before stating the theorems, we need a few assumptions. The independence property of neural networks, see [119], [4] and [53], states that the first-layer weights in $\boldsymbol{\theta}, \boldsymbol{t}$ satisfy

$$(\boldsymbol{\theta}_i, t_i) \neq \pm(\boldsymbol{\theta}_{i'}, t_{i'}), \forall i \neq i' = 1, \cdots, m$$

and

$$\boldsymbol{\theta}_i \neq \mathbf{0}, \forall i,$$

the set of dilated and translated functions $\mathbb{R}^p \to \mathbb{R}$

$$\{1, \sigma(\boldsymbol{\theta}_1^T \boldsymbol{x} + t_1), ..., \sigma(\boldsymbol{\theta}_m^T \boldsymbol{x} + t_m)\}$$

is linearly independent.

The independence property means that different nodes depend on the input predictor variables through different linear combinations and none of the hidden nodes is a linear combination of the other nodes, which is crucial in the universal approximation capability of neural networks. [113] proved that the above set is linearly independent if $\boldsymbol{\theta}_i' s$ are pairwise linearly independent, as long as the non-polynomial activation function is an integrable function which satisfies a polynomial growth condition.

According to [53], if the parameters $\boldsymbol{\phi} = (\boldsymbol{\theta}, \boldsymbol{t}, \boldsymbol{\beta}, b)$ satisfy the independence property,

the following equivalence class of parameters

$$EQ(\boldsymbol{\phi}) = \{\tilde{\boldsymbol{\phi}} \in \Theta : \eta_{\tilde{\boldsymbol{\phi}}}(\boldsymbol{x}) = \eta_{\boldsymbol{\phi}}(\boldsymbol{x}) \forall \boldsymbol{x}\}$$

contains only parameterizations that are sign-flips or rotations and has cardinality exactly $2^m m!$.

Let $\mathbb{P}$ be the distribution of Y for fixed $\boldsymbol{X}$ and $\mathbb{P}_n$ be the empirical measure. The best approximation in the neural network space is the equivalence class of parameters by minimizing the population loss

$$EQ_0 = \arg\min_{\boldsymbol{\phi} \in \Theta} \int_{\mathbb{R}^p \times \{0,1\}} l_{\boldsymbol{\phi}}(y, \boldsymbol{x}) dP_{\boldsymbol{X},Y},$$

where $l_{\boldsymbol{\phi}}(y, \boldsymbol{x}) dP_{\boldsymbol{X},Y}$ is the loss function with parameters $\boldsymbol{\phi}$. Let Q be the number of equivalent classes in EQ_0. The Excess loss is defined as

$$\varepsilon(\boldsymbol{\phi}) = \int_{\mathbb{R}^p \times \{0,1\}} \left(l_{\boldsymbol{\phi}}(Y, \boldsymbol{X}) - l_{\boldsymbol{\phi}^0}(Y, \boldsymbol{X}) \right) dP_{\boldsymbol{X},Y} \tag{3.4}$$

where $\boldsymbol{\phi}^0$ is a set of parameters in EQ_0. Moreover, when we refer to a set of parameters in EQ_0 for some parameter $\boldsymbol{\phi}$, we mean that $\boldsymbol{\phi}^0 \in EQ_0$ has the minimum distance to $\boldsymbol{\phi}$. [53] has shown that this excess loss plus the estimation of the irrelevant weights is bounded from above and may tend to zero with proper choices of n, p and the tuning parameters.

Another concern is the estimability of the parameters. A common remedy is to assume sparsity of the predictors. Thus we make the following assumption.

Assumption 3.1 (Sparsity)**.** Only s of the predictors are relevant in predicting y (without loss of generality, we assume the first s predictors, denoted S are relevant, and the rest,

denoted S^C, are irrelevant), all weights in $\boldsymbol{\theta}$ associated with S^C, denoted $\boldsymbol{\theta}_{S^C}$, are zero in the optimal neural network EQ_0.

The next assumption is a standard assumption in generalized models, which controls the variance of the response from below and above. Consider a general exponential family distribution on y with canonical function $b(\theta)$, common assumptions is to bound $b''(\theta)$ and $b'''(\theta)$ from above and below. However, in binary classification problems, these functions are automatically bounded from above by 1, thus we only need to assume the boundedness from below. Some literature assume constant bounds on these quantities, however, we do allow the bounds to change with n and the bounds may tend to zero as n goes to infinity.

Assumption 3.2 (Boundedness of variance)**.** The true conditional probability of y for a given $\boldsymbol{x}$ is bounded away from 0 and 1 by a quantity $\tilde{\epsilon}$, which might tend to zero.

The following two assumptions are inherited from [53]. The next assumption is a relatively weak assumption on the local convexity of the parameters.

Assumption 3.3 (Local convexity)**.** There is a constant $h_{min} > 0$ that may depend on m, s, f and the distribution $P_{\boldsymbol{X},Y}$, but does not depend on p such that for all $\boldsymbol{\phi} \in EQ_0$, we have

$$\left[\nabla^2_{\boldsymbol{\phi}} \left(\int_{\mathbb{R}^p \times \{0,1\}} l_{\boldsymbol{\phi}}(y, \boldsymbol{x}) dP_{\boldsymbol{X},Y} \right)\right]_{\boldsymbol{\phi}=\boldsymbol{\phi}^0} \succeq h_{min} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$$

where $A \succeq B$ means that $A - B$ is a positive semi-definite matrix.

The next assumption is made to bound the excess loss from below for the parameters outside EQ_0, i.e., the true model is identifiable. Let $d_0(\boldsymbol{\phi})$ be the minimum distance from an element in EQ_0 to $\boldsymbol{\phi}$, then we assume

Assumption 3.4 (Identifiability)**.** For all $\epsilon > 0$, there is an α_ϵ that may depend on m, s, f and the distribution $P_{\boldsymbol{X},Y}$, but does not depend on p such that

$$\alpha_\epsilon \leq \inf_{\boldsymbol{\phi}} \Bigg\{ \varepsilon(\boldsymbol{\phi}) : d_0(\boldsymbol{\phi}) \geq \epsilon \; and \; \|\boldsymbol{\theta}_{S^C}\|_1 \leq 3 \sum_{j \in S} \Omega_\alpha(\boldsymbol{\theta}_{(j)} - \boldsymbol{\theta}_{(j)}^{0,(\boldsymbol{\phi})}) + \|(\boldsymbol{t}, \boldsymbol{\beta}, b) - (\boldsymbol{t}^{0,(\boldsymbol{\phi})}, \boldsymbol{\beta}^{0,(\boldsymbol{\phi})}, b^{0,(\boldsymbol{\phi})})\|_2 \Bigg\}$$

Assumption 3.3 states that though neural network is a non-convex optimization problem globally, the parameters of the best neural network approximation of the true function $f(x)$ has a locally convex neighborhood. The assumption can be assured in this way. By the continuity of the representation of neural network and the loss function, the integration in assumption 3.3 is infinitely continuously differentiable with respect to the nonzero parameters, therefore the second derivative is a continuous function of the nonzero parameters. By the definition of the parameters of the best neural network approximation, $\boldsymbol{\phi}_0$ minimizes the integration in assumption 3.3. If there isn't a positive h_{min} that satisfies assumption, it either contradicts with the fact that the second derivative is continuous or the definition of $\boldsymbol{\phi}_0$.

Assumption 3.4 states that the non-optimal neural networks can be distinguished from the best neural network approximation in terms of the excess loss, if the parameters of the non-optimal neural network is not in the ϵ-neighborhood of any of the parameters of the best neural network class EQ_0. Similar to the compatibility condition in [18], the condition does not have to or even may not hold for the whole space, but is only needed in the subspace $\{\boldsymbol{\phi} : \|\boldsymbol{\theta}_{S^C}\|_1 \leq 3 \sum_{j \in S} \Omega_\alpha(\boldsymbol{\theta}_{(j)} - \boldsymbol{\theta}_{(j)}^{0,(\boldsymbol{\phi})}) + \|(\boldsymbol{t}, \boldsymbol{\beta}, b) - (\boldsymbol{t}^{0,(\boldsymbol{\phi})}, \boldsymbol{\beta}^{0,(\boldsymbol{\phi})}, b^{0,(\boldsymbol{\phi})})\|_2\}$, thus this is a weaker condition than imposing the lower bound on the excess loss. The subspace is

derived from the the basic inequality of the definition of $\hat{\boldsymbol{\phi}}$ with rearranging terms and norm inequalities, see for example [18]. Similar subspace can also been found in the compatible condition in [94]. Since s is unknown, it can not be checked in practice, but it is sufficient to check the inequality for all sets $S \in \{1, ..., p\}$ with cardinality s_0 if s_0 is known, which is a stronger version of assumption 3.4.

Now we are ready to state our main result. We have to admit that our theory based on the estimator from 3.3 is the global optima, which suffers from the biggest problem in optimization research: the gap between the global optima in theory and a local optima in practice. We will leave this computational issue to future research.

Theorem 3.1. Under assumptions 1-4, let our estimation be from 3.3, choosing tuning parameter $\lambda \geq 2T\tilde{\lambda}$ for some constant $T \geq 1$ and $\tilde{\lambda} = c\sqrt{m \log n/n}(\sqrt{\log Q} + \sqrt{m \log p} \log(nm)$ $/(1 - \alpha + \alpha/\sqrt{m}))$, if $\log(n)/(n\tilde{\epsilon}^2) \to 0$, $s^2 m\lambda^2/(n\tilde{\epsilon}^2) \to 0$ and $n^{-1}m^{9/2}s^{5/2}\sqrt{\log(p)} \to 0$ as $n \to \infty$, assume that our prediction is within a constant distance from the best approximation $\eta^0(\boldsymbol{x})$, then we have

$$R(\hat{C}) - R(C^*) \to 0 \; as \; n \to \infty$$

A proof of this theorem is given in appendix. This theorem states that with proper choice of tuning parameters and under some mild assumptions and controls of n, p and s, the high-dimensional neural network with sparse group lasso regularization tends to have the optimal classification risk. This is a significant improvement in the theoretical neural network study, since it gives the theoretical guarantee that high dimensional neural network will definitely work in such situations.

3.4 Simulation

In this section, we will show two examples. The first example is a revisit of the simulation study in [82], where we show numerical results that the SGLNN's performance is close to the DNP's performance in their set up. The second example considers a scenario where the sample size is much smaller, where we show numerical results that the SGLNN out-performs the DNP.

3.4.1 DNP Simulation: Revisit

In this subsection, we revisit the experiment in [82] and compare those models with the neural network with sparse group lasso regularization. We use exactly the same setup as [82]. The input variable $\boldsymbol{X}$ is drawn from $U(-1, 1)$, where the feature dimension p is fixed to be 10000. The corresponding labels are obtained by passing $\boldsymbol{X}$ into the feed forward neural network with hidden layer sizes $\{50, 30, 15, 10\}$ and ReLU activation functions. Input weights connecting the first m inputs are randomly sampled from $N(0, 0.5)$. The remaining input weights are kept zero. Furthermore, 5% of the labels are randomly chosen and flipped to add noises. For each $m = 2, 5, 10, 25$, we generate 800 training samples, 200 validation samples and 7500 test samples. We report the AUC and F1 scores of the models in table 3.1 on 5 repetitions of the data generation, which is exactly the same as in [82]. The DNP model is coded according to their algorithm outline in python with pyTorch. The HSICLasso is implemented with the package by the authors followed by the SVM package in scikit-learn. The LogR-l_1 model is implemented with the LogisticRegressionCV package in scikit-learn.

In this simulation study, the assumptions in our theory are all satisfied. First, we have a sparse level of 2, 5, 10 and 25, which are very small portions of the total number of features,

10000, thus assumption 2.1 is satisfied. Second, we've controlled the seed such that the labels are balanced between 0 and 1, and the true probabilities generated from the neural network are bounded away from 0 and 1 by a non-negligible constant, thus assumption 2.2 is satisfied. Then, we generate x from uniform distribution and y from a neural network structure, which is a continuous distribution plus a continuous map from the original space to the probabilities $[0, 1]$. Therefore, the local convexity assumption is justified by continuity. Finally, assumption 2.4 is a property of neural networks that has been argued in section 3.3. Therefore, this example not only serves as a revisit of the DNP paper, but also serves as a support for our theory.

Results of this example is shown in table 3.1. From the results, we see both the SGLNN model and the DNP model outperform the other two baseline models. The SGLNN's performance is very close to the performance of DNP with a small gap, which is in accordance to our expectations. Deeper neural networks have much higher representation powers of complicated functions. The stability of the two models are close, with the SGLNN having slightly smaller SE in the $m = 2$ and $m = 5$ cases. The DNP model does not use dropout in the prediction process, while the SGLNN uses l_1 norm penalty along with the group lasso penalty. The SGLNN is expected to show stabler results. The reason that we see no significant difference in SE is that the sample size, 800, is large enough to train the DNP with a full network on the selected variables without over-fitting. To further investigate the performance, we study the smaller sample scenario in the next subsection.

3.4.2 Smaller Sample Size Case

In this subsection, we consider a smaller sample size, which happens in many applications such as clinic trials, genetic expression data analysis and MRI data analysis. With the same

	True Dim	2	5	10	25
LogR-l_1	AUC(std)	0.897(0.015)	0.745(0.035)	0.661(0.029)	0.629(0.015)
HSIC-Lasso	AUC(std)	0.920(0.001)	0.844(0.015)	0.732(0.025)	0.638(0.021)
DNP	AUC(std)	0.925(0.020)	0.879(0.035)	0.784(0.020)	0.669(0.016)
SGLNN	AUC(std)	0.911(0.015)	0.862(0.021)	0.770(0.021)	0.658(0.016)
LogR-l_1	F1 score(std)	0.889(0.022)	0.748(0.032)	0.668(0.037)	0.638(0.027)
HSIC-Lasso	F1 score(std)	0.914(0.000)	0.791(0.010)	0.680(0.012)	0.368(0.028)
DNP	F1 score(std)	0.959(0.009)	0.849(0.033)	0.769(0.017)	0.811(0.015)
SGLNN	F1 score(std)	0.940(0.005)	0.839(0.012)	0.747(0.019)	0.754(0.015)

Table 3.1: The AUC and F1 score of the compared models in the simulation study. Standard errors are given in the parentheses.

Training size	Metric	100	200	300	500
DNP	AUC(std)	0.606(0.118)	0.701(0.091)	0.727(0.074)	0.828(0.043))
SGLNN	AUC(std)	0.624(0.099)	0.703(0.089)	0.762(0.053)	0.820(0.026)
DNP	F1 score(std)	0.567(0.095)	0.634(0.073))	0.679(0.043)	0.750(0.035)
SGLNN	F1 score(std)	0.602(0.073)	0.641(0.069)	0.690(0.029)	0.746(0.029)

Table 3.2: The AUC and F1 score of the compared models in a smaller sample size scenario with $m = 5$. Standard errors are given in the parentheses.

set up as the model generation in the last subsection, we generate a training sample of size $n = 100, 200, 300$ and 500. The number of active features is fixed to be $m = 5$. The total sample size is set to 10000, thus the corresponding testing sample sizes are $9900, 9800, 9700$ and 9500. We compare the performance between the SGLNN and DNP in this set up. The results are shown in table 3.2.

From the results, we see the SGLNN model outperforms the DNP in the smaller sample size scenario when $n = 100, 200, 300$. The reason is that DNP overfits the training data due to small sample size, while SGLNN has lower risk of over-fitting compared with DNP. In all the four sample sizes, the SGLNN has smaller SE than the DNP model's SE. SGLNN achieved this by a simpler representation and the extra l_1 norm regularization. The results suggest that we need a simpler model to prevent over-fitting when the training sample size

is very small, which is the case in most biology data. Moreover, an extra l_1 regularization makes the model more stable and reliable.

3.5 Real Data examples

In this section, we gave real data applications in different research areas: gene expression data, MRI data and computer vision data. These examples indicate that the sparse neural network has good performance and is useful in correlated predictor situations. Through all three examples, the regularized neural network was implemented with fast speed using the algorithm by [53] through their library in python3 on a desktop computer with Ubuntu 18 system on a i7 processor and GTX1660TI graphic cards.

To evaluate the model performance, all accuracy results are measured on the testing sets which are not used for training and averaged on different train test splits. The number of features in the results are medians among all numbers of features that correspond to the best models evaluated from cross-validation.

3.5.1 example 1: Prostate Cancer Data

In this example, we considered a prostate cancer gene expression data, which is publicly available in `http://featureselection.asu.edu/datasets.php`. The data set contains a binary response with 102 observations on 5966 predictor variables. Clearly, the data set is really a high dimensional data set. The responses have values 1 (50 sample points) and 2 (52 sample points), where 1 indicates normal and 2 indicates tumor. All predictors are continuous predictors, with positive values.

40 observations from no expression group and 40 observations from expression group were

randomly selected to form the training group. The rest 22 observations form the testing group. We run the sparse ANN model on a replication of 100 different train-test splits. On average, the sparse neural network selects only 18 predictors and uses 4 hidden nodes. Using a cross-validation technique, the hyper-parameters thus the number of features were decided. It has a average training error rate of 0.04 and a testing error rate of 0.045. The results compared with other methods are listed in table 3.3. The sparse ANN and l_1 penalized linear SVM perform the best with 95.5% and 95% accuracy, respectively. The gradient boosting tree classifier [25] is a powerful ensemble classification method but performs worse than regularized methods in the high-dimensional setting with an accuracy of 92.2% using 83.5 features (on an average). The logistic regression with l_1 regularization uses 36 features and achieves an accuracy of 93.3%. The GAM performs the worst with an accuracy of 91.8%, mainly due to the infeasibility of basis expansion in this data set, where the data distribution is highly skewed.

In summary, we showed numerically that the sparse group lasso penalized neural network is able to achieve at least as good as the existing methods along with providing strong theoretical support. The greater standard error mainly comes from additional tuning parameters. In terms of the number of predictor variables, it is not the best. However, as we found in the investigation, since ANN is a non-convex optimization problem, as people continue to train the model and tune the hyper-parameters, they get better accuracy rates with less number of predictor variables.

3.5.2 example 2: MRI Data for Alzheimer's Disease

Data used in this example is from the Alzheimer's disease Neuroimaging Initiative (ADNI) database (`http://www.loni.ucla.edu/ADNI`). We used T1-weighted MRI images from

Classifier	Test accuracy	Number of features
Regularized neural network	0.955(0.066)	18
Gradient boosting tree	0.922(0.058)	83.5
Logistic Regression with Lasso	0.933(0.058)	36
l1 penalized Linear SVM	0.950(0.052)	16
Generalized additive model with group lasso	0.918(0.061)	5

Table 3.3: Test accuracy with standard error in parentheses and median of number of features for different classifiers in the Prostate gene data example.

the collection of standardized datasets. The description of the standardized MRI imaging from ADNI can be found in `http://adni.loni.usc.edu/methods/mri-analysis/adni-\standardized-data/`. The images were obtained using magnetization prepared rapid gradient echo (MARAGE) or equivalent protocols with varying resolutions (typically 1.0×1.0 mm in plane spatial resolution and 1.2 mm thick sagittal slices with $256 \times 256 \times 256$ voxels). The images were then pre-processed according to a number of of steps detailed at `http://adni.loni.usc.edu/methods/mri-analysis/mri-pre-processing/`, which corrected gradient non-linearity, intensity inhomogeneity and phantom-based distortion. In addition, the pre-processed imaging were processed by FreeSurfer for cortical reconstruction and volumetric segmentation by Center for Imaging of Neurodegnerative Diseases, UCSF. The skull-stripped volume (brain mask) obtained by FreeSurfer cross-sectional processing were used in this study.

In our example, we used images from ADNI-1 subjects obtained using 1.5 T scanners at screening visits, and we used the first time point if there are multiple images of the same subject acquired at different times. There are totally 414 subjects, among which 187 are diagnosed as Alzheimer's disease and 227 healthy subjects at the screening visit. An R package ANTsR were applied for imaging registration. Then 3dresample command in AFNI were used to adjust the resolution and reduce the total number of voxels of the imaging to

$18 \times 22 \times 18$. The x axis and y axis for horizontal plane, x axis and z axis for coronal plane and y axis and z axis for sagittal plane. Only the 1100 voxels located in the center of the brain were used as features for classification. After removing the voxels with zero signal for most of the subjects, we have 1971 voxels left in use.

We randomly sampled 100 from the 187 AD subjects and 100 from the 227 healthy subjects as our training set, and the rest 214 subjects as testing set. The sparse ANN model was run on 100 replications of different train-test split. On average, the neural network finally selects 7 predictors and used 12 hidden nodes. It has a training error rate of 0.21 and a testing error rate of 0.224. The results compared with other methods are listed in table 3.4. The sparse ANN has a competitive performance which is slightly better than the PMLE-LDA with similar standard error. The goal of this experiment is not to show that the sparse group lasso neural network outperforms the other methods significantly, but just demonstrates that the model can be tuned to work as good as the other methods along with statistical theory. With finer tuning of the hyper-parameters, the results can be further improved. We compared the results with a few methods including the MLE-LDA, the PMLE-LDA (penalized MLE-LDA; [79]), the PREG-LDA (penalized regular LDA; [79]), the FAIR (Feature Annealed Independence Rule; [44]) and the NB (Naive Bayes; [14]). Methods without regularization are not presented, since the high-dimensionality prevents it from giving stable solutions. The penalized MLE-LDA produces an accuracy rate of 77.2% using only 5 predictor variables. The PREG-LDA method has the least number of predictors, 3, but has a lower accuracy rate 0.750. The rest of the methods perform worse. Besides the application, this example indicates that the ANN could be an alternative tool for spatial data modeling, at least for classification.

Classifier	Test accuracy	Number of features
Regularized neural network	0.776(0.031)	7
MLE-LDA	0.692(0.030)	1971
PREG	0.750(0.029)	3
PMLE-LDA	0.772(0.025)	5
FAIR	0.632(0.033)	7
NB	0.674(0.024)	1971

Table 3.4: Test accuracy with standard error in parentheses and median of number of features for different classifiers in the MRI Alzheimer's disease example.

Figure 3.1: example image from the KITTI 2D object detection data set.

3.5.3 example: KITTI Autonomous Driving Data

For the generalization, we also give an example on the computer vision tasks. The data used in this example is from [57], which has different aspect of images or stereos from high-resolution color and gray-scale video cameras or ladar (laser radar). The image and stereo data includes daily traffic data that help developing autonomous driving in different aspects. In our example, we used the 2D object detection data, for example, see figure 3.1.

The 2D object data set includes 7481 training images and 7518 testing images, comprising a total of 80256 labeled objects. All images are colored and saved as png. Since this was an open competition, the labels for testing data are not available. We only used the training data set as our example data. The original data includes 13 different label classes, however, to emphasize the binary classification ability of the regularized ANN model, we took only 2

Figure 3.2: example images of pedestrians and cars after pre-processing

classes: pedestrian and car. The sub-images of pedestrians and cars were extracted from the original images using the location information provided by the data set. This gives us 18000 car images and 7000 pedestrian images of resolution ranging from 40-by-40 to 180-by-150. Since the regularized ANN methods is especially useful when the sample size is small, we randomly sample 2000 cars and 2000 pedestrians as our data. We shuffled the 4000 images, and divided them into 800 training images and 3200 testing images. figure 3.2 show the images after pre-processing. A pre-processing steps are done with python libraries including matplotlib, PIL and pandas.

Since ANN does not handle local feature information as good as convolutional neural networks (CNN), we did a feature extraction using the pre-trained VGG19 CNN model [115], whose weights were trained on 120 million images of over 1000 classes. We adopted the convolutional layers from the model as a feature extractor. To feed the images into the model, a re-sizing (from the original size to 224-by-224-by-3) were performed using bi-linear interpolation methods. The VGG19 feature extractor generates a 4096-by-1 vector from each image. Our ANN model takes this 4096-dimensional input and trains on the 800 images. We compare our results with the regular ANN, the logistic regression with l_1 regularization, the

Classifier	Test accuracy	Number of features
SGLNN	0.994(0.001)	148
Neural network	0.514(0.102)	4096
Log-l_1	0.991(0.002)	176
SVM-l_1	0.993(0.001)	144
GAM	0.989(0.002)	152

Table 3.5: Test accuracy with standard error in parentheses and median of number of features for different classifiers in the KITTI autonomous driving example.

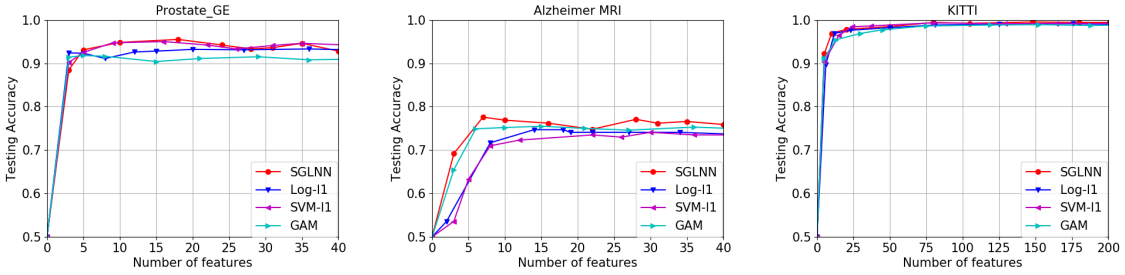

Figure 3.3: Test accuracy score vs sparsity level in the three examples.

SVM with l_1 regularization and the GAM with group lasso regularization. The results are shown in table 3.5. The regular neural network totally failed due to the high-dimensionality and hence estimability issue. The logistic regression and the GAM have more nonzero features and slightly greater standard error. The SVM has similar result to the SGLNN, but the SGLNN gains a slightly better performance.

Moreover, we have plotted the accuracy score for all three examples along with the sparsity level. l_1 logistic regression, l_1 SVM and group lasso penalized generalized additive model (GAM) are used along with the sparse group lasso neural network (SGLNN). These plots give us a hint how sparsity level influences the prediction results. The plots are given in figure 3.3. All these examples illustrate the usefulness and the numerical properties of our proposed high dimensional neural network model.

3.6 Discussion

In this paper, we considered the sparse group lasso regularization on high-dimensional neural networks and proved that under mild assumptions, the classification risk converges to the optimal Bayes classifier's risk. To the best of our knowledge, this is the first result that the classification risk of high-dimensional sparse neural network converges to the optimal Bayes risk. Neural networks are very good approximations to correlated feature functions, including computer vision tasks, MRI data analysis and spatial data analysis. We expect further investigation warranted in coming days. The computational issue of finding the global optimal in non-convex optimization problems is still waiting to be solved to eliminate the gap between the theory and in practice. This will pave way for further theoretical research.

3.6.1 The $l_1 + l_0$ Penalty and Algorithm

An innovative idea that deserves further investigation is to specify a larger number of hidden nodes and use a $l_0 + l_1$ norm penalty on the hidden nodes parameters $\boldsymbol{\beta}$. This methods searches a grander of solution field and will give a model at least as good as the l_2 norm penalty. Moreover, $l_0 + l_1$ norm penalty is proved to work well in low signal cases [92]. In detail, the formulation is to minimize 3.3 plus an extra regularization on $\boldsymbol{\beta}$: $\lambda_3\|\boldsymbol{\beta}\|_0+\lambda_4\|\boldsymbol{\beta}\|_1$. This formulation does not bring in extra tuning parameters, since we release the penalization on l_2 norm of $\boldsymbol{\beta}$ and the number of hidden nodes m. With $l_0 + l_1$ norm penalty, the parameters can be trained using coordinate descent algorithm with the $l_0 + l_1$ norm penalty being handled by mixed integer second order cone optimization (MISOCO) algorithms via optimization software like Gurobi. This algorithm adds an extra step to the algorithm in [53] to handle the $l_0 + l_1$ norm penalty.

Specifically, we are optimizing

$$\hat{\boldsymbol{\theta}}_{sgl}, \hat{\boldsymbol{t}}_{sgl}, \hat{\boldsymbol{\beta}}_{sgl}, \hat{b}_{sgl} = \underset{\boldsymbol{\theta}\in\mathbb{R}^{p\times m}, \boldsymbol{t}\in\mathbb{R}^m, \boldsymbol{\beta}\in\mathbb{R}^m, b\in\mathbb{R}}{\arg\min} -\frac{1}{n} l(\boldsymbol{\theta}, \boldsymbol{t}, \boldsymbol{\beta}, b) + \alpha\lambda_1 \sum_{j=1}^{p} \|\boldsymbol{\theta}_{(j)}\|_2 + (1-\alpha)\lambda_1 \|\boldsymbol{\theta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_1$$

such that

$$\|\boldsymbol{\beta}\|_0 \leq k$$

Let's start by computing the partial derivative of l with respective to $\boldsymbol{\theta}_{(j)}$. Let $\sigma(\cdot)$ be the sigmoid function

$$\sigma(x) = \frac{\exp(x)}{1+\exp(x)}$$

which is our choice of the activation function $\psi(\cdot)$. Noting that the sigmoid function has derivative $\sigma'(x) = \sigma(x)[1-\sigma(x)]$. Then we have

$$\frac{\partial l}{\partial \theta_{jk}} = \sum_{i=1}^{n} \left\{ [y_i - \sigma(\eta_i)] \beta_k \sigma'(\xi_{ik}) x_{ij} \right\}$$

Note that

$$\frac{\partial \theta_{jk}}{\partial \boldsymbol{\theta}_k} = (0, ..., 0, 1, 0, ..., 0)^T$$

Then we get,

$$\frac{\partial l}{\partial \boldsymbol{\theta}_k} = \sum_{i=1}^{n} \left\{ [y_i - \sigma(\eta_i)] \sigma'(\boldsymbol{\xi}_i) x_i \right\} \beta_k = \beta_k \boldsymbol{x}^T \mathrm{diag}[\boldsymbol{y} - \boldsymbol{\sigma}(\boldsymbol{\eta})] \sigma'(\boldsymbol{\xi}_k) \tag{3.5}$$

The partial derivative of l w.r.t $\boldsymbol{t}$ is given by

$$\frac{\partial l}{\partial \boldsymbol{t}} = \text{diag}[\boldsymbol{\beta}] \left[\boldsymbol{\sigma}'(\boldsymbol{\xi})(\boldsymbol{y} - \boldsymbol{\sigma}(\boldsymbol{\eta}))\right]$$

where $\boldsymbol{\sigma}'(\boldsymbol{\xi})_{ki} = \sigma'(\xi_{ki})$. The partial derivative of l w.r.t $\boldsymbol{\beta}$ is given by

$$\frac{\partial l}{\partial \boldsymbol{\beta}} = \boldsymbol{\sigma}(\boldsymbol{\xi})(\boldsymbol{y} - \boldsymbol{\sigma}(\boldsymbol{\eta}))$$

Finally, the partial derivative of l w.r.t b is given by

$$\frac{\partial l}{\partial b} = \boldsymbol{y} - \boldsymbol{\sigma}(\boldsymbol{\eta})$$

Since the sparse group lasso penalty only applies to $\boldsymbol{\theta}$, thus similar to [53], a generalized gradient descent can be applied to find a critical point. Now we fix $\boldsymbol{\beta}$ and iterate $\boldsymbol{\theta}$, $\boldsymbol{t}$ and b. This update includes three steps. The first step is to update the parameters $\boldsymbol{\theta}$, $\boldsymbol{t}$ and b in the likelihood part using gradient descent. Then let $S(\cdot, \cdot) : \mathbb{R}^{m \times n} \times \mathbb{R} \mapsto \mathbb{R}^{m \times n}$ be the coordinate-wise soft-thresholding operator

$$(S(x, c))_{ij} = \text{sign}(x_{ij})(|x_{ij}| - c)_+$$

The second step performs a soft-thresholding operation for each θ_{ij} due to the lasso penalty. Finally the third step performs a soft-scaling operation on each group $\boldsymbol{\theta}_{(j)}$ due to the group lasso penalty. The soft-scaling operation, see [114], is given by

$$\boldsymbol{\theta}_{(j)}^{new} = \left(1 - \frac{\gamma\lambda\alpha}{\|\boldsymbol{\theta}_{(j)}^{old}\|_2}\right)_+ \boldsymbol{\theta}_{(j)}^{old}$$

where $\theta_{(j)}^{old}$ is after the soft-thresholding operation and γ is the step size. The step size can be found using line search. In specific, let $\boldsymbol{\varepsilon}^{(k)}$ be the 2-norm of the change of $\boldsymbol{\theta}$, $\boldsymbol{t}$ and b in the $(k+1)^{th}$ iteration. The line search criterion is satisfied if

$$-\frac{1}{n}l(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{t}^{(k+1)}, b^{(k+1)}) \leq -\frac{1}{n}l(\boldsymbol{\theta}^{(k)}, \boldsymbol{t}^{(k)}, b^{(k)}) - t\gamma^{(k+1)}(\boldsymbol{\varepsilon}^{(k)})^2$$

for some fixed $t \in (0, 1)$.

After updating $\boldsymbol{\theta}$, $\boldsymbol{t}$ and b, we fix them and update $\boldsymbol{\beta}$. For some L, we have

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \underset{\boldsymbol{\beta}\in\mathbb{R}^m}{\arg\min} -\frac{1}{n}\sum_{i=1}^{n}\Big[y_i(\boldsymbol{\sigma}(\boldsymbol{\xi}_i)^T\boldsymbol{\beta} + b) - \log(1 + \exp(\boldsymbol{\sigma}(\boldsymbol{\xi}_i)^T\boldsymbol{\beta} + b))\Big] + \|\boldsymbol{\beta}\|_1 \ s.t.\ \|\boldsymbol{\beta}\|_0 \leq k \\ &:= f(\boldsymbol{\beta}) + \|\boldsymbol{\beta}\|_1 \ s.t.\ \|\boldsymbol{\beta}\|_0 \leq k\end{aligned}$$

Using Taylor expansion of $\boldsymbol{\beta}$ around some $\boldsymbol{\beta}_0$, we have

$$\begin{aligned}f(\boldsymbol{\beta}) &= f(\boldsymbol{\beta}_0) + \nabla f(\boldsymbol{\beta}_0)^T(\boldsymbol{\beta} - \boldsymbol{\beta}_0) + \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^T\Delta f(\boldsymbol{\beta}^*)(\boldsymbol{\beta} - \boldsymbol{\beta}_0) \\ &\leq f(\boldsymbol{\beta}_0) + \nabla f(\boldsymbol{\beta}_0)^T(\boldsymbol{\beta} - \boldsymbol{\beta}_0) + \frac{L}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2^2 \\ &:= M(\boldsymbol{\beta})\end{aligned}$$

for some $\boldsymbol{\beta}^*$ and L. By the properties of sigmoid function, $L \geq 1$ is preferred. Observe that

$$\text{minimize } M(\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1 \ s.t.\ \|\boldsymbol{\beta}\|_0 \leq k \ \Leftrightarrow \ \text{minimize } \tilde{M}(\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1 \ s.t.\ \|\boldsymbol{\beta}\|_0 \leq k$$

where

$$\tilde{M}(\boldsymbol{\beta}) = \frac{L}{2}\left\|\boldsymbol{\beta} - \left(\boldsymbol{\beta}_0 - \frac{1}{L}\nabla f(\boldsymbol{\beta}_0)\right)\right\|_2^2$$

$$
\begin{aligned}
&\text{minimize } u + \lambda_2 v \\
&\quad s.t.\ \frac{L}{2}\left\|\boldsymbol{\beta} - \left(\boldsymbol{\beta}^{(m)} - \frac{1}{L}\nabla f(\boldsymbol{\beta}^{(m)})\right)\right\|_2^2 \leq u \\
&\quad\quad -Mz_j \leq \beta_j \leq Mz_j, j = 1, ..., p \\
&\quad\quad z_j \in \{0, 1\}, j = 1, ..., p \\
&\quad\quad \sum_j z_j = k \\
&\quad\quad -\bar{\beta}_j \leq \beta_j \leq \bar{\beta}_j, j = 1, ..., p \\
&\quad\quad \bar{\beta}_j \geq 0, j = 1, ..., p, \sum_j \bar{\beta}_j \leq v
\end{aligned}
$$

Table 3.6: The MISOCO formulation for the $l_1 + l_0$ penalty in dealing with the l_0 norm step.

This suggest a simple cycling for $\boldsymbol{\beta}$, i.e.

$$
\boldsymbol{\beta}^{(m+1)} = \underset{\boldsymbol{\beta}\in\mathbb{R}^m}{\arg\min} \frac{L}{2}\left\|\boldsymbol{\beta} - \left(\boldsymbol{\beta}^{(m)} - \frac{1}{L}\nabla f(\boldsymbol{\beta}^{(m)})\right)\right\|_2^2 + \lambda_2\|\boldsymbol{\beta}\|_1 \ s.t.\ \|\boldsymbol{\beta}\|_0 \leq k
$$

This convert optimizing $\boldsymbol{\beta}$ to a mixed integer second order cone optimization (MISOCO) problem, see [92]. The formulation is given in table 3.6. The optimization problem is a second order cone with one quadratic constraint and some linear constraints. It can be solved using optimization software like Gurobi. Therefore, the results suggest a cycling iteration on the parameters algorithm to find the estimated parameters of the high-dimensional neural network model. The detail algorithm is given in algorithm 1.

Algorithm 1: Algorithm for training classification neural network with $l_1 + l_0$ penalty

Initial neural network parameters $\boldsymbol{\theta}^{(0)}$, $\boldsymbol{t}^{(0)}$, $\boldsymbol{\beta}^{(0)}$ and $b^{(0)}$. Choose initial step size $\gamma^{(0)}$ and step size scaling parameter $s \in (0, 1)$. Select tolerance τ;

while *Error of all parameters is greater than* τ **do**

  while *Error of* $\boldsymbol{\theta}$, $\boldsymbol{t}$ *and* b *is greater than* τ **do**

$$\boldsymbol{\theta}^{(k+1)}, \boldsymbol{t}^{(k+1)}, b^{(k+1)} = \boldsymbol{\theta}^{(k)}, \boldsymbol{t}^{(k)}, b^{(k)} - \gamma^{(m)} \nabla \left(-\frac{1}{n} l(\boldsymbol{\theta}^{(k)}, \boldsymbol{t}^{(k)}, b^{(k)}) \right)$$

$$\boldsymbol{\theta}^{(k+1)} = S\left(\boldsymbol{\theta}^{(k+1)}, \gamma^{(m)} \lambda_1 (1 - \alpha)\right)$$

    for *j=1,...,p* **do**

$$\boldsymbol{\theta}_{(j)}^{(k+1)} = \left(1 - \frac{\gamma^{(m)} \lambda \alpha}{\|\boldsymbol{\theta}_{(j)}^{(k+1)}\|_2}\right)_+ \boldsymbol{\theta}_{(j)}^{(k+1)};$$

    end

    $\gamma^{(m+1)} = s\gamma^{(m)}$;

    Until some line search criterion satisfies;

  end

  while *Error of* $\boldsymbol{\beta}$ *is greater than* τ **do**

    Use the MISOCO formulation above to optimize $\boldsymbol{\beta}$ as below

$$\boldsymbol{\beta}^{(k+1)} = \underset{\boldsymbol{\beta} \in \mathbb{R}^m}{\arg\min} \frac{L}{2} \left\| \boldsymbol{\beta} - \left(\boldsymbol{\beta}^{(k)} - \frac{1}{L} \nabla f(\boldsymbol{\beta}^{(k)}) \right) \right\|_2^2 + \lambda_2 \|\boldsymbol{\beta}\|_1 \; s.t. \; \|\boldsymbol{\beta}\|_0 \leq k$$

  end

end

Chapter 4

Ensemble Neural Network Selection (ENNS)

We have mentioned that directly optimizing the penalized neural network loss function is risky when the sample size is small, since it's easy to get trapped in a local minimal. As there is a connection between the $l_{p,1}$ norm penalty and the stage-wise selection algorithm, it's natural to consider adding variables one by one. However, the deep neural pursuit (DNP) algorithm in [82] does not implement methods to stabilize the selection process, and the estimation is based on a full neural network, which still bring in the estimability issue. In this chapter, we consider an ensemble stage-wise variable selection algorithm with deep neural networks, which is named as ensemble neural network selection (ENNS). Moreover, we propose a heuristic methods to further sparsify the neural network by imposing soft-thresholding functions to the parameters. We will provide methodology for the two-step approach, and we will show that the two-step approach enjoys selection consistency and risk consistency under certain conditions. Simulations and real data examples are used to support the arguments.

4.1 Introduction

High-dimensional statistics modeling [18] has been popular for decades. Consider a high-dimensional regression or binary classification problem. Let $\boldsymbol{x} \in \mathbb{R}^p$ be the feature vector, and let $y \in \mathbb{R}$ for regression problem and $y \in \{0, 1\}$ for classification problem be the response. Our goal is to build a model based on the training sample $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$. We have more features than the sample size, i.e., $p > n$. Moreover, many data have complicated relationship among different variables, which is hard to capture through a linear model. Neural network is the one of the best models at capturing complicated relationships. It's interesting to consider a neural network structure between $\boldsymbol{x}$ and y.

In general, high-dimensional model does not have consistent estimations since the systems have less constraints than number of variables. Two major approaches can be used to deal with the high-dimensionality. The first major approach is to assume that the feature space is sparse, i.e., only a small fraction of the variables are included in the modeling with y. A model with only a fraction of the original features enjoys simplicity and interpretability. Sparse solutions can be obtained using regularization [126, 65] or stage-wise algorithms [41]. Regularization obtains sparse solution by shrinking the unimportant features' coefficients to zero. The estimated coefficients are shrinkage estimators and thus have smaller variance [32]. However, regularization with multiple tuning parameters takes longer to run and may be sensitive in the tuning parameters in practice. Stage-wise algorithms adds variables one by one and stops at a preferred time.

The second major approach is projection-based. One finds a lower dimensional representation of the original feature space. Linear projection methods include the PCA [64] in the low dimensional case and some of its variants [68, 157] in the high-dimensional case. Kernel

PCA [109] performs PCA on a reproducing kernel Hilbert space to achieve non-linearity. Manifold learning [76] embeds the original feature space to a low-dimensional manifold. Except for the manifold learning algorithms that reduce the original dimensionality to 2 or 3 dimension for visualization, a few manifolding learning including the multidimensional sacling (MDS) by [130], the local linear embedding (LLE) by [107], and the Isomap by [124]. Applications of the manifold learning algorithms in the high-dimensional set up is studied for specific fields, but a general framework is not available. Another popular dimension reduction technique is the auto-encoder [75], which uses a neural network to encode the feature space and decode the representation to be as close to the original feature space as possible. All of the above methods are unsupervised, and the lower-dimensional representation is no longer any of the original features. Therefore, we lose interpretability by doing so. Current manifold learning algorithms focus more on data visualization, which reduces the dimensionality to two or three, see for example [139]. These applications are not useful in building models.

On the other hand, neural networks have been utilized to model complicated relationship since the 1940s [72], and gained much more attention since the computer hardware's great improvement in this century. Specifically, [101] showed that the computation of neural networks can be greatly improved by GPU acceleration than purely running on CPU, thus makes it easy to train deeper network structures. Nowadays, variant neural networks are being applied world-wide, including the convolutional neural network (CNN), recurrent neural network (RNN), residual network (ResNet), and etc. In theory, neural network works in representing complicated relationships mainly lies on the universal approximation theorem [33, 8, 4, 113]. The theorem states that a shallow neural network (neural network with one hidden layer) is able to approximate any continuous function with an arbitrarily small

error given a huge number of hidden nodes, under mild assumptions. In practice, to reach this good approximation, usually a huge set of training data is needed, since the number of parameters in a neural network is much more than that in other models. Moreover, the non-convexity of a neural network structure makes it impossible to obtain a global optimum. Luckily, a local optimum of the neural network provides a good approximation.

Deep Neural Network (DNN) is a neural network with a deep structure of hidden layers, which has better performance than the shallow neural network (neural network with only 1 hidden layer) in many aspects including a broad field of subjects including pattern recognition, speech recognition and etc., see for example [97, 80, 66]. The deep structure has a greater approximation power than a shallow neural network. There have been a few literature regarding the approximation power of deep neural networks [12, 104, 112, 43]. The results suggest that using a deep neural network helps reduce the approximation error, which is useful in the cases where the approximation error dominates the total error. Therefore, it's necessary to consider a deep neural network model over a shallow model. However, finding a way to make the deep neural network well trained on a small sample size is necessary.

In this paper, we will discuss the stage-wise variable selection algorithm with neural networks. We will show that the existing stage-wise algorithm performs well at the beginning and selects the correct variables. However, at the later steps, the probability that it will select a correct variable decreases. Thus we will propose an ensemble algorithm on the stage-wise variable selection algorithm, named ensemble neural network selection (ENNS) algorithm. We will show that the new algorithm select all correct variables with high probability and its false positive rate converges to zero. Moreover, instead of a regular neural network trained on the select variables, we proposed a few methods to further reduce the variance of the final model. We will give numerical comparison of these proposed algorithms, and propose

an algorithm for the l_1 penalized neural network with soft-thresholding operators.

In section 4.2, we will present the ideas and algorithms behind the ENNS algorithm and the methods of increasing stability during the estimation step. In section 4.3, we will provide the theory for the arguments in this paper. In section 4.4, we will present the results of some numeric study to support our theorems and arguments. In section 4.5, we will evaluate the new method on real world data. Section 4.6 will discuss some conclusion and future works.

4.2 The Two-step Variable Selection and Estimation Approach

Consider a feature vector $\boldsymbol{x} \in \mathbb{R}^p$ and a response $y \in \mathbb{R}$ for the regression set up and $y \in \{0, 1\}$ in the classification set up. We have data $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$ consisting of independent observations. Denote the design matrix $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)^T \in \mathbb{R}^{n \times p}$ and the response vector $\boldsymbol{y} = (y_1, ..., y_n)^T$. As we mentioned before, we have more variables than observations, i.e., $p > n$. According to the previous discussion, variable selection is an important step in high-dimensional modeling. If one includes all variables in the model, there will be at least p parameters to estimate, which can not be done stably with the n observations. If a more complicated model is needed, the number of parameters will be tremendous, which will cause severe over-fitting and high variance with a small training sample size.

Therefore, we hope a feature selection step at first can help pick the import variables, and another estimation step could build a more accurate model based on the selected variables. Moreover, we will use deep neural networks as the structure, since it will be able to capture the complicated relationships. We will consider a stage-wise algorithm in the variable selection step, which performs a similar function as the DNP model in [82]. However, we will

show that the stage-wise algorithm in DNP suffers from some disadvantage and propose an ensemble algorithm to relieve this situation. In the second step, we will discuss the methods to reduce variance and prevent over-fitting, since a deep neural network with only a few input variables can still have a huge number of parameters.

4.2.1 The Ensemble Neural Network Selection (ENNS) Algorithm

Consider the feature selection approach in [82]. Let $\mathcal{D} : \mathbb{R}^p \to \mathbb{R}$ be a deep neural network function that maps the original feature space to the output space. We don't specifically mark the number of hidden layers and hidden node sizes in the notation, but simply assume the deep neural network has m hidden layers with sizes $h_1, ..., h_m$. Denote the weight matrices in each layer to be $\boldsymbol{W}_0, ..., \boldsymbol{W}_m$, where $\boldsymbol{W}_0 \in \mathbb{R}^{p \times h_1}$, $\boldsymbol{W}_i \in \mathbb{R}^{h_i \times h_{i+1}}$ for $i = 1, ..., m-1$ and $\boldsymbol{W}_m \in \mathbb{R}^{h_m \times 1}$. Denote $\boldsymbol{t}_i$ the intercept for the i^{th} hidden layer and b the intercept of the output layer. Let $\boldsymbol{\theta} = (\boldsymbol{W}_0, ..., \boldsymbol{W}_m, \boldsymbol{t}_1, ..., \boldsymbol{t}_m, b)$ be the parameters in the neural network model. For an input $\boldsymbol{x} \in \mathbb{R}^p$, denote the output

$$\eta_{\boldsymbol{\theta},\boldsymbol{x}} = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{4.1}$$

where in the regression set up, the output is from a linear layer and $\eta \in \mathbb{R}$, while in the classification, an extra sigmoid layer is added and $\eta \in (0, 1)$. Moreover, we assume sparse feature, i.e., only a small fraction of the variables are significantly related to the response. Without loss of generality, we assume

$$\mathcal{S}_0 = \{1, ..., s\}$$

of the variables are truly nonzero variables.

Define the loss function for regression to be the squared error loss

$$l(\boldsymbol{\theta}) = \mathbb{E}\left[(y-\eta)^2\right] \tag{4.2}$$

In practice, we work with the empirical loss

$$l(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = \frac{1}{n}\|\boldsymbol{y}-\boldsymbol{\eta}\|_2^2 \tag{4.3}$$

where $\boldsymbol{\eta} \in \mathbb{R}^n$ with $\eta_i = \eta_{\boldsymbol{\theta},\boldsymbol{x}_i}$, $i = 1, ..., n$. Define the loss function for classification to be the negative log-likelihood, which is known as the cross-entropy loss

$$l(\boldsymbol{\theta}) = \mathbb{E}\left[y \log \eta + (1-y)\log(1-\eta)\right] \tag{4.4}$$

In practice, we work with the empirical loss

$$l(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n}\left[y_i \log \eta_i + (1-y_i)\log(1-\eta_i)\right] \tag{4.5}$$

Let $\boldsymbol{G}_i$ be the gradient of the loss function with respect to $\boldsymbol{W}_i$ in the back propagation process for $i = 0, ..., m$, i.e.,

$$\boldsymbol{G}_i = \frac{\partial}{\partial \boldsymbol{W}_i} l(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}), \quad i = 0, ..., m \tag{4.6}$$

The DNP algorithm starts with the null model and adds one variable at a time. Let $\mathcal{S}$ be

the selected set and $\mathcal{C}$ be the candidate set. At the beginning, we have

$$\mathcal{S} = \{intercept\} \quad and \quad \mathcal{C} = \{1, ..., p\} \tag{4.7}$$

The model is trained on $\mathcal{S}$ only and the submatrix of $\boldsymbol{W}_0$ corresponding to the features in $\mathcal{C}$ is kept zero. After the training done, one chooses a l_q norm (usually with $q = 2$) and compute the gradient' norm for each $j \in \mathcal{C}$ of $\boldsymbol{W}_0$.

$$\boldsymbol{G}_{0j} = \frac{\partial}{\partial \boldsymbol{W}_{0j}} l(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}), \quad j \in \mathcal{C} \tag{4.8}$$

The next variable that enters the model, j_+ is

$$j_+ = \arg\max_{j \in \mathcal{C}} \|\boldsymbol{G}_{0j}\|_q \tag{4.9}$$

Then $\mathcal{S} = \mathcal{S}/j_+$ and $\mathcal{C} = \mathcal{C} \cup \{j_+\}$ To increase the stability, instead of computing $\boldsymbol{G}_{0j}$ directly, the DNP algorithm computes $\boldsymbol{G}_{0j}$ through the average over multiple dropouts. Let B_1 be the number of dropouts, the next variable is

$$j_+ = \arg\max_{j \in \mathcal{C}} \frac{1}{B_1} \sum_{b=1}^{B_1} \|\boldsymbol{G}_{b0j}\|_q \tag{4.10}$$

where $\boldsymbol{G}_{b0j}$ denotes the gradient of the loss function with respect to the first layer's j^{th} weight vector after the b^{th} random dropout.

The algorithm works because $\|\boldsymbol{G}_{0j}\|_q$ describes how much the loss function will change when the next update on the corresponding variable's weight is performed [102]. [125] also indicates that selecting variable by comparing $\|\boldsymbol{G}_{0j}\|_q$ has an equivalence to applying the

group lasso penalization, see also [85].

The algorithm works well at the very beginning, which is described by proposition 4.1 and proposition 4.2 in section 4.3. However, it suffers from a few disadvantages. First, as we include more correct variables in the model, the probability that we select another correct variable decreases. A simulation study in section 4.4 provides numeric support for this argument. Second, one needs to pre-specify how many variables need to be selected before stopping, denoted s_0. If this number is chosen to be more than the number of true variables, denoted s, there will be $s_0 - s$ variables that should not be included but was included, i.e., the false positive rate could be high. Finally, the model does not use dropout or regularization during prediction, which has potential over-fitting risk. Here we propose the ensemble neural network selection (ENNS) algorithm to remedy these issues, and we will discuss possible solutions in preventing over-fitting in the prediction step.

One could observe that when a fraction of $\mathcal{S}_0$ are already involved in the model, i.e., in $\mathcal{S}$, the model is trained such that these variables are used to explain the variations by all variables in $\mathcal{S}_0$. This weakens the effect of those truly nonzero variables in $\mathcal{C}$. These variables become less important than when there's no variable in $\mathcal{S}$. Moreover, there are less truly nonzero zero variables in $\mathcal{C}$ than at the very beginning, the probability that we select a correct variable in the next step is

$$\mathbb{P}(j_{next} \in \mathcal{S}_0) = \sum_{j \in \mathcal{S}_0 \cap \mathcal{C}} \mathbb{P}(j_{next} = j) \tag{4.11}$$

which will be even lower as $|\mathcal{S}_0 \cap \mathcal{C}|$ decreases. Therefore, there will be a nonzero probability that at one stage the selected variable does not belong to $\mathcal{S}_0$. We consider an ensemble method to remedy this issue.

The idea behind this ensemble method is similar to bagging [17, 19]. Assume that we want to add s_j variables in one step. Consider a bootstrap sample of size n_b from the original sample. The DNP with random initialization is trained on this sample, which yields a selection set $\mathcal{S}_1 = \{j_1, ..., j_{s_j}\}$. In stead of just doing one pass, we propose that for b_2 in $1, ..., B_2$ and a bootstrap sample size n_r, we perform the feature selection on a random selection of n_r observations. Denote the features being selected in all B_2 rounds as

$$\mathcal{J}_1 = \{j_{11}, ..., j_{1s_j}\}, ..., \mathcal{J}_{B_2} = \{j_{B_2 1}, ..., j_{B_2 s_j}\}$$

We will only allow a variable to enter the model if it appears at least $[B_2 p_s]$ times in the B_2 rounds, for a fixed proportion p_s. Mathematically,

$$\mathcal{J} = \{j \ in \ at \ least \ [B_2 p_s] \ of \ \mathcal{J}_1, ..., \mathcal{J}_{B_2}\}$$

is the set of variables that will actually enter the model in this step.

The reason that this ensemble will improve the selection lies on three points. First, the algorithm is an averaging of different bootstrapping results, thus the effect of some extreme observations could be averaged out. The final selection result represents the common part of the whole sample. Secondly, neural network uses random initialization. In two different training, though the predictions seem similar, the estimated parameters are actually from different local minimums of the loss function. Therefore, these different training results represent different aspects of the model. Combining the two reasons, if we select a smaller n_b compared to n, the selection results are closer to independent. However, n_b can not be too small to avoid misleading the neural network. Finally, if a variable is selected by mistake

in some round, this is possibly due to the specific bootstrap sample making the relationship between the variable and the response stronger, which is not general in all bootstrap samples. In practice, one will observe that though false selection happens, those false variables are different in different rounds. Therefore, this ensemble will actually make the probability of false selection tend to zero. This result is described by theorem 4.1 in section 4.3. Moreover, if two variables' interaction effect is important in the model, they are likely to be included in the model in the same step.

It's possible that the proposed method selects less or more variables than the number of variables we specified, s_j. If the sample is not enough to represent the true relationship between the variables and the response, it's very possible that the number of selected variables, denoted $\hat{s}_j$, is less than s_j. In this case, we exclude the variables that are already in C from the neural network and perform another round of variable selection with $\mathcal{S} = \{1, ..., p\}/\mathcal{C}$ and then $\mathcal{C} = \{intercept\}$. The number of variables to be selected in this round will be $s_j - \hat{s}_j$. On the other hand, $\hat{s}_j$ being more than s_j happens when the selection proportion p_s is specified too small. In this case, one would sort the variables by their appearing proportions and only select the first s_j variables in the list.

In summary, we specify a number s_0 at the very beginning, which mean the final model will include s_0 variables. In the j^{th} iteration, let s_j be the number of variables to be selected. Right now there are $|\mathcal{S}_j|$ variables in the model, denoted $\mathcal{S}_j$. Let $\boldsymbol{X}_{-n}$ be the sub-matrix of $\boldsymbol{X}$ where the columns with indices in $\mathcal{S}_j$ removed. Train the ensemble on $\boldsymbol{X}_{-n}$ and obtain selection result $\hat{\mathcal{S}}_j$. Let $s_{j+1} = s_j - |\hat{\mathcal{S}}_j|$. The algorithm is repeated until the model has selected s_0 variables. An algorithm is given in Algorithm 2. Under mild assumptions, the algorithm will finally reach selection consistency. This argument is described in theorem 4.2 in section 4.3. Moreover, a comparison of the variable selection performances of different

modeling is presented in section 4.4.

Algorithm 2: Algorithm for feature selection ENNS

```
Initialize number of selected variables S = ∅, s = 0 and target s_0;
while |S| < s_0 do
    for b = 1, ..., B do
        Bootstrap sampling;
        Random initialization with zero feature;
        Run the DNP algorithm and obtain selection set J_b;
    end
    Obtain J = ⋃_{b=1}^{B} J_b;
    Compute J_T by filtering number of appearance;
    if J_T <= s_0 − |S| then
        S = S ∪ J_T;
        Remove the columns in J_T from training data;
    else
        J_T is the m − s elements with highest number of appearance;
        S = S ∪ J_T;
    end
end
```

The computation complexity of the ENNS algorithm on a single machine is the number of bagged neural networks times the computation complexity of training a single neural network, which is equal to $O(Bhsnp)$. Here B is the number of bagged neural networks, h is the neural network structure complexity, s is the number of variables to be selected, n is the sample size and p is the variable dimension. However, since bagging algorithm has independent elements, it's easy to parallelize the bagged neural networks by submitting different jobs. In this case, the computation complexity reduces to $O(hsnp)$, which is the same as that of DNP in [82]. As a comparison, [82] also mentioned the computation complexity of HSIC-Lasso in [144], which grows cubicly with the sample size as $O(sn^3p)$.

4.2.2 Estimation With Regularization

In this subsection ,we will discuss possible procedures to prevent over-fitting. After feature selection, the deep neural network can be trained on the selected features. However, the number of parameters in the neural network model is still huge. A 4 hidden layer neural network with s selected variables and hidden layer sizes $h_1, ..., h_4$ has $sh_1 + h_1h_2 + h_2h_3 + h_3h_4 + h_4$ parameters (without counting the intercepts). For example, if we use the common hidden layer sizes $[50, 30, 15, 10]$ with the number of selected variables being 5, this brings 2466 parameters. As a comparison, the linear model has 6 parameters, while the GAM with 4 knots and degree 3 has 36 parameters. Compared to the number of parameters, the small sample size is still a challenging issue. Therefore, we need to be careful on the training of the neural network on the selected variables. A few methods are discussed below. Moreover, the Xavier initialization [59] is used here to assure that the initial weights are in a proper range.

4.2.2.1 Dropping Out and Bagging

In the variable selection step, over-fitting is overcome by dropout layers, where we randomly set parameters to zero in the later layers. However, using dropout layers in prediction is risky, since we are not able to measure the performance of doing a random dropout. One way is to use bagging again in this step. First, the connections in the estimated neural network, denoted $\mathcal{N}$ is randomly cut off, i.e., the weights are set to zero. By doing this we obtained $\mathcal{N}_r$, where r stands for reduced. Then a prediction is made on model $\mathcal{N}_r$, denoted $\hat{y}_r$. This process is repeated for K times. Denote the reduced neural networks to be $\mathcal{N}_{kr}$

and their predictions with $\hat{y}_{kr}$. In the regression set up, the final prediction is defined as

$$\hat{\boldsymbol{y}} = \frac{1}{n}\sum_{k=1}^{K}\hat{y}_{kr}$$

In the classification set up, the final prediction is defined as

$$\hat{y}_i = \begin{cases} 1, & if\ \hat{p}_i > p_c \\ & \\ 0, & if\ \hat{p}_i < p_c \end{cases}, \quad for\ i = 1, ..., n$$

where p_c is some pre-specified threshold.

$$\hat{p} = \frac{1}{n}\sum_{k=1}^{K}\hat{y}_{kr}$$

and $\hat{p}_i$ is the i^{th} element of $\hat{p}$. A simulation study is performed in Section 4.4.

4.2.2.2 Stage-wise Training

The stage-wise training idea comes from [82], where the authors used it as a step-wise variable selection technique. However, here we adopt the idea to train the final model on the selected variables. The intuition behind this is that at each step, the information that is already trained remains in the training process. Therefore, adding a new variable adjusts the previous trained weights. Moreover, training with adaptive gradient algorithm (Adagrad, [40]) allows adaptive learning rates for different parameters and thus ensures faster and more accurate convergence. In detail, assume that we have selected m variables $\mathcal{J}$ from the ENNS algorithm. Let $\boldsymbol{X}_{\mathcal{J}}$ be the sub-matrix of $\boldsymbol{X}$ whose columns' indices are in $\mathcal{J}$. Then the DNP algorithm in [82] is trained on $\boldsymbol{X}_{\mathcal{J}}$ with $|\mathcal{J}|$ being the target number of variables. A

simulation study is performed in Section 4.4.

4.2.2.3 L1 Norm Regularization

It's mentioned that l_1 regularization gives sparse neural network and controls over-fitting by shrinking parameters towards zero, and some parameters can be shrunk to zero exactly. Therefore, we choose to use l_1 norm regularization to control the parameter size and the number of nonzero parameters.

Let $\hat{S}$ be the set of indices of the variables that are selected from the first step. Let $\boldsymbol{\Theta} = \theta_1, ..., \theta_L$ be the hidden layer weights and $\boldsymbol{T} = t_1, ..., t_L$ be the hidden layer intercepts (including the output layer). Let $f(x; \boldsymbol{\Theta}, \boldsymbol{T})$ be the neural network structure with such parameters that maps the original input to the output. In the classification problem, define

$$\hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{T}} = \underset{\boldsymbol{\Theta}, \boldsymbol{T}}{\arg\min} -\frac{1}{n} \sum_{i=1}^{n} \left[y_i f(\boldsymbol{x}_{\hat{S},i}) - \log(1 + \exp(f(\boldsymbol{x}_{\hat{S},i}))) \right] + \sum_{l=1}^{L} \lambda_{nl} |\theta_L|, \tag{4.12}$$

where $\boldsymbol{x}_{\hat{S},i}$ denotes the i^{th} observation with only the selection variables included.

A direct training of the loss function 4.12 with the built-in l_1 loss penalty directly added to the cross-entropy loss does not work well in the current neural network libraries including tensorflow and pytorch. Therefore, a coordinate descent algorithm is needed to obtain sparsity in the neural network. Define the soft-thresholding operator $S(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ as

$$(S(\boldsymbol{x}, c))_i = sign(x_i)(|x_i| - c)_+, \quad i = 1, ..., d. \tag{4.13}$$

The algorithm consists an iterative process of updating the neural network weights without the l_1 penalty and then applying the soft-thresholding operator 4.13. The number of epochs is pre-specified. However, the performance on the validation set can be monitored

and an early-stopping criterion can be specified. The training will be stopped if the performance on the validation set does not improve for a pre-specified number of patience level. It worth noting that instead of selecting the tuning parameter, a sparsity level of each layer can be specified. Assuming there are M hidden layers with sizes $h_1, h_2, ..., h_M$. One may specify percentile p_m for $m = 1, ..., M$. Denote W_m the weight of layer m and W_{pm} the p_m^{th} percentile of W_{pm}. Then for layer m, the soft-thresholding operator can be applied as $S(W_m, W_{pm})$. For example, choosing a percentile of 50 will make a certain layer have 50% sparsity level. An algorithm is given in Algorithm 3. A simulation that compares the built-in l_1 penalty and the soft-thresholding operator is given is Section 4.4.

Algorithm 3: Algorithm for l_1 norm estimation using coordinate descent

```
Initialize the weights with Xavier initialization;
while Early stopping False OR epochs < k do
    One step gradient descent for the neural network part;
    for weights in layers do
        Apply the soft-thresholding function with a pre-specified percentile;
    end
    Check early stopping criterion;
end
```

4.3 Theoretical Guarantee

In this section, we will develop theoretical supports for the proposed methodology. The methodology supports the intuitions used in the method. A few assumptions are made in the derivations of the theorems. The first famous assumption in high-dimensional modeling is sparsity. Here we provide two versions of sparsity: a stronger version and a weaker version. These are stated in assumption 4.1 and assumption 4.2.

Assumption 4.1 (Sparsity (weak))**.** The features are sparse, i.e., only $s < n = o(p)$ of the

p variables are strongly related with the response. Specifically, if y depends on x through a linear relationship with coefficients $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)^T$, we have

$$\min_{j=1,...,s} |\beta_j| \geq \gamma_n \qquad and \qquad \sum_{j=s+1}^{p} |\beta|_j = \tau_n = o(\gamma_n) \tag{4.14}$$

where γ_n is a sequence that may go to zero as n goes to infinity.

Assumption 4.2 (Sparsity (strong))**.** The features are sparse, i.e., only $s < n = o(p)$ of the p variables are related with the response. Specifically, if y depends on x through a linear relationship with coefficients $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)^T$, we have

$$\min_{j=1,...,s} |\beta_j| \geq \gamma_n \qquad and \qquad \sum_{j=s+1}^{p} |\beta|_j = 0 \tag{4.15}$$

where γ_n is a sequence that may go to zero as n goes to infinity.

We know that in most cases the predictors are dependent, or at least weak correlation exists. However, even weak correlation put great complexity in the methodology. Therefore, we assume independent predictors in this section when deriving methodology. Extensive simulation study will cover the cases when the predictors are correlated. Though the theorems are proved under this assumption, the results can be extended to the assumption that the correlation are up to $o(1)$.

Assumption 4.3 (Independence)**.** The predictors in the design matrix satisfy

$$cor(x_j, x_k) = 0, \ 1 \leq j < k \leq p \tag{4.16}$$

The behavior of the design matrix should also be controlled. Here we consider a random

design and assume the following assumption

Assumption 4.4 (Design matrix)**.** The covariate vector $\boldsymbol{x}$ has a continuous density and there exist constants C_1 and C_2 such that the density function g_j of $\boldsymbol{x}_j$ satisfies $0 < C_1 \leq g_j(x) \leq C_2 < \infty$ on $[a, b]$ for every $1 \leq j \leq p$.

As a typical assumption for bagging, we require the bagging sample proportion to be not too small.

Assumption 4.5 (Sample proportion)**.** In each bagging round, every sample has q_n probability to be included, where q_n satisfies

$$nq_n \to \infty \quad as\ n \to \infty$$

Note that if we choose the bootstrap sample size to be the same as the sample size n, by law of large numbers, we have approximately $q_n = 1 - 1/e$.

The following two propositions considers a scenario that the true underlying relationship between the predictors and the response is linear, which demonstrates how the probability of choosing one variable over another in the first step is decided. The first proposition gives the probability that we select one variable over another, and the second proposition gives the probability that we will select a correct variable in the first step.

Proposition 4.1. Consider the case where y depends on x through a linear structure with coefficients $\beta_1, ..., \beta_p$ which satisfy assumption 4.1. Also under assumption 4.3 and 4.4, if the sub-matrix of $\boldsymbol{x}$ consisting of the columns corresponding to the nonzero coefficients is column-wise orthogonal, let c_j be the criterion to select predictor j. Recall that we will select predictor j if $j = \arg\max_j c_j$, where c_j is the L_2 norm of the gradient with respect to

the j^{th} input. Then we have

$$\begin{aligned}\mathbb{P}(c_j < c_k) =& 2L\left(\frac{|\beta_j|-|\beta_k|}{\sqrt{2}\sigma}, -\frac{|\beta_k|}{\sigma}, \frac{1}{\sqrt{2}}\right) + 2L\left(\frac{|\beta_j|+|\beta_k|}{\sqrt{2}\sigma}, \frac{|\beta_k|}{\sigma}, \frac{1}{\sqrt{2}}\right) + \\ & \Phi\left(\frac{|\beta_j|-|\beta_k|}{\sqrt{2}\sigma}\right) + \Phi\left(\frac{|\beta_j|+|\beta_k|}{\sqrt{2}\sigma}\right) - 2 \end{aligned} \tag{4.17}$$

where

$$L(a, b, \rho) = \mathbb{P}(X_1 > a, X_2 > b) \tag{4.18}$$

is the bivariate orthant probability with correlation ρ and $\Phi(\cdot)$ is the standard normal distribution CDF.

Proposition 4.2. Under assumptions 4.1, 4.3 and 4.4, the probability that we select a nonzero predictor at the very beginning using the stage-wise neural network selection is

$$\mathbb{P}(\text{A nonzero predictor enters the model first}) = \sum_{k=1}^{s} \int_0^\infty f_k(x) \prod_{j \neq k}^{p} F_j(x) dx \tag{4.19}$$

where

$$F_k(x) = \frac{1}{2}\left[erf\left(\frac{x+|\beta_k|}{\sqrt{2\sigma^2}}\right) + erf\left(\frac{x-|\beta_k|}{\sqrt{2\sigma^2}}\right)\right]$$

and

$$f_k(x) = \frac{\partial}{\partial x} F_k(x) = \sqrt{\frac{2}{\pi\sigma^2}} e^{-\frac{x^2+\beta_k^2}{2\sigma^2}} \cosh\frac{\beta_k x}{\sigma^2} \tag{4.20}$$

and $erf(\cdot)$ is the error function. Moreover, if $\beta_{max} = \max_{j=1,\ldots,s}$ is bounded, as $s \to \infty$,

the probability is bounded from above

$$\mathbb{P} \leq 1 - \delta$$

where δ is a nontrivial quantity.

Proofs of the two propositions are given in the Appendix. Proposition 4.1 and 4.2 describe the behavior of neural network stage-wise algorithm at the very beginning. The probability that we select one predictor over another depends on the sum of their signal strength and the difference of their signal strength. The greater the difference, the higher probability that we will select the predictor with higher signal strength. The probability that we will select a correct predictor at the very beginning is described by the error function and standard normal density functions. Though the form of the probability looks complicated, since error function can be approximated by an exponential function with proper constants, we will be able to show that in some cases, it is not guaranteed that a variable entering the model first is a nonzero variable. Specifically, this happens when we have a low signal strength or a huge number of candidate variables.

So there is a concern that a wrong variable will mistakenly enter the model due to a special training case of the neural network model, as shown in the previous proposition. With the bagging algorithm, we are able to eliminate the false positive selections with probability tending to 1. The intuition is that false positive selection of a certain predictor happens due to a specific observation of the design matrix, which appears to be more correlated to the response or residual. However, with different sub-samplings, it's very unlikely that they yield the same wrong selection. This property is captured by the following theorem.

Theorem 4.1. If one of the two following cases happen, then in each selection step of the

ENNS algorithm, the probability of false positive converges to zero, i.e.

$$\mathbb{P}(j \in \hat{\mathcal{S}} \; and \; j \notin \mathcal{S}) \to 0 \quad as \; n \to \infty \; and \; B_2 \to \infty \tag{4.21}$$

(a) Under assumptions 4.1, 4.3, 4.4 and 4.5, also assume that

$$s_0 \leq s \quad and \quad \frac{p - s_0}{s} e^{\tau_n - \gamma_n} \to 0 \; as \; n \to \infty$$

where τ_n and γ_n are defined in assumption 4.1.

(b) Under assumptions 4.2 4.3, 4.4 and 4.5, also assume that the number of variables to be selected satisfies

$$s_0 \leq C \cdot s = o(p)$$

for some constant C.

A proof is given in the appendix. In variable selection algorithms, the most important property is to be able to selection the correct predictors consistently. Here we show that ENNS enjoys this property in the following theorem.

Theorem 4.2. Under assumptions 4.2, 4.3, 4.4 and 4.5, let K_n be the upper bound of the norm of the best parameters in the neural network model when $\mathcal{S}$ is included, and K be the size of the first hidden layer, with the ensemble, if γ_n satisfies

$$K \frac{\log(p - s)}{n} \log\left(1 - \frac{1}{2} e^{-c\gamma_n^2}\right) \to 0 \quad as \; n \to \infty$$

for some constant c, and

$$K_n^2\sqrt{\frac{\log(nK_n)}{n}} \to \infty \quad as\ n \to \infty$$

the probability that all nonzero variables are selected converges to 1, i.e.,

$$\mathbb{P}(\hat{\mathcal{S}} = \mathcal{S}) \to 1 \quad as\ n \to \infty\ and\ B_2 \to \infty \tag{4.22}$$

A proof is given in the appendix. In theorem 4.2, we showed that with strong enough signal strength in the true nonzero variables, the algorithm will be able to select all nonzero variables with probability tending to 1. The conditions are not verifiable in practice, however, extensive examples in section 4.4 shows that the ENNS algorithm reaches selection consistency easier than the other algorithms.

For the estimation step, there has been some discussion about the asymptotic properties such as [53, 146], where the results of using sparse group lasso penalty are given. The l_1 norm penalty is actually a special case of the sparse group lasso with the lasso parameter being 1 and the group lasso parameter being 0. Therefore, the results of these papers hold as long as we have $\hat{\mathcal{S}} = \mathcal{S}_0$, which has probability tending to 1 by theorem 4.2. Here we will adapt the theory in [61] and will provide the following result.

Theorem 4.3. Assume assumptions 4.2, 4.3, 4.4 and 4.5, consider the variables selected from the ENNS algorithm and the estimation with the l_1 regularization method. Denote the l_1 regularization tuning parameter with λ_n and the corresponding Lagrangian parameter K_n. Denote the hidden layer size with k_n. In the regression set up, assume $\mathbb{E}(Y^2) < \infty$, if

$K_n \to \infty$, $k_n \to \infty$ and $k_n s^2 K_n^4 \log(k_n s K_n^2)/n \to 0$, we have

$$\lim_{n\to\infty, B_2\to\infty} \mathbb{P}\left(\mathbb{E}\int |f_n(\boldsymbol{x}) - f(\boldsymbol{x})|^2 \mu(dx) \to 0\right) = 1$$

where f_n is the estimated neural network and f is the true function. In the classification set up, assuming that the probability of response being 1 is bounded away from 0 and 1 by $\tilde{\epsilon}$, denote with Q the maximum number of equivalent neural network classes, choosing tuning parameter $\lambda_n \geq c\sqrt{k_n \log n/n}(\sqrt{\log Q} + \sqrt{k_n \log s}\log(nk_n)$, if $\log(n)/(n\tilde{\epsilon}^2) \to 0$, $s^2 k_n \lambda_n^2/(n\tilde{\epsilon}^2) \to 0$ and $n^{-1}k_n^{9/2}s^{5/2}\sqrt{\log(s_n)} \to 0$ as $n \to \infty$, we have

$$\lim_{n\to\infty, B_2\to\infty} \mathbb{P}\left(R(f_n) - R(f^*) \to 0\right) = 1$$

where $R(f_n)$ is the risk of neural network classifier and $R(f^*)$ is the risk of Bayes classifier.

Theorem 4.3 states that under the previously discussed conditions, the regression reaches weak estimation consistency of the non-parametric function defined in [61]. For the classification, the neural network classifier's risk tends to the optimal risk, Bayes risk, see for example [37]. The theorem is a direct result from the existing results of the low dimension neural network regression model and classifiers. Conditioning on the fact that we can select all correct variables with probability tending to 1, applying the full probability formula, the consistency of the two-step approach can be derived with the low dimensional consistency plus the probability of non-selection-consistency.

The consistency error comes from two aspects: the variable selection error and the estimation error. The intuition behind this is that with a wrong selection result, the estimation error may be big, however, this happens with probability tending to zero. With a correct

selection result, the estimation error behaves the same as in the low dimensional case, which converges to zero.

4.4 Simulation Study

In this section, we use a few examples as numerical supports to our arguments in the previous sections. The code for DNP is composed according to the algorithm outline in [82], and the code of ENNS is composed according to the algorithm in this paper. Both codes can be found at `https://github.com/KaixuYang/ENNS`.

4.4.1 Stage-wise Correct Selection Probability Decreasing Study

In this subsection, we use an example to demonstrate that the chance of selecting a correct variable in a stage-wise neural network decreases as we have more correct variables in the model. Consider a design matrix $\boldsymbol{X}$ that is drawn from a uniform distribution between -1 and 1. The sample size is set to $n = 1000$ and the number of predictors is set to $p = 10000$. $s = 5$ of the predictors are related with the response. We consider three different true structures of the relationship between the predictors and the response: linear, additive non-linear and neural network. For the response, we consider two different cases: regression and classification. In the linear case, the coefficients are drawn from a standard normal distribution. In the additive non-linear case, the functions are set to

$$\eta = \sin(x_1) + x_2 + \exp(x_3) + x_4^2 + \log(x_5 + 2) - 2 \tag{4.23}$$

where $y = \eta + \epsilon$ in the linear case and $prob = \sigma(\eta)$ in the classification case. In the neural network case, we set hidden layers as $[50, 30, 15, 10]$ and weights from standard normal distribution.

For each of the cases, we test the cases when we start from 0 to 4 correct predictors. In order to eliminate the effect of different signal strength from different predictors, we random sample i indices from $1, ..., 5$ as the selected variables, for $i = 0, ..., 4$, and include these i indices predictors as the initially selected variables. We run a repetition of 1000 times and report the proportion that the next variable that enters the model is a correct predictor. The results are summarized in table 4.1.

y	structure	0 variable	1 variable	2 variables	3 variables	4 variables
Reg	Linear	0.995(0.002)	0.952(0.006)	0.863(0.010)	0.774(0.013)	0.430(0.016)
	Additive	0.993(0.003)	0.847(0.011)	0.905(0.009)	0.794(0.012)	0.531(0.016)
	NN	0.998(0.001)	0.971(0.005)	0.932(0.007)	0.788(0.013)	0.574(0.016)
Cls	Linear	0.989(0.003)	0.918(0.009)	0.873(0.009)	0.813(0.011)	0.552(0.016)
	Additive	0.992(0.003)	0.957(0.006)	0.911(0.009)	0.706(0.014)	0.633(0.015)
	NN	0.994(0.002)	0.968(0.006)	0.947(0.004)	0.895(0.009)	0.762(0.013)

Table 4.1: The proportion of correct variable selection after 0-4 correct variables in the model, for different cases over 1000 repetitions. The results show the mean. The results show three different data generation structures: linear, additive non-linear and neural network for both regression and classification.

In the table, we see that the probability of selecting a correct predictor decreases as we have more correct predictors in the model, in all cases. The only exception is in the regression set up with additive non-linear structure from 1 variable to 2 variables, which may due to random error.

4.4.2 False Positive Rate Study

In this subsection, we use an example to demonstrate that the false positive rate of ENNS (the probability of selecting a wrong variable) is superior than the pure stage-wise algorithm.

Note that if one set the number of variables to be s, stage wise algorithm always select 5 variables, while ENNS will stop if there isn't any new variable that satisfy the condition to be added. Therefore, it's possible that ENNS selects less number of variables and avoid wrong selection. We used the same setup as [82] to generate responses. Two different types of responses including regression and classification will be considered here. The input variable $\boldsymbol{X}$ was drawn from $U(-1,1)$, where the feature dimension p was fixed to be $10,000$. The corresponding labels were obtained by passing $\boldsymbol{X}$ into the feed forward neural network with hidden layer sizes $\{50,30,15,10\}$ and ReLU activation functions. Input weights connecting the first s inputs were randomly sampled from $N(1,1)$ for regression and $N(0,1)$ for classification. The remaining input weights were kept zero. For each $s = 2,5,10$, we generated 1000 training samples. In table 4.2, we report the false positive rate between the ENNS algorithm and the neural network stage-wise algorithm.

Response	Method	s=2	s=5	s=10
Regression	ENNS	10.4%(21.5%)	11.5%(22.1%)	12.8%(23.6%)
	DNP	22.5%(29.5%)	30.2%(28.7%)	41.4%(33.2%)
Classification	ENNS	4.7%(17.9%)	7.4%(18.6%)	9.8%(20.3%)
	DNP	16.5%(24.4%)	24.8%(29.7%)	40.5%(32.8%)

Table 4.2: Selection false positive rate average of the ENNS and DNP under different number of true variables in 101 repetitions. Standard deviations are given in parenthesis.

It can be tested that the ENNS's false positive rate is significantly less than the false positive rate of DNP under significance level $\alpha = 0.05$. This provides strong evidence that the ENNS is useful in reducing the probability of selecting an incorrect variable.

4.4.3 Variable Selection Simulation Study

In this subsection, we study the variable selection capability of the ensemble neural network selection (ENNS) algorithm in the complicated set up. We used similar setup as in the last

subsection to generate responses. Two different types of responses including regression and classification will be considered here. The input variable $\boldsymbol{X}$ was drawn from $U(-1,1)$, where the feature dimension p was fixed to be $10,000$. The corresponding labels were obtained by passing $\boldsymbol{X}$ into the feed forward neural network with hidden layer sizes $\{50,30,15,10\}$ and ReLU activation functions. Input weights connecting the first s inputs were randomly sampled from $N(0,2)$ for regression and $N(0,1)$ for classification. The remaining input weights were kept zero. The DNP model was coded according to their algorithm outline in python with pyTorch. The ENNS algorithm is based on the DNP algorithm with an ensemble wrapper. The LASSO [126] is implemented by the scikit learn library, and the HSIC lasso [144] is implemented using the HSICLasso library. In all four algorithms, the number of selected variables are strictly restricted to the same as the true number of variables. In the ENNS, we run a bagging of 10 rounds with selection proportion 0.3. We report the average number of correct variables that are selected on 101 repetitions of the data generation in table 4.3.

Response	Method	s=2	s=5	s=10
Regression	ENNS	1.73(0.52)	4.21(0.56)	9.25(1.11)
	DNP	1.61(0.50)	3.92(0.56)	8.77(1.13)
	LASSO	1.65(0.57)	3.87(0.62)	9.62(1.38)
	HSIC-LASSO	1.67(0.47)	3.80(0.66)	3.61(1.17)
Classification	ENNS	1.81(0.49)	4.24(0.87)	8.04(1.25)
	DNP	1.67(0.76)	3.76(1.06)	5.95(1.29)
	LASSO	1.67(0.56)	3.76(0.75)	5.76(1.38)
	HSIC-LASSO	1.67(0.47)	2.80(0.91)	3.61(1.17)

Table 4.3: Variable selection capacity of ENNS and other methods with low signal strength in the regression (top) and classification (bottom) set up. The numbers reported are the average number of selected variables which are truly nonzero. The standard errors are given in parenthesis.

We observe that the ENNS outperforms the other variable selection algorithms in all three cases, and the difference is significant when $s = 10$ under a t-test. The ENNS performs

better when there are more nonzero variables. None of the algorithms were able to reconstruct the original feature indices due to a few reasons: the sample size is relatively small compared to the number of variables; the data generation through neural network structures is complicated; the signal strength is relatively low.

To fully study the variable selection power of the ENNS algorithm, we implemented another simulation case in classification where we have a higher signal strength while keeping all other conditions the same. In this simulation study, we increase the mean of the weights of the nonzero variables to 3.5. With the same implementations, we summarize the results in table 4.4. Moreover, table 4.5 summarizes the results for signal strength 10.

Method	s=2	s=5	s=10
ENNS	2.00(0.00)	4.71(0.55)	8.38(2.06)
DNP	1.86(0.35)	4.38(0.84)	7.43(2.36)
LASSO	1.81(0.39)	4.19(1.01)	7.47(2.40)
HSIC-LASSO	1.71(0.45)	3.71(1.12)	4.95(2.13)

Table 4.4: Variable selection capacity of ENNS and other methods with normal signal strength. The numbers reported are the average number of selected variables which are truly nonzero. The standard errors are given in parenthesis.

The ENNS reaches selection consistency when $s = 2$, while the other compared algorithms still do not have selection consistency. However, all algorithms have obvious improvements in all cases. We have to admit that selecting the correct subset of variables in all 101 repetitions is extremely challenging, since the data have great variable in different repetitions. Moreover, when s gets greater, the importance of a few variables are less likely to be observed from the data.

Moreover, as we know, the bagging algorithm can be paralyzed since different runs are independent of each other. Therefore, the computational efficiency of this variable selection algorithm is almost the same as the computation efficiency of a single run.

Method	s=2	s=5	s=10
ENNS	2.00(0.00)	5.00(0.00)	9.90(0.29)
DNP	2.00(0.00)	5.00(0.00)	9.47(1.10)
LASSO	2.00(0.00)	4.90(0.29)	9.23(1.74)
HSIC-LASSO	2.00(0.00)	4.62(0.84)	7.76(2.76)

Table 4.5: Variable selection capacity of ENNS and other methods with high signal strength. The numbers reported are the average number of selected variables which are truly nonzero. The standard errors are given in parenthesis.

4.4.4 Estimation Simulation Study

In this subsection, we compare the estimation methods in section 4.2. To fully study the difference between these methods without the effects of other factors, in this subsection we assume correct selection and perform the estimation on the correct subset of variables. The data are generated according to the same scheme as in the last subsection. We will compare the performance of these different estimation methods for $s = 2, 5, 10$ assuming that we know the correct subset of variables. The simulation is run on 101 repetitions of data generation using different seeds. In the results, we report the RMSE, the MAE and the MAPE for regression, and the accuracy, the auc score and the f1 score for classification. These are in table 4.6.

On average, we see l_1 norm regularization gives best performance, except for the MAPE of $s = 10$ in regression. Moreover, we observe that both built-in l_1 and soft-thresholding gives smaller standard errors, which coincides with the shrinkage estimator's properties. However, soft-thresholding provides better performance in average than built-in. The reason is that sparsity is not well supported with most libraries, thus a manual operation is needed to obtain sparsity.

4.4.5 Variable Selection and Estimation

In this subsection, we study the prediction capability of the two-stage approach – ENNS algorithm with l_1 neural network, and compare it with the DNP model, the logistic regression and the HSIC-lasso with SVM. We use the same neural network structure to generate data as in this section. Over 101 repetitions, we report the average RMSE (rooted mean squared error), MAE (mean absolute error) and MAPE (mean absolute percent error) for regression and average accuracy, AUC and F1 Score for classification, as well as their standard errors. The results are summarized in table 4.7.

We observe that our proposed algorithm obtained a slight performance boost via the ensemble method. Moreover, the standard errors of these results are slight greater than the standard errors in the last subsection, where the estimation was done assuming correct selection. The increase of standard errors is mainly due to the selection variations.

4.4.6 Correlated Predictors

In this subsection, we use an example to study the numerical performance of the proposed algorithm in correlated predictor situation. We will consider two different correlations: $\rho = 0.3$ and $\rho = 0.7$. As a comparison, the results for $\rho = 0$ will also be included. Let $u_1, ..., u_n$ be i.i.d. standard normal random variables, x_{ij} be i.i.d. standard normal random variables, which are independent of $u_1, ..., u_n$, for $i = 1, ..., n$ and $j = 1, ..., p$. Do the transformation $x_{ij} = (x_{ij} + tu_i)/\sqrt{1+t^2}$ for some t, then we obtained standard normal correlated variables

$$cor(x_{ij}, x_{ik}) = \frac{t^2}{1+t^2},\ i = 1, ..., n; j = 1, ..., p$$

Taking $t = \sqrt{3/7}$ gives correlation 0.3 and taking $t = \sqrt{7/3}$ gives correlation 0.7. Then we truncate the random variables to interval $[-1, 1]$. The structure to generate response is kept the same as in the last subsection. The results of variable selection and estimation is given in table 4.8.

y	Metric	Method	s=2	s=5	s=10
Reg	RMSE	Neural Network	31.24(13.32)	69.46(37.40)	136.64(60.54)
		Xavier	18.64(11.07)	58.89(27.73)	136.58(65.57)
		l_1 built-in	20.47(9.62)	59.37(23.61)	129.55(50.48)
		l_1 **soft**	**5.97(4.18)**	**45.83(33.06)**	**109.31(47.24)**
		Stage-wise	10.59(11.20)	47.64(22.69)	117.65(43.96)
		Bagging	25.48(10.89)	59.49(26.53)	133.45(59.72)
	MAE	Neural Network	16.45(10.91)	52.85(28.47)	103.76(45.99)
		Xavier	13.65(8.06)	45.34(22.18)	105.17(53.66)
		l_1 built-in	15.56(7.76)	45.32(18.34)	98.54(38.36)
		l_1 **soft**	**4.37(3.02)**	**35.49(26.21)**	**83.85(36.45)**
		Stage-wise	7.91(8.23)	38.86(20.00)	89.82(33.82)
		Bagging	14.77(7.92)	43.16(20.51)	99.38(41.66)
	MAPE	Neural Network	0.012(0.015)	0.030(0.026)	0.033(0.026)
		Xavier	0.009(0.009)	0.027(0.022)	0.029(0.017)
		l_1 built-in	0.011(0.012)	0.029(0.023)	0.032(0.021)
		l_1 **soft**	**0.005(0.007)**	**0.017(0.010)**	0.029(0.023)
		Stage-wise	0.007(0.007)	0.019(0.015)	**0.027(0.016)**
		Bagging	0.010(0.010)	0.026(0.024)	0.030(0.022)
Cls	Accuracy	Neural Network	0.944(0.026)	0.886(0.037)	0.841(0.041)
		Xavier	0.952(0.026)	0.891(0.034)	0.831(0.036)
		l_1 built-in	0.927(0.031)	0.844(0.085)	0.752(0.110)
		l_1 **soft**	**0.964(0.028)**	**0.908(0.029)**	**0.855(0.031)**
		Stage-wise	0.945(0.030)	0.886(0.038)	0.804(0.042)
		Bagging	0.877(0.069)	0.806(0.068)	0.753(0.087)
	AUC	Neural Network	0.942(0.027)	0.882(0.038)	0.837(0.042)
		Xavier	0.951(0.027)	0.891(0.034)	0.825(0.037)
		l_1 built-in	0.924(0.031)	0.833(0.100)	0.734(0.123)
		l_1 **soft**	**0.964(0.029)**	**0.905(0.029)**	**0.851(0.032)**
		Stage-wise	0.943(0.031)	0.884(0.038)	0.800(0.041)
		Bagging	0.877(0.065)	0.803(0.063)	0.751(0.084)
	F1 Score	Neural Network	0.943(0.027)	0.887(0.045)	0.841(0.049)
		Xavier	0.952(0.026)	0.892(0.041)	0.832(0.048)
		l_1 built-in	0.927(0.031)	0.824(0.192)	0.732(0.200)
		l_1 **soft**	**0.965(0.026)**	**0.908(0.036)**	**0.857(0.033)**
		Stage-wise	0.944(0.031)	0.883(0.042)	0.806(0.049)
		Bagging	0.870(0.077)	0.792(0.060)	0.748(0.088)

Table 4.6: Prediction results on the testing set using neural networks with and without l_1 norm regularization for $s = 2, 5, 10$. RMSE is rooted mean squared error, MAE is mean absolute error, and MAPE is mean absolute percent error. Accuracy is the percentage of correct prediction, auc is area under the ROC curve, and f1 score is the inverse of inverse precision plus the inverse recall.

y	Metric	Method	s=2	s=5	s=10
Reg	RMSE	**ENNS+l1**	**15.67(30.35)**	**48.14(21.16)**	**174.08(65.38)**
		DNP	25.42(33.16)	62.63(29.02)	**178.91(60.15)**
		Lasso	79.44(67.31)	104.19(49.38)	192.04(77.34)
		HSIC-Lasso	56.32(59.41)	86.77(47.51)	188.35(56.48)
	MAE	**ENNS+l1**	**12.03(23.68)**	**40.12(19.95)**	**132.07(44.99)**
		DNP	20.15(27.15)	47.85(22.31)	**136.06(45.95)**
		Lasso	64.11(54.63)	81.97(39.76)	147.86(60.21)
		HSIC-Lasso	42.89(34.66)	70.04(41.23)	144.37(48.15)
	MAPE	**ENNS+l1**	**0.012(0.025)**	**0.028(0.036)**	**0.041(0.037)**
		DNP	0.017(0.028)	0.032(0.032)	**0.042(0.041)**
		Lasso	0.042(0.029)	0.046(0.035)	0.046(0.025)
		HSIC-Lasso	0.033(0.021)	0.036(0.025)	0.048(0.024)
Cls	Accuracy	**ENNS+l1**	**0.967(0.029)**	**0.848(0.025)**	**0.756(0.067)**
		DNP	0.933(0.076)	0.822(0.068)	0.736(0.064)
		Lasso	0.732(0.103)	0.726(0.071)	0.692(0.075)
		HSIC-Lasso	0.805(0.094)	0.798(0.094)	0.706(0.081)
	AUC	**ENNS+l1**	**0.959(0.036)**	**0.834(0.024)**	**0.709(0.058)**
		DNP	0.898(0.148)	0.780(0.100)	0.699(0.052)
		Lasso	0.652(0.121)	0.640(0.102)	0.625(0.068)
		HSIC-Lasso	0.774(0.125)	0.748(0.121)	0.677(0.061)
	F1-Score	**ENNS+l1**	**0.962(0.037)**	**0.859(0.036)**	**0.708(0.089)**
		DNP	0.903(0.208)	0.761(0.199)	0.705(0.100)
		Lasso	0.590(0.299)	0.604(0.250)	0.634(0.192)
		HSIC-Lasso	0.744(0.206)	0.731(0.242)	0.666(0.208)

Table 4.7: Model performance of the combination of ENNS algorithm and l_1 thresholding estimation, compared with DNP, Lasso and HSIC-Lasso for $s = 2, 5, 10$ cases in both regression and classification. The average performance of 101 repetitions with their standard errors in parenthesis are presented.

Res	Model	selection			estimation		
		$\rho = 0.0$	$\rho = 0.3$	$\rho = 0.7$	$\rho = 0.0$	$\rho = 0.3$	$\rho = 0.7$
Reg	ENNS+l_1	3.81(0.79)	3.27(0.75)	2.29(0.70)	40.82(19.46)	37.17(27.29)	43.18(44.47)
	DNP	3.48(0.96)	2.95(0.79)	2.14(0.56)	81.43(46.00)	92.91(65.25)	101.15(90.63)
	LASSO	3.38(0.90)	2.85(0.79)	2.11(1.12)	131.37(74.22)	151.16(108.88)	113.30(97.54)
Cls	ENNS+l_1	3.66(1.05)	3.25(0.76)	2.38(0.72)	0.856(0.040)	0.875(0.061)	0.907(0.030)
	DNP	3.62(1.09)	3.43(0.91)	2.71(1.03)	0.774(0.100)	0.766(0.106)	0.793(0.092)
	LASSO	3.55(0.79)	2.90(1.31)	1.95(0.72)	0.598(0.083)	0.634(0.117)	0.683(0.116)

Table 4.8: Selection and estimation comparison for predictors with correlation 0, 0.3 and 0.7. The number of nonzero predictors is set to 5. For selection, the average number of correct selected variables with its standard error is given. For estimation the average RMSE or AUC with their standard errors is given. The results are averaged over 101 repetitions.

From the table we see that in the correlated cases the model works almost as well as when there's no correlation. All models select less variables when the correlation is higher, and this is a well-known symptom of variable selection with correlated predictors. However, this does not affect the estimation step, and in some cases even makes the estimation results better. The reason could be that we have less variables thus the model is simpler. Since the predictors are correlated, we do not lose too much information by not selecting some of them. Moreover, some results not in the table includes the false positive rate, where the average for ENNS is 0.05 ± 0.03, while that of the DNP is 0.29 ± 0.14. Therefore, ENNS includes less redundant variables in the estimation step and achieves better performance.

4.5 Real Data examples

In this section, we evaluate the performance of the two-step model on some real world data sets.

4.5.1 Variable Selection: MRI Data

In this example, we evaluate the variable selection capability with other variable selection models, and compare the results with the biological ground truth. The data used in this example come from Alzheimer's disease neuroimaging initiatives (ADNI), see `http://adni.loni.usc.edu/`. The data includes $n = 265$ patients' neuroimaging results, including 164 Alzheimer's (AD) patients and 101 cognitively normal (CN) individuals. 145 regions of interested (ROIs) spanning the entire brain were calculated using Multi-Atlas ROI segmentation, and 114 ROIs were derived by combining single ROIs within a tree hierarchy to obtain volumetric measurements from larger structures. Therefore, $p = 259$ ROIs were used in this

example. Details of the pre-processing method can be found at `https://adni.bitbucket.\io/reference/docs/UPENN_ROI_MARS/Multi-atlas_ROI_ADNI_Methods_mod_April2016.pdf`. Among those ROIs, biologically important features are picked, see table 4.9, where red indicated most important, yellow means secondly important, and green means thirdly important. The combinations and all other ROIs are not listed.

The full data set is used for variable selection, and the selection result is chosen such that a 3-fold cross-validation performs best. We run the ENNS algorithm along with the LASSO and the DNP. The selection result are presented in table 4.9. Note here if a model selects a simple combination of some features, these features are also marked as selected. Moreover, table 4.10 shows the number of combined features selected for the models and number of false positive selections. We observe that LASSO misses a lot of important features and selected about only 1/4 of the combined features as neural networks. This indicates that the features may have complicated relationship with the response. ENNS performs better than the shallow DNP in terms of the metrics in table 4.10, where IS is a weighted average score with the weights for red, yellow and green being 3, 2 and 1, respectively; NI is the number of selected important variables; and NU is the number of selected unimportant variables. As a property of the ENNS, it selects less false positive variables. It's hard to track the combined features, since a lot are involved, however, the combinations represent biological intuitions. Neural networks selects more combined features and perform better in this sense.

ROI	R31	R32	R36	R37	R38	R39	R40	R41	R47	R48	R49	R50	R51
Lasso		✓	✓							✓	✓	✓	
DNP	✓	✓				✓	✓		✓	✓	✓		
ENNS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
ROI	R52	R55	R56	R57	R58	R59	R60	R81	R82	R85	R86	R87	R88
Lasso		✓							✓				
DNP		✓						✓	✓	✓	✓	✓	
ENNS			✓					✓	✓	✓	✓	✓	✓
ROI	R100	R101	R102	R103	R106	R107	R116	R117	R118	R119	R120	R121	R122
Lasso		✓				✓	✓					✓	
DNP		✓		✓		✓	✓	✓			✓		
ENNS	✓	✓			✓	✓	✓						
ROI	R123	R124	R125	R132	R133	R136	R137	R138	R139	R140	R141	R142	R143
Lasso	✓	✓			✓	✓	✓	✓		✓		✓	
DNP		✓	✓		✓	✓	✓	✓	✓			✓	
ENNS		✓	✓		✓	✓	✓	✓	✓	✓		✓	
ROI	R146	R147	R152	R153	R154	R155	R162	R163	R164	R165	R166	R167	R168
Lasso					✓	✓	✓		✓	✓		✓	
DNP					✓	✓	✓					✓	
ENNS					✓	✓	✓	✓	✓	✓	✓	✓	
ROI	R169	R170	R171	R178	R179	R186	R187	R190	R191	R194	R195	R198	R199
Lasso		✓	✓	✓	✓		✓	✓					
DNP		✓	✓		✓			✓	✓				
ENNS		✓	✓		✓			✓	✓		✓		
ROI	R200	R201	R202	R203	R204	R205	R206	R207					
Lasso		✓		✓	✓								
DNP	✓		✓	✓		✓							
ENNS	✓	✓		✓		✓							

Table 4.9: Variable selection result for the AD data. The table includes all biologically important variables with three levels: red (very important), yellow (secondly important) and green (thirdly important). The non-important variables are not included in the model. Checkmarks indicate whether the corresponding algorithm selected the variable or not.

Variable selection method	IS	NI	NU
LASSO	1.094	32/86	25/59
DNP	1.428	40/86	15/59
ENNS	1.624	49/86	6/59

Table 4.10: Variable selection result for the AD data. The reported numbers include IS, the weighted average of selected important variables with the weights being 3, 2 and 1 for red (most important), yellow (secondly important) and green (thirdly important), respectively; NI, number of important variables selected; and NU, number of unimportant variables selected.

4.5.2 Regression: Riboflavin Production Data

In this example, we consider the riboflavin production with bacillus subtilis data, which is publicly available in the 'hdi' package in R. The data set contains a continuous response, which is the logarithm of the riboflavin production rate, and $p = 4088$ predictors which are the logarithm of the expression level of 4088 genes. There are $n = 71$ observations available. All predictors are continuous with positive values.

We perform 50 repetitions of the following actions. The data is split into training (56) and testing (15) observations. The training data is further split into training (49) and validation (7). The training data is normalized with mean zero and standard deviation one. We train the ENNS algorithm to select variables and perform the l_1 neural network to make prediction. Along with our proposed algorithms, we compare the performance with the lasso penalized linear regression, which is implemented by the scikit-learn library in python; the group lasso penalized generalized additive model in [145], where the code can be found at `https://github.com/KaixuYang/PenalizedGAM`; and the sparse group lasso penalized neural network in [53]. figure (4.1) shows the average testing mean squared error (MSE) along with the number of selected features for different models. Our proposed algorithm converges fast and performs competitive. table 4.11 shows the average prediction accuracy with standard error in parenthesis and the median number of variables selected. Our proposed method has

Model	Test MSE	Number of features
ENNS+l_1 neural network	0.268(0.115)	42
Regularized neural network	0.273(0.135)	44
Linear model with LASSO	0.286(0.124)	38
Generalized additive model with group lasso	0.282(0.136)	46

Table 4.11: Test MSE with standard error in parentheses and median of number of features for different models in the riboflavin gene data example.

competitive mean performance but lower standard error.

The final model of small sample utilizes only 2 hidden layers, with over 90% sparsity to prevent over-fitting, which is necessary for this small training sample size, 49. Training with large batch size, small learning rate, huge number of epochs and early stopping help the model learn better and prevent over-fitting. We admit that tuning the network structure and learning parameters are hard, but we obtain better and stabler results once we have the right numbers.

4.5.3 Classification: Prostate Cancer Data

In this example, we considered a prostate cancer gene expression data, which is publicly available in `http://featureselection.asu.edu/datasets.php`. The data set contains a binary response with 102 observations on 5966 predictor variables. Clearly, the data set is really a high dimensional data set. The responses have values 1 (50 sample points) and 2 (52 sample points), where 1 indicates normal and 2 indicates tumor. All predictors are continuous predictors, with positive values.

We perform 50 repetitions of the following actions. The data is split into training (81) and testing (21) observations. The training data is further split into training (70) and validation data (11). In each split, the number of class 0 observations and number of class 1 observations

Classifier	Test accuracy	Number of features
ENNS+l_1 neural network	0.956(0.053)	15
Regularized neural network	0.955(0.066)	18
Logistic Regression with Lasso	0.933(0.058)	36
l1 penalized Linear SVM	0.950(0.052)	16
Generalized additive model with group lasso	0.918(0.061)	5

Table 4.12: Test accuracy with standard error in parentheses and median of number of features for different classifiers in the Prostate gene data example.

are kept roughly the same. We train the ENNS algorithm to select variables and perform the l_1 neural network to make predictions. Along with our proposed algorithms, we compare the performance with the l_1 norm penalized logistic regression; the l_1 support vector machine (SVM), both of which are implemented with the scikit-learn library in python; the group lasso penalized generalized additive model in [145], where the code can be found at `https://github.com/KaixuYang/PenalizedGAM`; and the sparse group lasso penalized neural network in [53]. figure (4.2) shows the average testing accuracy over the 20 repetitions along with the number of selected features for different models. Our proposed algorithm converges fast and performs competitive. table 4.12 shows the average prediction accuracy with standard error in parenthesis, and the median number of variables selected. Our proposed methods has competitive mean performance but lower standard error. One needs to notice that the mean performance is hard to improve further, since the results are already good and reach the bottleneck of the current explanatory variables. The reason that GAM performs worse than the other models is that the range of predictor variables are relatively small and skewed, thus the basis expansion on GAM does not work well.

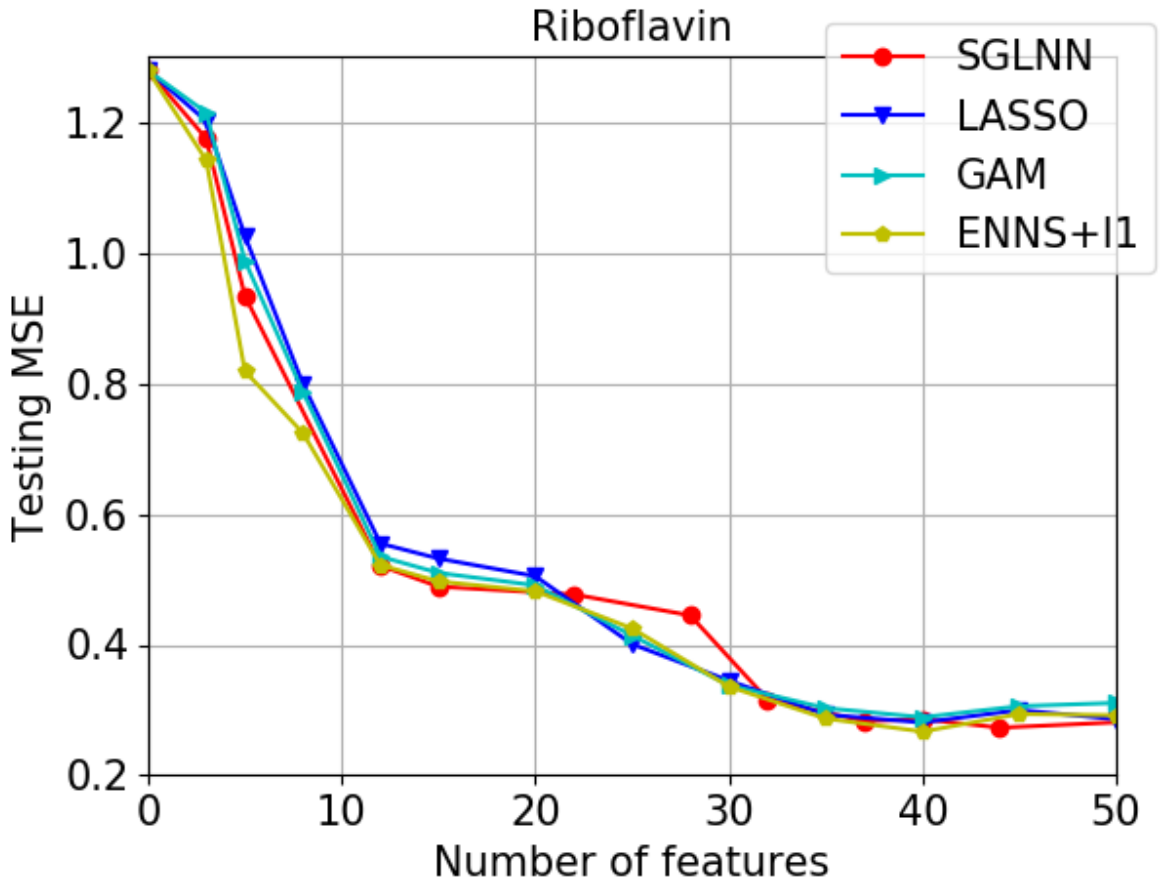

Figure 4.1: Testing mean squared error (MSE) for different models on the riboflavin data.

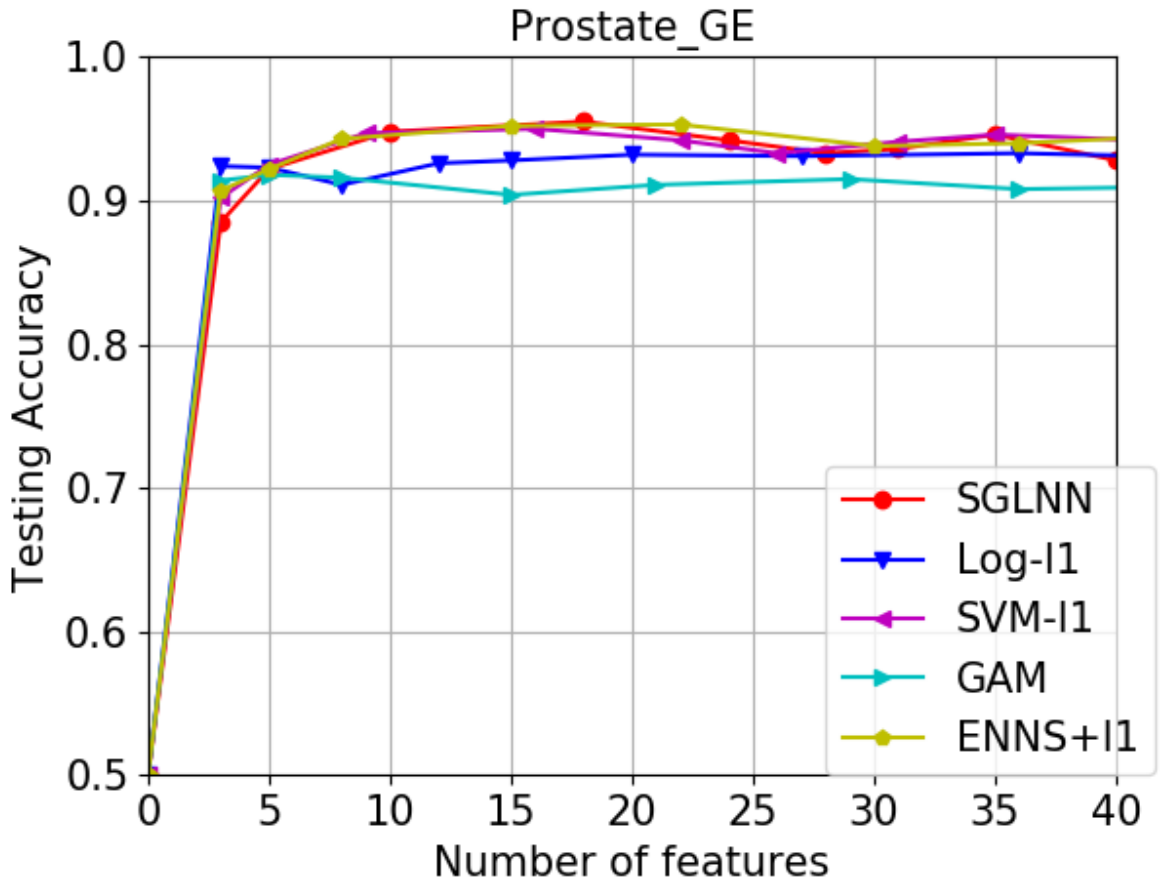

Figure 4.2: Testing accuracy for different models on the prostate cancer data.

4.6 Conclusion

In this paper, we discussed the existing methods to deal with high-dimensional data and how to apply the stage-wise algorithm with neural networks. We discussed the shortage of the current stage-wise neural network variable selection algorithms and proposed the ENNS to overcome these shortage. Moreover, we also compared different methods to further reduce the over-fitting issue after the variable selection step. Methodology was given to support the argument and new algorithm, and simulation studies were given as extra evidence for the algorithm to work.

Though there's a few simulation and methodology study in neural network variable selection, the theory for neural network still deserves much more investigation. We hope this paper could work as a pioneer and attracts more people's attention to the neural network theory field.

Chapter 5

Epilogue

The main goal of this dissertation is to establish theory and propose new methodologies for non-parametric statistical machine learning modeling for the high-dimensional low sample size (HDLSS) problems. We have studied three different non-parametric approaches: the generalized additive model (GAM) in Chapter 2, the sparse group lasso penalized shallow neural network in Chapter 3 and the ensemble neural network selection (ENNS) algorithm in Chapter 4. These three models are appropriate in different situations. The GAM is useful when there is not too much interactions among the variables or there is only weak interactions among the variables, which has many applications. Actually, in most fields, the additive structure is enough to obtain a fairly good result. The shallow neural network is used when there are interactions among the variables, but the relationship between the response and the variables is not too complicated. If we have huge interactions and very complicated relationships, a deep neural network makes better approximation to this relationship. However, on the other hand, the level of difficulty in training the three models also increase gradually, especially it's easy to get stuck in a local minimum when training the neural network. From this aspect, GAM is a convex optimization problem and is guaranteed to reach a global minimum, thus has smaller training difficulty.

The theory we proved also make great contribution towards this field. These theoretical results guarantee that these methods will work under certain conditions, which is better sup-

port than the numerical results only. Moreover, we have applied these methods to different fields of examples, including genetic data, computer vision data, autonomous driving data and MRI data. These non-parametric models are proved to work in different aspects.

In the future, the neural network still have a huge room to investigate. How do we find a more stable way to train a neural network? How do we prevent neural network from overfitting small sample? How do we eliminate the gap between the global minimum used in neural network theory and the local minimum obtain in practice? These are left for future research.

APPENDICES

APPENDIX A

Technical Details and Supplementary Materials for Chapter 2

Derivation of Assumption 2.2

Though assumption 2.2 is imposed on the fixed design matrix, however, it holds if the design matrix $\boldsymbol{X}$ is drawn from a continuous density and the density g_j of X_j is bounded away from 0 and infinity by b and B, respectively, on the interval $[a, b]$. Let $\boldsymbol{\delta}_A$ be the sub-vector of $\boldsymbol{\delta}$ which include all nonzero entries. Without loss of generality, let $\boldsymbol{\delta}_A = \{\boldsymbol{\delta}_1, ..., \boldsymbol{\delta}_k\}$, where $\boldsymbol{\delta}_k \in \mathbb{R}^{mn}$ and $k = O(s_n)$. Let Φ_A be the corresponding sub-matrix of Φ.

By lemma 3 in [120], if the design matrix $\boldsymbol{X}$ is drawn from a continuous density and the density g_j of X_j is bounded away from 0 and infinity by b and B, respectively, on the interval $[a, b]$, and $\text{card}_B(\boldsymbol{\delta}) = O(s_n)$, we have

$$\|\Phi_1\boldsymbol{\delta}_1 + ... + \Phi_k\boldsymbol{\delta}_k\|_2 \geq \gamma_2^{k-1}(\|\Phi_1\boldsymbol{\delta}_1\|_2 + ... + \|\Phi_k\boldsymbol{\delta}_k\|_2)$$

for some positive constant γ_2 such that $\delta_0 < 1 - 2\gamma_2^2 < 1$, where $\delta_0 = ((1 - bB^{-1})/2)$. Together with the triangle inequality, we have

$$\gamma_2^{k-1}(\|\Phi_1\boldsymbol{\delta}_1\|_2 + ... + \|\Phi_k\boldsymbol{\delta}_k\|_2) \leq \|\Phi_A\boldsymbol{\delta}_A\|_2 \leq \|\Phi_1\boldsymbol{\delta}_1\|_2 + ... + \|\Phi_k\boldsymbol{\delta}_k\|_2$$

By simple algebra, we have

$$\gamma_2^{2k-2}(\|\Phi_1\boldsymbol{\delta}_1\|_2^2 + ... + \|\Phi_k\boldsymbol{\delta}_k\|_2^2) \leq \|\Phi_A\boldsymbol{\delta}_A\|_2^2 \leq 2(\|\Phi_1\boldsymbol{\delta}_1\|_2^2 + ... + \|\Phi_k\boldsymbol{\delta}_k\|_2^2)$$

For any $j = 1, ..., k$, by lemma 6.2 in [153], we have

$$c_1 m_n^{-1} \leq \lambda_{min}(n^{-1}\Phi_j^T\Phi_j) \leq \lambda_{max}(n^{-1}\Phi_j^T\Phi_j) \leq c_2 m_n^{-1}$$

for some c_1 and c_2. Then we have

$$\begin{aligned}
\frac{\boldsymbol{\delta}^T\Phi^T\Phi\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|_2^2} &= \frac{\|\Phi_A\boldsymbol{\delta}_A\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2} \\
&\geq \frac{\gamma_2^{2k-2}(\|\Phi_1\boldsymbol{\delta}_1\|_2^2 + ... + \|\Phi_k\boldsymbol{\delta}_k\|_2^2)}{\|\boldsymbol{\delta}_A\|_2^2} \\
&= \gamma_2^{2k-2}\left(\frac{\|\Phi_1\boldsymbol{\delta}_1\|_2^2}{\|\boldsymbol{\delta}_1\|_2^2}\frac{\|\boldsymbol{\delta}_1\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2} + ... + \frac{\|\Phi_k\boldsymbol{\delta}_k\|_2^2}{\|\boldsymbol{\delta}_k\|_2^2}\frac{\|\boldsymbol{\delta}_k\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2}\right) \\
&\geq \gamma_2^{2k-2}c_1 n m_n^{-1}\left(\frac{\|\boldsymbol{\delta}_1\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2} + ... + \frac{\|\boldsymbol{\delta}_k\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2}\right) \\
&= \gamma_2^{2k-2}c_1 n m_n^{-1}
\end{aligned}$$

Let $\gamma_0 = \gamma_2^{-2}c_1$ and observe that $k = O(s_n)$, we have

$$\frac{\boldsymbol{\delta}^T\Phi^T\Phi\boldsymbol{\delta}}{n\|\boldsymbol{\delta}\|_2^2} \geq \gamma_0\gamma_2^{2s_n}m_n^{-1}$$

Similarly, we have

$$\begin{aligned}
\frac{\boldsymbol{\delta}^T\Phi^T\Phi\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|_2^2} &= \frac{\|\Phi_A\boldsymbol{\delta}_A\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2} \\
&\leq \frac{2(\|\Phi_1\boldsymbol{\delta}_1\|_2^2 + ... + \|\Phi_k\boldsymbol{\delta}_k\|_2^2)}{\|\boldsymbol{\delta}_A\|_2^2} \\
&= 2\left(\frac{\|\Phi_1\boldsymbol{\delta}_1\|_2^2}{\|\boldsymbol{\delta}_1\|_2^2}\frac{\|\boldsymbol{\delta}_1\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2} + ... + \frac{\|\Phi_k\boldsymbol{\delta}_k\|_2^2}{\|\boldsymbol{\delta}_k\|_2^2}\frac{\|\boldsymbol{\delta}_k\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2}\right) \\
&\leq 2c_2nm_n^{-1}\left(\frac{\|\boldsymbol{\delta}_1\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2} + ... + \frac{\|\boldsymbol{\delta}_k\|_2^2}{\|\boldsymbol{\delta}_A\|_2^2}\right) \\
&= 2c_2nm_n^{-1}
\end{aligned}$$

Let $\gamma_1 = c_2$, we have

$$\frac{\boldsymbol{\delta}^T\Phi^T\Phi\boldsymbol{\delta}}{n\|\boldsymbol{\delta}\|_2^2} \leq \gamma_1 m_n^{-1}$$

Proofs of Lemma and Theorems

The following lemmas are needed in proving theorems.

Lemma A.1. For any sequence $r_n > 0$, under assumption 2.1 and 3, we have for bounded response such that $|y_i| < c/2$ that

$$\mathbb{P}\left(\left\|\frac{\Phi^T\left(\boldsymbol{y} - \boldsymbol{\mu}_y\right)}{n}\right\|_\infty \leq r_n\right) \geq 1 - 2p_nm_n\exp(-\frac{nr_n^2}{2c^2c_\Phi^2}) \tag{A.1}$$

Specifically, for a diverging sequence t_n, taking

$$r_n = \sqrt{2}cc_\Phi\sqrt{\frac{\log(p_nm_n) + t_n}{n}}$$

we have for response such that $|y_i| < c/2$ that

$$\mathbb{P}\left(\left\|\frac{\Phi^T\left(\boldsymbol{y}-\boldsymbol{\mu}_y\right)}{n}\right\|_\infty \leq r_n\right) \geq 1-2\exp(-t_n) \tag{A.2}$$

Proof. Observe that

$$\frac{\Phi_j^T\left(\boldsymbol{y}-\boldsymbol{\mu}_y\right)}{n} = \sum_{i=1}^n\left(\frac{\phi_{ij}(y_i-\mu_{y_i})}{n}\right) := \sum_{i=1}^n \gamma(y_i)$$

It's easy to verify that $E\gamma(y_i) = 0$ for $i = 1, ..., n$ and $|\gamma(y_i)| = \left|\phi_{ij}(y_i - \mu_{y_i})/n\right| \leq cd_i$ for $i = 1, ..., n$. By assumption 2.1, we have $\sum_{i=1}^n d_i^2 \leq c_\Phi^2/n$ for $i = 1, ..., n$. Apply Bonferroni's inequality and Hoeffding's inequality, we have

$$\begin{aligned}
\mathbb{P}\left(\left\|\frac{\Phi^T\left(\boldsymbol{y}-\boldsymbol{\mu}_y\right)}{n}\right\|_\infty \leq r_n\right) &= 1-\mathbb{P}\left(\left\|\frac{\Phi^T\left(\boldsymbol{y}-\boldsymbol{\mu}_y\right)}{n}\right\|_\infty \geq r_n\right) \\
&= 1-\mathbb{P}\left(\bigcup_{j=1}^{m_n\times p_n}\left\{\left|\frac{\Phi_j^T\left(\boldsymbol{y}-\boldsymbol{\mu}_y\right)}{n}\right| \geq r_n\right\}\right) \\
&\geq 1-\sum_{j=1}^{m_n\times p_n}\mathbb{P}\left(\left|\frac{\Phi_j^T\left(\boldsymbol{y}-\boldsymbol{\mu}_y\right)}{n}\right| \geq r_n\right) \\
&\geq 1-m_n\times p_n\times 2\exp\left(-\frac{nr_n^2}{2u_n^2c_\Phi^2}\right) - c_2n^{1-c_3c_4^2}
\end{aligned}$$

with our choice of

$$r_n = \sqrt{2}cc_\Phi\sqrt{\frac{\log(p_nm_n)+t_n}{n}}$$

we have

$$\begin{aligned}\mathbb{P}\left(\left\|\frac{\Phi^T\left(\boldsymbol{y}-\boldsymbol{\mu}_y\right)}{n}\right\|_\infty \leq r_n\right) &\geq 1 - m_n \times p_n \times 2\exp\left(-\frac{nr_n^2}{8c^2c_\Phi^2}\right) \\ &= 1 - m_n \times p_n \times 2\exp\left(-\frac{n2c^2c_\Phi^2(\log(p_nm_n)+t_n)}{2c^2c_\Phi^2 n}\right) \\ &= 1 - 2\exp(-t_n)\end{aligned}$$

□

Lemma A.2. In the unbounded response case, under assumptions 2.1 and 2.3, let $T_n = n^{-1}\|\Phi_j^T(\boldsymbol{y}-\boldsymbol{\mu}_y)\|_\infty = \max_{j=1,\ldots,p_nm_n} n^{-1}|\Phi_j^T(\boldsymbol{y}-\boldsymbol{\mu}_y)|$, we have

$$ET_n = O(1)n^{-1/2}\sqrt{p_nm_n} \tag{A.3}$$

and then for any diverging sequence a_n,

$$\mathbb{P}\left(T_n \geq a_n\sqrt{\frac{\log(p_nm_n)}{n}}\right) \to 0 \text{ as } n \to 0 \tag{A.4}$$

Proof. By the maximal inequality for sub-Gaussian random variables, for example, see Lemmas 2.2.1 and 2.2.2 in [136] and application see lemma 2 of [65], we have

$$ET_n \leq Cn^{-1}\sqrt{\log(p_nm_n)}\max_j\|\Phi_j\|_2$$

Then by assumption 2.1, we have

$$ET_n = O(1)n^{-1/2}\sqrt{p_nm_n}$$

Since $T_n \geq 0$, by Markov's inequality, we have

$$\mathbb{P}\left(T_n \geq a_n\sqrt{\frac{\log(p_n m_n)}{n}}\right) \leq \frac{ET_n}{n^{-1/2}\sqrt{\log(p_n m_n)}} = \frac{C}{a_n} \to 0 \text{ as } n \to \infty \tag{A.5}$$

□

Remark A.1. From the two lemmas we see that the difference between the bounded response case and the unbounded response case is the upper bound for the maximum of the random errors. For the bounded case, the error could be bounded by

$$r_n = C\sqrt{\frac{\log(p_n m_n) + t_n}{n}}$$

with any diverging sequence t_n. If we take $t_n = O(\log(p_n m_n))$, we have for a different C, the bounded response errors to be bounded by

$$r_n = C\sqrt{\frac{\log(p_n m_n)}{n}}$$

with probability converging to 1. For the unbounded response case, with probability converging to 1, we need a diverging sequence a_n instead of a constant multiplied to the main term, i.e.,

$$r_n = a_n\sqrt{\frac{\log(p_n m_n)}{n}}$$

This difference is reflected on the choice of the tuning parameter λ.

Proof of Theorem 2.1

Proof. First observe that due to the spline approximation, an error is bought into the model. Let $\theta = \sum_{j=1}^{p_n} f_j$ and $\theta^* = \sum_{j=1}^{p_n} f_{nj}$. By the proof of theorem 1 in [65], we have

$$\|f_j - f_{nj}\|_\infty = O(m_n^{-d})$$

Therefore, we have

$$|\theta - \theta^*| \leq \|\sum_{j=1}^{p_n}(f_j - f_{nj})\|_\infty \leq \sum_{j=1}^{s_n}\|f_j - f_{nj}\|_\infty = O(s_n m_n^{-d})$$

Use Taylor expansion on $b'(\theta)$ around θ^*, we have

$$b'(\theta) - b'(\theta^*) = b''(\theta^{**})(\theta - \theta^*)$$

where θ^{**} lies between θ and θ^*. By assumption 2.3, we have

$$|\mu_{y_i} - \mu_{y_i}^*| = |b'(\theta) - b'(\theta^*)| \leq c_1^{-1}|\theta - \theta^*| = O(s_n m_n^{-d}),\ i = 1, ..., n \tag{A.6}$$

where $\mu_{y_i}^*$ is the mean of the i^{th} observation evaluated at the spline approximated functions. Therefore, we have

$$\|\boldsymbol{\mu}_y - \boldsymbol{\mu}_y^*\|_\infty = O(s_n m_n^{-d})$$

As a direct result, we have

$$\frac{1}{n}\|\boldsymbol{\mu}_y - \boldsymbol{\mu}_y^*\|_2^2 = O(s_n^2 m_n^{-2d}) \tag{A.7}$$

We start with part (i). The proof of this part is similar to the proof of part (i) of theorem 1 in [65]. But because of the non-identity link function, here we have to make some changes. By KKT conditions, a necessary and sufficient condition for $\hat{\boldsymbol{\beta}}$ to be a minimiser of the target function is

$$\begin{cases} \frac{1}{n}\Phi_k^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{\boldsymbol{y}}^*) = \frac{\lambda_{n1}\hat{\boldsymbol{\beta}}_k}{\|\hat{\boldsymbol{\beta}}_k\|_2},\ \forall\ k\ s.t.\ \|\hat{\boldsymbol{\beta}}_k\|_2 > 0 \\ \frac{1}{n}\Phi_k^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{\boldsymbol{y}}^*) \in [-\lambda_{n1},\lambda_{n1}],\ \forall\ k\ s.t.\ \|\hat{\boldsymbol{\beta}}_k\|_2 = 0 \end{cases} \tag{A.8}$$

where $\hat{\boldsymbol{\mu}}_{\boldsymbol{y}}^*$ is the mean of response approximated by splines and evaluated at the solution $\hat{\boldsymbol{\beta}}$ and the second belonging relationship is element-wise. Let

$$s_k = \frac{\Phi_k^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{\boldsymbol{y}}^*)}{n\lambda_{n1}}$$

Then we have

$$\begin{cases} \|s_k\|_2 = 1,\ \forall\ k\ s.t.\ \|\hat{\boldsymbol{\beta}}_k\|_2 > 0 \\ \|s_k\|_2 \le 1,\ \forall\ k\ s.t.\ \|\hat{\boldsymbol{\beta}}_k\|_2 = 0 \end{cases} \tag{A.9}$$

We consider the following subsets of $\{1,...,p\}$. Let A_1 be such that

$$\left\{k : \|\hat{\boldsymbol{\beta}}_k\|_2 > 0\right\} \subset A_1 \subset \left\{k : \frac{1}{n}\Phi_k^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{\boldsymbol{y}}^*) = \frac{\lambda_{n1}\hat{\boldsymbol{\beta}}_k}{\|\hat{\boldsymbol{\beta}}_k\|_2}\right\} \cup \{1,...,s_n\} \tag{A.10}$$

Let $A_2 = \{1,...,p\}\backslash A_1$, $A_3 = A_1\backslash T$, $A_4 = A_1 \cap T^c$, $A_5 = A_2\backslash T^c$ and $A_6 = A_2 \cap T^c$. Therefore, the relationships are

	$j \in T$	$j \in T^c$
A_1: selected j and some $j \in T$	A_3	A_4
A_2: j not in A_1 (includes unselected only)	A_5	A_6

Then we have

$$\Phi_{A_1}^T(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{A_1}^*) = S_{A_1} \tag{A.11}$$

where $S_{A_1} = (S_{K_1}^T, ..., S_{K_{q_1}}^T)^T$, $S_{K_i} = n\lambda_{n1} s_{k_i}$ and $\hat{\boldsymbol{\mu}}_{A_1}^* = b'(\Phi_{A_1}\hat{\boldsymbol{\beta}}_{A_1})$. Also from the inequality in KKT, we have

$$-C_{A_2} \leq \Phi_{A_2}^T(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{A_1}^*) \leq C_{A_2} \tag{A.12}$$

where $C_{A_2} = (C_{K_1}^T, ..., C_{k_{q_2}}^T)^T$ and $C_{K_i} = n\lambda_{n1}\mathbb{1}_{\{\|\hat{\boldsymbol{\beta}}_{K_i}\|_2=0\}} \cdot e_{m_n \times 1}$, where all the elements of e are 1. Let $\boldsymbol{\varepsilon}^* = \boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{y}}^*$, then from (A.11) we have

$$\Phi_{A_1}^T(\boldsymbol{\mu}_{\boldsymbol{y}}^* + \boldsymbol{\varepsilon}^* - \hat{\boldsymbol{\mu}}_{A_1}^*) = S_{A_1}$$

use Taylor expansion on $\boldsymbol{\mu}_{\boldsymbol{y}}^*$ around $\hat{\boldsymbol{\mu}}_{A_1}^*$, we have

$$\Phi_{A_1}^T \boldsymbol{\Sigma}_1 \Phi_{A_1}(\boldsymbol{\beta}_{A_1} - \hat{\boldsymbol{\beta}}_{A_1}) + \Phi_{A_1}^T \boldsymbol{\Sigma}_1 \Phi_{A_2} \boldsymbol{\beta}_{A_2} + \Phi_{A_1}^T \boldsymbol{\varepsilon}^* = S_{A_1}$$

where $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}(\boldsymbol{\theta}_1)$, $\boldsymbol{\theta}_1$ lies on the line segment joining $\Phi\boldsymbol{\beta}$ and $\Phi_{A_1}\hat{\boldsymbol{\beta}}_{A_1}$, and $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \text{diag}(b''(\theta_1), ..., b''(\theta_n))$ is the diagonal variance matrix evaluated at $\boldsymbol{\theta}$. From (A.12), we have

$$-C_{A_2} \leq \Phi_{A_2}^T \boldsymbol{\Sigma}_1 \Phi_{A_1}(\boldsymbol{\beta}_{A_1} - \hat{\boldsymbol{\beta}}_{A_1}) + \Phi_{A_2}^T \boldsymbol{\Sigma}_1 \Phi_{A_2} \boldsymbol{\beta}_{A_2} + \Phi_{A_2}^T \boldsymbol{\varepsilon}^* \leq C_{A_2}$$

Let $\boldsymbol{\Sigma}_{ij} = \Phi_{A_i}^T \boldsymbol{\Sigma}(\boldsymbol{\theta}_1)\Phi_{A_j}/n$, we have

$$\boldsymbol{\Sigma}_{11}(\boldsymbol{\beta}_{A_1} - \hat{\boldsymbol{\beta}}_{A_1}) + \boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_{A_2} = S_{A_1}$$

and

$$-C_{A_2} \leq \boldsymbol{\Sigma}_{21}(\boldsymbol{\beta}_{A_1} - \hat{\boldsymbol{\beta}}_{A_1}) + \boldsymbol{\Sigma}_{22}\boldsymbol{\beta}_{A_2} + \Phi_{A_2}^T \boldsymbol{\varepsilon}^* \leq C_{A_2}$$

With our choice of λ_{n1}, the constants are sufficient large, by lemma 1 in [140], the eigenvalues of $\boldsymbol{\Sigma}_{11}$ are bounded from below. Thus without loss of generality, we assume $\boldsymbol{\Sigma}_{11}$ is invertible. Then we have

$$\frac{\boldsymbol{\Sigma}_{11}^{-1} S_{A_1}}{n} = \boldsymbol{\beta}_{A_1} - \hat{\boldsymbol{\beta}}_{A_1} + \boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_{A_2} + \frac{\boldsymbol{\Sigma}_{11}^{-1}}{n}\Phi_{A_1}^T \boldsymbol{\varepsilon}^* \tag{A.13}$$

and

$$\begin{aligned} &\frac{\|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{\mu}_y - \boldsymbol{\mu}_y^*)\|_2}{n} + n\boldsymbol{\Sigma}_{22}\boldsymbol{\beta}_{A_2} - n\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}11^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_{A_2} \\ \leq & C_{A_2} - \Phi_{A_2}^T \boldsymbol{\varepsilon}^* - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1} S_{A_1} + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\Phi_{A_1}^T \boldsymbol{\varepsilon}^* \end{aligned} \tag{A.14}$$

Define

$$V_{1j} = \frac{1}{\sqrt{n}}\boldsymbol{\Sigma}_{11}^{-1/2} Q_{A_j 1}^T S_{A_j},\ j = 1, 3, 4$$

and

$$w_k = \boldsymbol{\Sigma}_1^{1/2}(\boldsymbol{I} - \boldsymbol{P}_1)\boldsymbol{\Sigma}_1^{1/2}\Phi_{A_k}\boldsymbol{\beta}_{A_k}, k = 2, ..., 6$$

where

$$\boldsymbol{P}_1 = \boldsymbol{\Sigma}^{1/2}\Phi_{A_1}(\Phi_{A_1}^T\boldsymbol{\Sigma}\Phi_{A_1})^{-1}\Phi_{A_1}^T\boldsymbol{\Sigma}^{1/2}$$

and $Q_{A_j k}$ is the matrix representing the selection of variables in A_k from A_j.

Consider $j = 4$. For any $k \in A_4$, we have $\|\hat{\boldsymbol{\beta}}_k\|_2 > 0$, then $\|s_k\|_2^2 = 1$. Then we have

$\|S_{A_4}\|_2^2 = \sum_{k\in A_4} N(A_4)$, where $N(A_4)$ is the number of predictors in A_4. Thus

$$\begin{aligned}\|V_{14}\|_2^2 &= \frac{1}{n}\|\boldsymbol{\Sigma}_{11}^{-1/2} Q_{A_4 1}^T S_{A_4}\|_2^2 \\ &\geq \frac{1}{n} c_1 \|Q_{A_4 1}^T S_{A_4}\|_2^2 \\ &= c_1 n \sum_{k\in A_4} \|\lambda_{n1} s_k\|_2^2 \\ &\geq c_1 n \lambda_{n1}^2 (q_1 - s_n)\end{aligned}$$

That is

$$(q_1 - s_n)^+ \leq \frac{\|V_{14}\|_2^2}{c_1 n \lambda_{n1}^2} \tag{A.15}$$

Then, we need to find a bound for $\|V_{14}\|_2^2$ and $q_1 \leq (q_1 - s_n)^+ + s_n$ will be bounded. Using (A.13) and consider

$$\begin{aligned}V_{14}^T(V_{14} + V_{13}) &= S_{A_4}^T Q_{A_4 1} \frac{\boldsymbol{\Sigma}_{11}^{-1}}{n} S_{A_1} \\ &= S_{A_4}^T Q_{A_4 1}(\boldsymbol{\beta}_{A_1} - \hat{\boldsymbol{\beta}}_{A_1} + \boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_{A_2} + \frac{\boldsymbol{\Sigma}_{11}^{-1}}{n}\Phi_{A_1}^T \boldsymbol{\varepsilon}^*) \\ &= S_{A_4}^T Q_{A_4 1}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_{A_2} + \frac{S_{A_4}^T Q_{A_4 1}\boldsymbol{\Sigma}_{11}^{-1}}{n}\Phi_{A_1}^T \boldsymbol{\varepsilon}^* + S_{A_4}^T(\boldsymbol{\beta}_{A_4} - \hat{\boldsymbol{\beta}}_{A_4})\end{aligned}$$

Observe $\boldsymbol{\beta}_{A_4} = 0$, and

$$S_{A_4}^T \hat{\boldsymbol{\beta}}_{A_4} = \sum_{k\in A_4} \frac{\lambda_{n1}\hat{\boldsymbol{\beta}}_k^T \hat{\boldsymbol{\beta}}_k}{\|\hat{\boldsymbol{\beta}}_k\|_2} = \sum_{k\in A_4} \lambda_{n1}\|\hat{\boldsymbol{\beta}}_k\|_2 > 0$$

we have

$$V_{14}^T(V_{14} + V_{13}) \leq S_{A_4}^T Q_{A_4 1}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_{A_2} + \frac{S_{A_4}^T Q_{A_4 1}\boldsymbol{\Sigma}_{11}^{-1}}{n}\Phi_{A_1}^T \boldsymbol{\varepsilon}^*$$

On the other hand, by (A.14),

$$\begin{aligned}
\|w_2\|_2^2 &= \boldsymbol{\beta}_{A_2}^T \Phi_{A_2}^T \boldsymbol{\Sigma}_1^{1/2} (\boldsymbol{I} - \boldsymbol{P}_1) \boldsymbol{\Sigma}_1 (\boldsymbol{I} - \boldsymbol{P}_1) \boldsymbol{\Sigma}_1^{1/2} \Phi_{A_2} \boldsymbol{\beta}_{A_2} \\
&\leq c_1^{-1} \boldsymbol{\beta}_{A_2}^T \Phi_{A_2}^T \boldsymbol{\Sigma}_1^{1/2} (\boldsymbol{I} - \boldsymbol{P}_1) \boldsymbol{\Sigma}_1^{1/2} \Phi_{A_2} \boldsymbol{\beta}_{A_2} \\
&= c_1^{-1} \boldsymbol{\beta}_{A_2}^T \Phi_{A_2}^T \boldsymbol{\Sigma}_1 \Phi_{A_2} \boldsymbol{\beta}_{A_2} + c_1^{-1} \boldsymbol{\beta}_{A_2}^T \Phi_{A_2}^T \boldsymbol{\Sigma}_1 \Phi_{A_1} (\Phi_{A_1}^T \boldsymbol{\Sigma}_1 \Phi_{A_1})^{-1} \Phi_{A_1}^T \boldsymbol{\Sigma}_1 \Phi_{A_2} \boldsymbol{\beta}_{A_2} \\
&= c_1^{-1} \boldsymbol{\beta}_{A_2}^T (n \boldsymbol{\Sigma}_{22} \boldsymbol{\beta}_{A_2} - n \boldsymbol{\Sigma}_{21} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{12} \boldsymbol{\beta}_{A_2}) \\
&\leq c_1^{-1} \boldsymbol{\beta}_{A_2}^T (C_{A_2} - \Phi_{A_2}^T \boldsymbol{\varepsilon}^* - \boldsymbol{\Sigma}_{21} \boldsymbol{\Sigma}_{11}^{-1} S_{A_1} + \boldsymbol{\Sigma}_{21} \boldsymbol{\Sigma}_{11}^{-1} \Phi_{A_1}^T \boldsymbol{\varepsilon}^*) \\
&= c_1^{-1} \boldsymbol{\beta}_{A_2}^T C_{A_2} - c_1^{-1} \boldsymbol{\beta}_{A_2}^T (\Phi_{A_2}^T - \boldsymbol{\Sigma}_{21} \boldsymbol{\Sigma}_{11}^{-1} \Phi_{A_1}^T) \boldsymbol{\varepsilon}^* - c_1^{-1} \boldsymbol{\beta}_{A_2}^T \boldsymbol{\Sigma}_{21} \boldsymbol{\Sigma}_{11}^{-1} S_{A_1} \\
&= c_1^{-1} \boldsymbol{\beta}_{A_2}^T C_{A_2} - c_1^{-1} \boldsymbol{\beta}_{A_2}^T \boldsymbol{\Sigma}_{21} \boldsymbol{\Sigma}_{11}^{-1} S_{A_1} - c_1^{-1} w_2^T \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\varepsilon}^*
\end{aligned}$$

Then we have

$$\begin{aligned}
& V_{14}^T (V_{14} + V_{13}) + c_1 \|w_2\|_2^2 \\
\leq & \left(\frac{S_{A_4}^T Q_{A_4 1} \boldsymbol{\Sigma}_{11}^{-1}}{n} \Phi_{A_1}^T - w_2^T \boldsymbol{\Sigma}_1^{-1} \right) \boldsymbol{\varepsilon}^* - S_{A_3}^T Q_{A_3 1} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{12} \boldsymbol{\beta}_{A_2} + \boldsymbol{\beta}_{A_2}^T C_{A_2}
\end{aligned}$$

Define

$$u = \frac{\Phi_{A_1} \boldsymbol{\Sigma}_{11}^{-1} Q_{A_4 1}^T S_{A_4} / n - \boldsymbol{\Sigma}_1^{-1} w_2}{\| \Phi_{A_1} \boldsymbol{\Sigma}_{11}^{-1} Q_{A_4 1}^T S_{A_4} / n - \boldsymbol{\Sigma}_1^{-1} w_2 \|_2}$$

Observe

$$\begin{aligned}
&\|\Phi_{A_1}^T \boldsymbol{\Sigma}_{11}^{-1} Q_{A_4 1}^T S_{A_4}/n - \boldsymbol{\Sigma}_1^{-1} w_2\|_2 \\
\leq & 2(\|\Phi_{A_1}^T \boldsymbol{\Sigma}_{11}^{-1} Q_{A_4 1}^T S_{A_4}/n\|_2^2 + \|\boldsymbol{\Sigma}_1^{-1} w_2\|_2^2) \\
\leq & 2\|\Phi_{A_1}^T \boldsymbol{\Sigma}_{11}^{-1} Q_{A_4 1}^T S_{A_4}/n\|_2^2 + 2c_1^{-2}\|w_2\|_2^2 \\
= & 2\|V_{14}\|_2^2 + 2c_1^{-2}\|w_2\|_2^2
\end{aligned}$$

Observe $c_1 < c_1^{-1}$ implies $c_1 < 1$, then

$$\begin{aligned}
\|V_{14}\|_2^2 + c_1\|w_2\|_2^2 + V_{14}^T V_{13} \leq & (2c_1^{-2}\|V_{14}\|_2^2 + 2c_1^{-2}\|w_2\|_2^2)^{1/2} |u^T \boldsymbol{\varepsilon}^*| \\
& + \sqrt{n}\|V_{13}\|_2 \|\boldsymbol{\Sigma}_{11}^{-1/2} \boldsymbol{\Sigma}_{12} \boldsymbol{\beta}_{A_2}\|_2 + \lambda_{n1}\|\boldsymbol{\beta}_{A_5}\|_1 \qquad \text{(A.16)}
\end{aligned}$$

By (A.15), we have

$$\begin{aligned}
\|V_{13}\|_2^2 &= \frac{1}{n}\|\boldsymbol{\Sigma}_{11}^{-1/2} Q_{A_3 1}^T S_{A_3}\|_2^2 \\
&\leq c_1^{-1} \frac{\|Q_{A_3 1} S_{A_3}\|_2^2}{n} \\
&= c_1^{-1} \sum_{k \in A_3} \|\lambda_{n1} s_k\|_2^2 \\
&\leq c_1^{-1} n \lambda_{n1}^2 N(A_3)
\end{aligned}$$

By (A.16), we have

$$\begin{aligned}
&\|V_{14}\|_2^2 + c_1\|w_2\|_2^2 \\
\leq& c_1^{-1}(2\|V_{14}\|_2^2 + 2\|w_2\|_2^2)^{1/2}|u^T\boldsymbol{\varepsilon}^*| + \sqrt{c_1^{-1}n\lambda_{n1}^2 N(A_3)}\|V_{14}\|_2 \\
&+ \sqrt{c_1^{-1}n\lambda_{n1}^2 N(A_3)}\|\boldsymbol{\Sigma}_{11}^{-1/2}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_{A_2}\|_2 + \lambda_{n1}\|\boldsymbol{\beta}_{A_5}\|_1
\end{aligned} \tag{A.17}$$

Define

$$B_1 = \sqrt{c_1 n\lambda_{n1}^2 s_n} \text{ and } B_2 = \sqrt{c_1^{-1}n\lambda_{n1}^2 s_n}$$

consider the event

$$\mathcal{E} = \left\{ |u^T\boldsymbol{\varepsilon}^*|^2 \leq \frac{(|A_1| \vee m_n)c_1^2 n\lambda_{n1}^2}{4m_n} = (|A_1| \vee m_n)\frac{c_1^3 B_1^2}{4s_n m_n} \right\}$$

later we will show that this event holds with probability tending to 1. On the event $\mathcal{E}$, by (A.15), we have

$$\|V_{14}\|_2^2 \geq \frac{q_1}{s_n}B_1^2 - B_1^2$$

then

$$|u^T\boldsymbol{\varepsilon}^*|^2 \leq \frac{c_1^3 q_1 m_n B_1^2}{4s_n m_n} \leq \frac{c_1^3}{4}(\|V_{14}\|_2^2 + B_1^2)$$

and we have

$$\begin{aligned}
c_1^{-1}(2\|V_{14}\|_2^2 + 2\|w_2\|_2^2)^{1/2}|u^T\boldsymbol{\varepsilon}^*| &\leq c_1^{-3}|u^T\boldsymbol{\varepsilon}^*|^2 + \frac{c_1^3}{4}c_1^{-2}(2\|V_{14}\|_2^2 + 2\|w_2\|_2^2) \\
&\leq \frac{1}{4}(\|V_{14}\|_2^2 + B_1^2) + \frac{c_1^3}{4}c_1^{-2}(2\|V_{14}\|_2^2 + 2\|w_2\|_2^2) \\
&\leq \frac{3}{4}\|V_{14}\|_2^2 + \frac{1}{4}B_1^2 + \frac{c_1}{2}\|w_2\|_2^2
\end{aligned}$$

Then we have

$$\|V_{14}\|_2^2 + c_1\|w_2\|_2^2 \leq \frac{3}{4}\|V_{14}\|_2^2 + \frac{1}{4}B_1^2 + \frac{c_1}{2}\|w_2\|_2^2 + \sqrt{c_1^{-1}n\lambda_{n1}^2 N(A_3)}\|V_{14}\|_2 + \sqrt{c_1^{-1}n\lambda_{n1}^2 N(A_3)}\|\boldsymbol{\Sigma}_{11}^{-1/2}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_{A_2}\|_2 + \lambda_{n1}\|\boldsymbol{\beta}_{A_5}\|_1$$

i.e.

$$\|V_{14}\|_2^2 + 2c_1\|w_2\|_2^2 \leq B_1^2 + 4\sqrt{c_1^{-1}n\lambda_{n1}^2 N(A_3)}(\|V_{14}\|_2 + \|\boldsymbol{\Sigma}_{11}^{-1/2}\boldsymbol{\Sigma}_{12}\boldsymbol{\beta}_{A_2}\|_2) + \lambda_{n1}\|\boldsymbol{\beta}_{A_5}\|_1$$

Consider the set A_1 that contains all $\boldsymbol{\beta}_k \neq 0$. We have $q_1 \geq s_n$ and

$$\left\{k : \|\hat{\boldsymbol{\beta}}_k\|_2 > 0 \text{ or } k \notin T^c\right\} \subset A_1 \subset \left\{k : \frac{1}{n}\Phi_k^T(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{\boldsymbol{y}}^*) = \frac{\lambda_{n1}\hat{\boldsymbol{\beta}}_k}{\|\hat{\boldsymbol{\beta}}_k\|_2} \text{ or } k \notin T^c\right\} \tag{A.18}$$

Then we have $A_5 = \emptyset$, $N(A_3) = s_n \leq q_1$ and $\boldsymbol{\beta}_{A_2} = \mathbf{0}$. Then we have

$$\|V_{14}\|_2^2 \leq B_1^2 + 4\sqrt{c_1^{-1}n\lambda_{n1}^2 s_n}\|V_{14}\|_2 = B_1^2 + 4B_2\|V_{14}\|_2$$

Use the truth that $x^2 \leq c + 2bx$ implies $x^2 \leq 2c + 4b^2$, we have

$$\|V_{14}\|_2^2 \leq 2B_1^2 + 16B_2^2$$

Then we have from (A.15) that

$$(q_1 - s_n)^+ \leq \frac{\|V_{14}\|_2^2}{c_1 n\lambda_{n1}^2} \leq \frac{2B_1^2 + 16B_2^2}{c_1 n\lambda_{n1}^2} = c_5 s_n$$

where $c_5 = (2c_1^2 + 16)/c_1^2$, i.e.

$$(q_1 - s_n)^+ + s_n \leq (c_5 + 1)s_n \tag{A.19}$$

We note that the constant c_5 only depends on c_1 and (A.18) simply requires larger A_1, (A.19) holds for all A_1 satisfying (A.10). Note that (A.19) holds if

$$q_1 \leq N(A_1 \cup A_5) \leq \frac{n}{m_n} \text{ and } |u^T \boldsymbol{\varepsilon}^*|^2 \leq \frac{(|A_1| \vee m_n)c_1^2 n\lambda_{n1}^2}{4m_n} \tag{A.20}$$

So it remains to show that (A.20) holds with probability tending to 1. Define

$$x_m^* = \max_{|A|=m} \max_{\|U_{A_k}\|_2=1, k=1,...,m} \left| \boldsymbol{\varepsilon}^{*T} \frac{\Phi_A(\Phi_A^T \boldsymbol{\Sigma}_A \Phi_A)^{-1} \bar{S}_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - \boldsymbol{\Sigma}_A^{1/2}\Phi_A(\Phi_A^T\boldsymbol{\Sigma}_A\Phi_A)^{-1}\Phi_A^T\boldsymbol{\Sigma}_A^{1/2})\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}}{\|\Phi_A(\Phi_A^T\boldsymbol{\Sigma}_A\Phi_A)^{-1}\bar{S}_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - \boldsymbol{\Sigma}_A^{1/2}\Phi_A(\Phi_A^T\boldsymbol{\Sigma}_A\Phi_A)^{-1}\Phi_A^T\boldsymbol{\Sigma}_A^{1/2})\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}\|_2} \right| \tag{A.21}$$

for $|A| = q_1 = m \geq 0$, $\bar{S}_A = (\bar{S}_{A_1}^T, ..., \bar{S}_{A_m}^T)^T$ where $\bar{S}_{A_k} = \lambda_{n1} U_{A_k}$, $\|U_{A_k}\|_2 = 1$ and $\boldsymbol{\Sigma}_A$ is the variance matrix evaluated at some θ corresponding to the remainder of the Taylor expansion when the subset A is considered. To simplify the notations, let $Q_A = \lambda_{n1}\Phi_A(\Phi_A^T\boldsymbol{\Sigma}_A\Phi_A)^{-1}$ and $P_A = \boldsymbol{\Sigma}_A^{1/2}\Phi_A(\Phi_A^T\boldsymbol{\Sigma}_A\Phi_A)^{-1}\Phi_A^T\boldsymbol{\Sigma}_A^{1/2}$, then we have

$$x_m^* = \max_{|A|=m} \max_{\|U_{A_k}\|_2=1, k=1,...,m} \left| \boldsymbol{\varepsilon}^{*T} \frac{Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}}{\|Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}\|_2} \right| \tag{A.22}$$

Define

$$\Omega^*_{m_0} = \{(U, \boldsymbol{\varepsilon}^*) : x^*_m \le C\sqrt{(|A| \vee 1)m_n \log(p_n m_n)}, \forall m = |A| \ge m_0\}$$

and

$$\Omega_{m_0} = \{(U, \boldsymbol{\varepsilon}) : x^{**}_m \le C\sqrt{(|A| \vee 1)m_n \log(p_n m_n)}, \forall m = |A| \ge m_0\}$$

for a large enough generic constant C, where

$$x^{**}_m = \max_{|A|=m} \max_{\|U_{A_k}\|_2=1, k=1,\ldots,m} \left| \boldsymbol{\varepsilon}^T \frac{Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}}{\|Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}\|_2} \right|$$

By triangle inequality and Cauchy-Schwarz inequality, we have

$$\begin{aligned}
&\left| \boldsymbol{\varepsilon}^{*T} \frac{Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}}{\|Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}\|_2} \right| \\
\le &\left| \boldsymbol{\varepsilon}^{T} \frac{Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}}{\|Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}\|_2} \right| + \|\boldsymbol{\theta}_n\|_2 \\
\le &\left| \boldsymbol{\varepsilon}^{T} \frac{Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}}{\|Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}\|_2} \right| + Cn^{1/2}s_n m_n^{-d} \\
\le &\left| \boldsymbol{\varepsilon}^{T} \frac{Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}}{\|Q_A U_A - \boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I} - P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}\|_2} \right| + C\sqrt{(|A| \vee 1)m_n \log(p_n m_n)}
\end{aligned}$$

Then we have

$$(U, \boldsymbol{\varepsilon}) \in \Omega_{m_0} \;\Rightarrow\; (U, \boldsymbol{\varepsilon}^*) \in \Omega^*_{m_0} \;\Rightarrow\; |u^T \boldsymbol{\varepsilon}^*|^2 \le |x^*_m|^2 \le \frac{(|A_1| \vee m_n)c_1^2 n\lambda^2_{n1}}{4m_n} \text{ for } q_1 \ge m_0 \ge 0$$

Since ϵ_i's are sub-Gaussian random variables by assumption 2.2, we have

$$
\begin{aligned}
&1-\mathbb{P}\left((U,\boldsymbol{\varepsilon})\in\Omega_q\right)\\
=&\mathbb{P}\left(x_m^{**}>C\sqrt{(m\vee 1)m_n\log(p_nm_n)},\forall m=|A|\geq m_0\right)\\
\leq&\sum_{m=0}^{\infty}\mathbb{P}\left(x_m^{**}>C\sqrt{(m\vee 1)m_n\log(p_nm_n)}\right)\\
\leq&\sum_{m=0}^{\infty}\binom{p_n}{m}\mathbb{P}\left(\left|\boldsymbol{\varepsilon}^T\frac{Q_AU_A-\boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I}-P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}}{\|Q_AU_A-\boldsymbol{\Sigma}_A^{-1/2}(\boldsymbol{I}-P_A)\boldsymbol{\Sigma}_A^{1/2}\Phi\boldsymbol{\beta}\|_2}\right|>C\sqrt{(m\vee 1)m_n\log(p_nm_n)}\right)\\
\leq&2\sum_{m=0}^{\infty}\binom{p_n}{m}\exp\left(-C(m\vee 1)m_n\log(p_nm_n)\right)\\
=&2(p_nm_n)^{-Cm_n}+2\sum_{m=1}^{\infty}\binom{p_n}{m}(p_nm_n)^{-Cmm_n}\\
\leq&2(p_nm_n)^{-Cm_n}+2\sum_{m=1}^{\infty}\frac{1}{m!}\left(\frac{p_n}{(p_nm_n)^{Cm_n}}\right)^m\\
=&2(p_nm_n)^{-Cm_n}+2\exp\left(\frac{p_n}{(p_nm_n)^{Cm_n}}\right)-2\to 0\text{ as }n\to\infty
\end{aligned}
$$

Therefore, the proof of part (i) is complete.

Then we prove part (ii). Consider the bounded response case. For a sequence N_n such that $\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_2\leq N_n$, define $t=N_n/(N_n+\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_2)$, then consider the convex combination $\boldsymbol{\beta}^*=t\hat{\boldsymbol{\beta}}+(1-t)\boldsymbol{\beta}^0$. We have $\boldsymbol{\beta}^*-\boldsymbol{\beta}^0=t(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0)$, which implies

$$
\|\boldsymbol{\beta}^*-\boldsymbol{\beta}^0\|_2=t\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_2=\frac{N_n\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_2}{N_n+\|\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0\|_2}\leq N_n \tag{A.23}
$$

Recall the log likelihood function

$$
l_n(\boldsymbol{\beta})=\frac{1}{n}\sum_{i=1}^{n}\left[y_i\left(\alpha+\boldsymbol{\beta}^T\Phi_i\right)-b\left(\alpha+\boldsymbol{\beta}^T\Phi_i\right)\right]
$$

$$
\begin{aligned}
l_n(\boldsymbol{\beta}^*) &= l_n(\boldsymbol{\beta}^0) + \frac{1}{n}\sum_{i=1}^{n}\left[y_i\Phi_i - \mu^*_{y_i}\Phi_i\right]^T(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) \\
&\quad - \frac{1}{2n}\sum_{i=1}^{n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T\Phi_i^T b''(\alpha + \boldsymbol{\beta}^{**T}\Phi_i)\Phi_i(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) \\
&= l_n(\boldsymbol{\beta}^0) + \frac{(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T\Phi^T(\boldsymbol{y} - \boldsymbol{\mu}^*_y)}{n} - \frac{1}{2n}(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T\Phi^T\boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})\Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) \quad \text{(A.24)}
\end{aligned}
$$

where $\boldsymbol{\beta}^{**}$ lines on the line joining $\boldsymbol{\beta}^*$ and $\boldsymbol{\beta}^0$, and

$$
\boldsymbol{\Sigma}(\boldsymbol{\beta}^{**}) = \text{diag}\left(b''(\alpha + \boldsymbol{\beta}^{**T}\Phi_1), ...b''(\alpha + \boldsymbol{\beta}^{**T}\Phi_n)\right)
$$

is the variance matrix of response when the coefficients take value on $\boldsymbol{\beta}^{**}$. On the other hand, by convexity of the log likelihood function,

$$
l_n(\boldsymbol{\beta}^*) = l_n(t\hat{\boldsymbol{\beta}} + (1-t)\boldsymbol{\beta}^0) \geq tl_n(\hat{\boldsymbol{\beta}}) + (1-t)l_n(\boldsymbol{\beta}^0)
$$

by norm inequality, we have

$$
\sum_{j=1}^{p_n}\|\boldsymbol{\beta}^*_j\|_2 = \sum_{j=1}^{p_n}\|t\hat{\boldsymbol{\beta}}_j + (1-t)\boldsymbol{\beta}^0_j\|_2 \leq \sum_{j=1}^{p_n}(t\|\hat{\boldsymbol{\beta}}_j\|_2 + (1-t)\|\boldsymbol{\beta}^0_j\|_2)
$$

joining the two inequalities above and by the definition of $\hat{\boldsymbol{\beta}}$ gives

$$
\begin{aligned}
& l_n(\boldsymbol{\beta}^*) - \lambda_{n1}\sum_{j=1}^{p_n}\|\boldsymbol{\beta}^*_j\|_2 \\
\geq & tl_n(\hat{\boldsymbol{\beta}}) + (1-t)l_n(\boldsymbol{\beta}^0_j) - \lambda_{n1}\sum_{j=1}^{p_n}(t\|\hat{\boldsymbol{\beta}}_j\|_2 + (1-t)\|\boldsymbol{\beta}^0_j\|_2) \\
\geq & l_n(\boldsymbol{\beta}^0) - \lambda_{n1}\sum_{j=1}^{p_n}\|\boldsymbol{\beta}^0_j\|_2
\end{aligned}
$$

which implies

$$l_n(\boldsymbol{\beta}^*) - l_n(\boldsymbol{\beta}^0) \geq \lambda_{n1} \sum_{j=1}^{p_n} \|\boldsymbol{\beta}_j^*\|_2 - \lambda_{n1} \sum_{j=1}^{p_n} \|\boldsymbol{\beta}_j^0\|_2 \tag{A.25}$$

By (A.24) and (A.25) together we have

$$\lambda_{n1} \sum_{j=1}^{p_n} \left(\|\boldsymbol{\beta}_j^*\|_2 - \|\boldsymbol{\beta}_j^0\|_2 \right) \leq \frac{(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T (\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{y}}^*)}{n} - \frac{1}{2n} (\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**}) \Phi (\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)$$

and move one term to the left hand side, we have

$$\begin{aligned}
&\frac{1}{2n} (\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**}) \Phi (\boldsymbol{\beta}^* - \boldsymbol{\beta}^0) \\
\leq & \frac{\left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right)^T \Phi^T \left(\boldsymbol{y} - \boldsymbol{\mu}_y^*\right)}{n} + \lambda_{n1} \sum_{j=1}^{p_n} \left(\|\boldsymbol{\beta}_j^0\|_2 - \|\boldsymbol{\beta}_j^*\|_2 \right) \\
= & \frac{\left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right)^T \Phi^T \left(\boldsymbol{y} - \boldsymbol{\mu}_y\right)}{n} + \frac{\left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right)^T \Phi^T \left(\boldsymbol{\mu}_y^* - \boldsymbol{\mu}_y\right)}{n} + \lambda_{n1} \sum_{j=1}^{p_n} \left(\|\boldsymbol{\beta}_j^0\|_2 - \|\boldsymbol{\beta}_j^*\|_2 \right)
\end{aligned} \tag{A.26}$$

We have for the second term

$$\begin{aligned}
&\frac{\left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right)^T \Phi^T \left(\boldsymbol{\mu}_y^* - \boldsymbol{\mu}_y\right)}{n} \\
= & \frac{\left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{1/2} \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{-1/2} \left(\boldsymbol{\mu}_y^* - \boldsymbol{\mu}_y\right)}{n} \\
\leq & \frac{\|\boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{1/2} \Phi \left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right) \|_2 \|\boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{-1/2} \left(\boldsymbol{\mu}_y^* - \boldsymbol{\mu}_y\right) \|_2}{n} \\
\leq & \frac{(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**}) \Phi (\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)}{4n} + \frac{\|\boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{-1/2} \left(\boldsymbol{\mu}_y^* - \boldsymbol{\mu}_y\right) \|_2^2}{n} \\
\leq & \frac{(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**}) \Phi (\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)}{4n} + c_1 d_n
\end{aligned} \tag{A.27}$$

where $d_n = O(s_n^2 m_n^{-2d})$, the first inequality follows from Cauchy-Schwarz inequality, the

second inequality follows from the identity $uv \leq u^2/4 + v^2$, and the third inequality follow from assumption 2.3 and (A.7). Then joining (A.26) and (A.27), we have

$$\begin{aligned}
&\frac{(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**}) \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)}{4n} \\
\leq &\frac{\left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right)^T \Phi^T \left(\boldsymbol{y} - \boldsymbol{\mu}_y\right)}{n} + \lambda_{n1} \sum_{j=1}^{p_n} \left(\|\boldsymbol{\beta}_j^0\|_2 - \|\boldsymbol{\beta}_j^*\|_2 \right) + c_1 d_n
\end{aligned} \tag{A.28}$$

For the first term on the right hand side of (A.28), we have

$$\begin{aligned}
&\frac{\left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right)^T \Phi^T \left(\boldsymbol{y} - \boldsymbol{\mu}_y\right)}{n} \\
= &\frac{\left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{1/2} \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{-1/2} \left(\boldsymbol{y} - \boldsymbol{\mu}_y\right)}{n} \\
\leq &\frac{(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**}) \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)}{8n} + \frac{2\|\boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{-1/2} \left(\boldsymbol{y} - \boldsymbol{\mu}_y\right)\|_2^2}{n}
\end{aligned} \tag{A.29}$$

where the inequality is by the identity $a^T b \leq \|a\|_2^2/8 + 2\|b\|_2^2$. Joining (A.31) and (A.29), we have

$$\begin{aligned}
&\frac{(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**}) \Phi(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0)}{8n} \\
\leq &\frac{2\|\boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{-1/2} \left(\boldsymbol{y} - \boldsymbol{\mu}_y\right)\|_2^2}{n} + \lambda_{n1} \sum_{j=1}^{p_n} \left(\|\boldsymbol{\beta}_j^0\|_2 - \|\boldsymbol{\beta}_j^*\|_2 \right) + c_1 d_n
\end{aligned} \tag{A.30}$$

By remark 2.1, we have

$$\begin{aligned}
&\frac{\gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1}}{8} \|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2^2 \\
\leq &\frac{2\|\boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{-1/2} \left(\boldsymbol{y} - \boldsymbol{\mu}_y\right)\|_2^2}{n} + \lambda_{n1} \sum_{j=1}^{p_n} \left(\|\boldsymbol{\beta}_j^0\|_2 - \|\boldsymbol{\beta}_j^*\|_2 \right) + c_1 d_n
\end{aligned} \tag{A.31}$$

Observe that

$$\begin{aligned}\|\boldsymbol{\Sigma}(\boldsymbol{\beta}^{**})^{-1/2}\left(\boldsymbol{y}-\boldsymbol{\mu}_y\right)\|_2^2 \leq & c_1^{-1}\|\boldsymbol{y}-\boldsymbol{\mu}_y\|_2^2 \\ \leq & \frac{c_1^{-1}m_n}{\gamma_0\gamma_2^{2s_n}}\|\Phi^T(\boldsymbol{y}-\boldsymbol{\mu}_y)\|_2^2\end{aligned}$$

Then by lemma A.1, we have

$$\begin{aligned}&\frac{\gamma_0 c_1\gamma_2^{2s_n}m_n^{-1}}{8}\|\boldsymbol{\beta}^*_{\{T\cup\hat{T}\}}-\boldsymbol{\beta}^0_{\{T\cup\hat{T}\}}\|_2^2 \\ \leq & O_P\left(s_n m_n\frac{\log(p_n m_n)}{n\gamma_2^{2s_n}}\right)+\lambda_{n1}\sum_{j=1}^{p_n}\left(\|\boldsymbol{\beta}_j^0\|_2-\|\boldsymbol{\beta}_j^*\|_2\right)+O(s_n^2 m_n^{-2d})\end{aligned} \tag{A.32}$$

Observe that

$$\begin{aligned}&\lambda_{n1}\sum_{j=1}^{p_n}\left(\|\boldsymbol{\beta}_j^0\|_2-\|\boldsymbol{\beta}_j^*\|_2\right) \\ \leq & \lambda_{n1}\sum_{j\in T\cup\hat{T}}\left\|\boldsymbol{\beta}_j^0-\boldsymbol{\beta}_j^*\right\|_2 \\ \leq & \lambda_{n1}\sqrt{s_n}\left\|\boldsymbol{\beta}^*_{\{T\cup\hat{T}\}}-\boldsymbol{\beta}^0_{\{T\cup\hat{T}\}}\right\|_2 \\ \leq & \frac{\gamma_0 c_1\gamma_2^{2s_n}m_n^{-1}}{16}\left\|\boldsymbol{\beta}^*_{\{T\cup\hat{T}\}}-\boldsymbol{\beta}^0_{\{T\cup\hat{T}\}}\right\|_2^2+\frac{4\lambda_{n1}^2 s_n}{\gamma_0 c_1\gamma_2^{2s_n}m_n^{-1}}\end{aligned} \tag{A.33}$$

where the first two inequalities are by norm inequality, and the third inequality is by the identity $a^Tb\leq\|a\|_2^2+\|b\|_2^2/4$. Joining (A.32) and (A.33), we have

$$\left\|\boldsymbol{\beta}^*-\boldsymbol{\beta}^0\right\|_2^2=O_P\left(s_n\gamma_2^{-2s_n}\frac{m_n^2\log(p_n m_n)}{n}\right)+O(\lambda_{n1}^2 m_n^2 s_n\gamma_2^{-2s_n})+O(s_n^2 m_n^{1-2d}\gamma_2^{-2s_n}) \tag{A.34}$$

For some N_n such that

$$\|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2 \leq N_n/2$$

By definition of $\boldsymbol{\beta}^*$, we have

$$\|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2 = \frac{N_n}{N_n + \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2}\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2 \leq \frac{N_n}{2}$$

The inequality above implies

$$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2 \leq N_n$$

Therefore,

$$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 = O_P\left(s_n\gamma_2^{-2s_n}\frac{m_n^2\log(p_nm_n)}{n}\right) + O(\lambda_{n1}^2 m_n^2 s_n\gamma_2^{-2s_n}) + O(s_n^2 m_n^{1-2d}\gamma_2^{-2s_n})$$

In the unbounded response case, the only difference that we have to make is in (A.29), we have

$$\begin{aligned}&\frac{\left(\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\right)^T \Phi^T\left(\boldsymbol{y} - \boldsymbol{\mu}_y\right)}{n}\\ \leq&\frac{\gamma_0 c_1}{8}\|\boldsymbol{\beta}^*_{\{T\cup\hat{T}\}} - \boldsymbol{\beta}^0_{\{T\cup\hat{T}\}}\|_2^2 + O_P\left(s_n m_n a_n\frac{\log(p_nm_n)}{n}\right)\end{aligned} \tag{A.35}$$

where the convergence rate is by lemma A.2. Then with the choice of λ_{n1} for this case, we have

$$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 = O_P\left(s_n\gamma_2^{-2s_n}\frac{m_n^2\log(p_nm_n)}{n}\right) + O(\lambda_{n1}^2 m_n^2 s_n\gamma_2^{-2s_n}) + O(s_n^2 m_n^{1-2d}\gamma_2^{-2s_n})$$

Part (iii) is a direct result of part (ii). By assumption 2.4, we have $\|f_j\|_2 \geq c_f > 0$, and we

have

$$\|f_{nj}\|_2 \geq \|f_j\|_2 - \|f_j - f_{nj}\|_2 \geq c_f - O(m_n^{-d}) \geq \frac{1}{2}c_f$$

for large n. By the properties of spline in [35], see for example [121] and [65], there exist positive constants c_1 and c_2 such that

$$c_1 m_n^{-1}\|\boldsymbol{\beta}_j^0\|_2^2 \leq \|f_{nj}\|_2 \leq c_2 m_n^{-1}\|\boldsymbol{\beta}_j^0\|_2$$

Then we have $\|\boldsymbol{\beta}_j^0\|_2^2 \geq c_2^{-1} m_n \|f_{nj}\|_2^2 \geq 0.25 c_2^{-1} m_n c_f^2$. Suppose there is a $j \in T$ such that $\|\hat{\boldsymbol{\beta}}_j\|_2 = 0$, then we have

$$\|\boldsymbol{\beta}_j^0\|_2 \geq 0.25 c_2^{-1} m_n c_f^2$$

which is a contradiction to the result in (ii) and the theorem assumption. Therefore, part (iii) follows. □

Proof of Theorem 2.2

Proof. By the definition of $\hat{f}_{nj},\ j = 1, ..., p$, part (i) and part (iii) follows directly from part (i) and part (iii) in theorem 3.1. It remains to show part (ii). By the properties of spline in [35], see for example [121] and [65], there exist positive constants c_1 and c_2 such that

$$c_1 m_n^{-1}\|\hat{\boldsymbol{\beta}}_{nj} - \boldsymbol{\beta}_{nj}\|_2^2 \leq \|\hat{f}_{nj} - f_{nj}\|_2^2 \leq c_2 m_n^{-1}\|\hat{\boldsymbol{\beta}}_{nj} - \boldsymbol{\beta}_{nj}\|_2^2 \tag{A.36}$$

Therefore, we have

$$\|\hat{f}_{nj} - f_{nj}\|_2^2 = O_P\left(s_n \gamma_2^{-2s_n} \frac{m_n \log(p_n m_n)}{n}\right) + O({\lambda_{n1}^b}^2 m_n s_n \gamma_2^{-2s_n}) + O(s_n^2 m_n^{-2d} \gamma_2^{-2s_n}) \tag{A.37}$$

for the bounded response case and

$$\|\hat{f}_{nj} - f_{nj}\|_2^2 = O_P\left(s_n a_n \gamma_2^{-2s_n} \frac{m_n \log(p_n m_n)}{n}\right) + O({\lambda_{n1}^b}^2 m_n s_n \gamma_2^{-2s_n}) + O(s_n^2 m_n^{-2d} \gamma_2^{-2s_n}) \tag{A.38}$$

for the unbounded response case. This, together with (2.7) and triangle inequality implies part (iii). □

Proof of Theorem 2.3

Proof. We start with part (i). To prove part (i), it's equivalent to prove that the selection is done as it is performed right on the active set, and none of the nonzero components are dropped with probability tending to 1. Let

$$\hat{\boldsymbol{\beta}}_{NZ} = \underset{\boldsymbol{\beta} \in \mathbb{R}^{p_n m_n} : \boldsymbol{\beta}_{T^c} = 0}{\arg\min} L_a(\boldsymbol{\beta}; \lambda_{n2})$$

be the adaptive group lasso estimator restricted to the true nonzero components. First we show that with probability converging to 1, $\hat{\boldsymbol{\beta}}_{NZ}$ is the solution to minimizing (2.16), i.e., with probability converging to 1, the minimizer of 2.16 is $\hat{\boldsymbol{\beta}}_{NZ}$. Note that the adaptive group lasso is a convex optimization problem with affine constraints, therefore the KKT conditions are necessary and sufficient. The KKT conditions for a vector $\boldsymbol{\beta} \in \mathbb{R}^{p_n m_n}$ to be the solution

of (2.16) is

$$\begin{cases} \frac{1}{n}\Phi_j^T(\boldsymbol{y}-\boldsymbol{\mu}^*) = \lambda_{n2}w_{nj}\frac{\boldsymbol{\beta}_j}{\|\boldsymbol{\beta}_j\|_2}, \text{ if } \|\boldsymbol{\beta}_j\|_2 > 0 \\ \|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\boldsymbol{\mu}^*)\|_2 \leq \lambda_{n2}w_{nj}, \text{ if } \|\boldsymbol{\beta}_j\|_2 = 0 \end{cases} \tag{A.39}$$

where $\boldsymbol{\mu}^* = b'(\Phi\boldsymbol{\beta})$. It is sufficient to show that

$$\mathbb{P}\left(\hat{\boldsymbol{\beta}}_{NZ} \text{ satisfies } (A.39)\right) \to 1$$

Note that for any $j \in T$, we have the KKT conditions for $\hat{\boldsymbol{\beta}}_{NZ}$ that

$$\begin{cases} \frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{NZ}^*) = \lambda_{n2}w_{nj}\frac{\hat{\boldsymbol{\beta}}_{NZj}}{\|\hat{\boldsymbol{\beta}}_{NZj}\|_2}, \text{ if } \|\boldsymbol{\beta}_j\|_2 > 0, \ j \in T \\ \|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{NZ}^*)\|_2 \leq \lambda_{n2}w_{nj}, \text{ if } \|\boldsymbol{\beta}_j\|_2 = 0, \ j \in T \end{cases} \tag{A.40}$$

which are the equality condition in (A.39) and part of the inequality condition in (A.39). Therefore, it suffices to show that

$$\mathbb{P}\left(\|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{NZ}^*)\|_2 \leq \lambda_{n2}w_{nj}, \ \forall \ j \notin T\right) \to 1 \tag{A.41}$$

This is equivalent to show that

$$\mathbb{P}\left(\|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{NZ}^*)\|_2 > \lambda_{n2}w_{nj}, \ \exists \ j \notin T\right) \to 0 \tag{A.42}$$

Use Taylor expansion on $\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{NZ}^*)$, we have

$$\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}_{NZ}^*) = \frac{1}{n}\Phi_j(\boldsymbol{y}-\boldsymbol{\mu}_y) + \frac{1}{n}\Phi_j^T(\boldsymbol{\mu}_y - b'(\Phi\boldsymbol{\beta}^0)) + \frac{1}{n}\Phi_j^T\boldsymbol{\Sigma}\Phi(\hat{\boldsymbol{\beta}}_{NZ}-\boldsymbol{\beta}^0)$$

where $\mathbf{\Sigma}$ is the variance matrix evaluated at some $\boldsymbol{\beta}^*$ located on the line segment joining $\boldsymbol{\beta}^0$ and $\hat{\boldsymbol{\beta}}_{NZ}$. Then we have

$$\begin{aligned}
&\mathbb{P}\left(\|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}^*_{NZ})\|_2 > \lambda_{n2}w_{nj},\ \exists\ j\notin T\right)\\
\leq&\mathbb{P}\left(\|\frac{1}{n}\Phi_j(\boldsymbol{y}-\boldsymbol{\mu}_y)\|_2 > \frac{\lambda_{n2}w_{nj}}{3},\ \exists\ j\notin T\right)\\
&+\mathbb{P}\left(\|\frac{1}{n}\Phi_j^T(\boldsymbol{\mu}_y - b'(\Phi\boldsymbol{\beta}^0))\|_2 > \frac{\lambda_{n2}w_{nj}}{3},\ \exists\ j\notin T\right)\\
&+\mathbb{P}\left(\|\frac{1}{n}\Phi_j^T\mathbf{\Sigma}\Phi(\hat{\boldsymbol{\beta}}_{NZ}-\boldsymbol{\beta}^0)\|_2 > \frac{\lambda_{n2}w_{nj}}{3},\ \exists\ j\notin T\right)\\
\equiv&P_1+P_2+P_3
\end{aligned}$$

Now let's consider P_1. By assumption 2.3, the errors $y_i-\mu_{y_i}$'s are sub-Gaussian. For bounded responses, we have by lemma A.1 and assumption 2.6 that

$$\begin{aligned}
P_1 &= \mathbb{P}\left(\|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}^*_{NZ})\|_2 > \lambda_{n2}w_{nj},\ \exists\ j\notin T\right)\\
&\leq \mathbb{P}\left(\|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}^*_{NZ})\|_2 > C\lambda_{n2}r_n,\ \exists\ j\notin T\right)+o(1)\\
&= \mathbb{P}\left(\max_{j\notin T}\|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}^*_{NZ})\|_2 > C\lambda_{n2}r_n\right)+o(1)\\
&\leq \mathbb{P}\left(\max_{j\notin T}\|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}^*_{NZ})\|_2 > C\lambda_{n2}r_n|\max_{j\notin T}\|\frac{1}{n}\Phi_j^T(\boldsymbol{y}-\hat{\boldsymbol{\mu}}^*_{NZ})\|_2\right.\\
&\qquad\left.\leq Cn^{-1/2}\sqrt{\log(s_n^*m_n)}\right)+o(1)\\
&\to 0\ as\ n\to\infty
\end{aligned}$$

By lemma A.2, we have

$$E\left(\max_{j\notin T,k=1,\ldots,m_n}\left\|\frac{1}{n}\Phi_{jk}^T(\boldsymbol{y}-\boldsymbol{\mu}_y)\right\|_2\right)\leq c_6n^{-1/2}\sqrt{\log(s_n^*m_n)} \tag{A.43}$$

for some constant c_6. Observe that by assumption 2.5, we have $w_{nj} = O_P(r_n) \leq Cr_n$ for some general constant C. Then we have by Markov's inequality and assumption 2.6 that

$$\begin{aligned}
P_1 &= \mathbb{P}\left(\|\frac{1}{n}\Phi_j^T(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{NZ}^*)\|_2 > \lambda_{n2} w_{nj},\ \exists\ j \notin T \right) \\
&\leq \mathbb{P}\left(\|\frac{1}{n}\Phi_j^T(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{NZ}^*)\|_2 > C\lambda_{n2} r_n,\ \exists\ j \notin T \right) + o(1) \\
&= \mathbb{P}\left(\max_{j \notin T} \|\frac{1}{n}\Phi_j^T(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{NZ}^*)\|_2 > C\lambda_{n2} r_n \right) + o(1) \\
&\leq \frac{E\left(\max_{j \notin T, k=1,\ldots,m_n} \left\| \frac{1}{n}\Phi_{jk}^T(\boldsymbol{y} - \boldsymbol{\mu}_y) \right\|_2 \right)}{C\lambda_{n2} r_n} + o(1) \\
&\leq \frac{c_6\sqrt{\log(s_n^* m_n)}}{Cn^{1/2}\lambda_{n2} r_n} + o(1) \to 0 \ as\ n \to \infty
\end{aligned}$$

Then we consider P_2. We have shown that

$$\frac{1}{n}\|\boldsymbol{\mu}_y - \boldsymbol{\mu}_y^*\|_2^2 = O(s_n^2 m_n^{-2d})$$

This implies that

$$\frac{1}{\sqrt{n}}\|\boldsymbol{\mu}_y - \boldsymbol{\mu}_y^*\|_2 = O(s_n m_n^{-d})$$

Then by assumption 2.1,

$$\begin{aligned}
&\max_{j \notin T} \left\| \frac{1}{n}\Phi_j(\boldsymbol{\mu}_y - \boldsymbol{\mu}_y^*) \right\|_2 \\
\leq & C m_n^{-1/2} \frac{1}{\sqrt{n}} \left\| \boldsymbol{\mu}_y - \boldsymbol{\mu}_y^* \right\|_2 \\
= & O(s_n m_n^{-d-1/2})
\end{aligned}$$

By assumption 2.6, we have $P_2 \to 0$ as $n \to \infty$. Next, we look at P_3. By the definition of $\hat{\boldsymbol{\beta}}_{NZ}$, we have by norm inequality

$$\frac{1}{n}\Phi_j^T \boldsymbol{\Sigma}\Phi(\hat{\boldsymbol{\beta}}_{NZ} - \boldsymbol{\beta}^0) = \frac{1}{n}\Phi_j^T \boldsymbol{\Sigma}\Phi_T(\hat{\boldsymbol{\beta}}_{NZT} - \boldsymbol{\beta}_T^0)$$

The MLE on the true nonzero set has a rate of convergence $\sqrt{s_n m_n/n}$. The penalized solution has been proved to be close to the MLE asymptotically ([149]; [46]; [88]). Knowing the true nonzero set, the rate of convergence of $\hat{\boldsymbol{\beta}}_{NZ}$ is $\sqrt{s_n m_n/n}$. Then we have

$$\begin{aligned}
P_3 =& \mathbb{P}\left(\left\|\frac{1}{n}\Phi_j^T \boldsymbol{\Sigma}\Phi_T(\hat{\boldsymbol{\beta}}_{NZT} - \boldsymbol{\beta}_T^0)\right\|_2 > \frac{\lambda_{n2} w_{nj}}{3},\ \exists\ j \notin T\right) \\
\leq& \mathbb{P}\left(\left\|\frac{1}{n}\Phi_j^T \boldsymbol{\Sigma}\Phi_T(\hat{\boldsymbol{\beta}}_{NZT} - \boldsymbol{\beta}_T^0)\right\|_2 > C\lambda_{n2} r_n,\ \exists\ j \notin T\right) + o(1) \\
\leq& \mathbb{P}\left(\max_{j \notin T}\left\|\frac{1}{n}\Phi_j^T \boldsymbol{\Sigma}\Phi_T\right\|_2 > \frac{C\lambda_{n2} r_n}{a_n\sqrt{s_n m_n/n}}\right) + \mathbb{P}\left(\left\|\hat{\boldsymbol{\beta}}_{NZT} - \boldsymbol{\beta}_T^0\right\|_2 > a_n\sqrt{\frac{s_n m_n}{n}}\right) + o(1) \\
\to& 0 \text{ as } n \to \infty
\end{aligned}$$

for any diverging sequence a_n, where the first probability in the last step goes to 0 by assumption 2.1 that the left hand side is of order $m_n^{-1/2}$ and assumption 2.6. The second probability goes to 0 by the rate of convergence of $\hat{\boldsymbol{\beta}}_{NZT}$.

Therefore, we have that $\hat{\boldsymbol{\beta}}_{NZ}$ is our adaptive group lasso solution with probability converging to 1. The components selected by adaptive group lasso is asymptotically at most those which are actually nonzero. Then we want to prove that the true nonzero components

are all selected with probability converging to 1. By our assumptions, we have

$$\begin{aligned}\min_{j\in T}\|\hat{\boldsymbol{\beta}}_{NZj}\|_2 &\geq \min_{j\in T}\|\boldsymbol{\beta}_j^0\|_2 - \|\hat{\boldsymbol{\beta}}_{NZj}-\boldsymbol{\beta}_j^0\|_2\\ &\geq c_2^{-1/2}m_n^{1/2}c_f - o_P(1)\\ &> 0\end{aligned}$$

Therefore, none of the true nonzero components are estimated as zero. Combining the two results above, we have that with probability converging to 1, the components selected by the adaptive group lasso are exactly the true nonzero components, i.e.,

$$\mathbb{P}\left(\hat{\boldsymbol{\beta}}_{AGL}\overset{0}{=}\boldsymbol{\beta}^0\right)\to 1 \text{ as } n\to\infty$$

Part (i) is proved. Then we look at part (ii), where based on the result in part (i), we only consider the high probability event that the selection of the adaptive group lasso estimator is perfect. Similar to part (ii) of theorem 2.1, we consider a convex combination of $\boldsymbol{\beta}^0$ and $\hat{\boldsymbol{\beta}}_{AGL}$

$$\boldsymbol{\beta}^* = t\hat{\boldsymbol{\beta}}_{AGL} + (1-t)\boldsymbol{\beta}^0$$

where $t = N_n/(N_n + \|\hat{\boldsymbol{\beta}}_{AGL}-\boldsymbol{\beta}^0\|_2)$ for some sequence N_n. Similar to (A.26), we have

$$\begin{aligned}\frac{1}{2n}(\boldsymbol{\beta}_T^*-\boldsymbol{\beta}_T^0)^T\Phi_T^T\boldsymbol{\Sigma}\Phi_T(\boldsymbol{\beta}_T^*-\boldsymbol{\beta}_T^0) \leq\ & \frac{(\boldsymbol{\beta}_T^*-\boldsymbol{\beta}_T^0)^T\Phi_T^T(\boldsymbol{y}-\boldsymbol{\mu}_y)}{n}\\ &+\frac{(\boldsymbol{\beta}_T^*-\boldsymbol{\beta}_T^0)^T\Phi_T^T(\boldsymbol{\mu}_y-\boldsymbol{\mu}_y^*)}{n}+\lambda_{n2}\sum_{j=1}^{s_n}w_{nj}(\|\boldsymbol{\beta}^0\|_2-\|\boldsymbol{\beta}^*\|_2)\end{aligned} \tag{A.44}$$

Then by the fact that $|a^T b| \leq \|a\|_2^2 + \|b\|_2^2/4$, we have

$$\begin{aligned}
&\frac{1}{2n}(\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0)^T \Phi_T^T \boldsymbol{\Sigma} \Phi_T (\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0) \\
\leq &\frac{(\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0)^T \Phi_T^T (\boldsymbol{y} - \boldsymbol{\mu}_y)}{n} + \frac{1}{4n}(\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0)^T \Phi_T^T \boldsymbol{\Sigma} \Phi_T (\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0) \\
&+ \frac{\|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{\mu}_y - \boldsymbol{\mu}_y^*)\|_2^2}{n} + \lambda_{n2} \sum_{j=1}^{s_n} w_{nj} (\|\boldsymbol{\beta}^0\|_2 - \|\boldsymbol{\beta}^*\|_2)
\end{aligned}$$

Then by (A.7),

$$\begin{aligned}
&\frac{1}{4n}(\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0)^T \Phi_T^T \boldsymbol{\Sigma} \Phi_T (\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0) \\
\leq &\frac{(\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0)^T \Phi_T^T (\boldsymbol{y} - \boldsymbol{\mu}_y)}{n} + O(s_n^2 m_n^{-2d}) + \lambda_{n2} \sum_{j=1}^{s_n} w_{nj} (\|\boldsymbol{\beta}^0\|_2 - \|\boldsymbol{\beta}^*\|_2)
\end{aligned}$$

By (2.13), the fact that $|a^T b| \leq \|a\|_2^2 + \|b\|_2^2/4$ and norm inequality, we have

$$\begin{aligned}
\frac{\gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1}}{4} \|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2^2 \leq &\frac{(\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0)^T \Phi_T^T (\boldsymbol{y} - \boldsymbol{\mu}_y)}{n} + O(s_n^2 m_n^{-2d}) \\
&+ \frac{2(\max_{j \in T} w_{nj})^2}{\gamma_0 c_1} \lambda_{n2}^2 s_n \\
&+ \frac{\gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1}}{8} \|\boldsymbol{\beta}^* - \boldsymbol{\beta}^0\|_2^2
\end{aligned}$$

Then by assumption 2.6,

$$\frac{\gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1}}{8} \|\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0\|_2^2 \leq \frac{(\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0)^T \Phi_T^T (\boldsymbol{y} - \boldsymbol{\mu}_y)}{n} + O(s_n^2 m_n^{-2d}) + O(\lambda_{n2}^2 s_n)$$

Use the fact that $|a^T b| \leq \|a\|_2^2 + \|b\|_2^2/4$ on the first term of the right hand side, we have

$$\begin{aligned}\frac{\gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1}}{8}\|\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0\|_2^2 \leq & \frac{\gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1}}{16}\|\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0\|_2^2 \\ & + \frac{4}{\gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1} n^2}\|\Phi_T^T(\boldsymbol{y} - \boldsymbol{\mu}_y)\|_2^2 + O(s_n^2 m_n^{-2d}) + O(\lambda_{n2}^2 s_n)\end{aligned}$$

By norm inequality and lemma A.1, we have

$$\begin{aligned}& \frac{4}{\gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1} n^2}\|\Phi_T^T(\boldsymbol{y} - \boldsymbol{\mu}_y)\|_2^2 \\ \leq & \frac{4}{\gamma_0 c_1 \gamma_2^{2s_n} m_n^{-1} n^2} s_n m_n \|\Phi_T^T(\boldsymbol{y} - \boldsymbol{\mu}_y)\|_\infty \\ = & O_P\left(s_n \gamma_2^{-2s_n} m_n \frac{\log(s_n m_n)}{n}\right)\end{aligned}$$

Combine the last two results, we have with probability converging to 1,

$$\|\boldsymbol{\beta}_T^* - \boldsymbol{\beta}_T^0\|_2^2 = O_p\left(s_n \gamma_2^{-2s_n} m_n^2 \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{1-2d}) + O(\lambda_{n2}^2 m_n^2 s_n \gamma_2^{-2s_n})$$

Then similar to the argument in the proof of part (ii) of theorem 2.1, we have

$$\begin{aligned}& \sum_{j \in T} \|\hat{\boldsymbol{\beta}}_{AGLj} - \boldsymbol{\beta}_j^0\|_2^2 \\ = & O_p\left(s_n \gamma_2^{-2s_n} m_n^2 \frac{\log(s_n m_n)}{n}\right) + O(s_n^2 \gamma_2^{-2s_n} m_n^{1-2d}) + O(\lambda_{n2}^2 m_n^2 s_n \gamma_2^{-2s_n})\end{aligned}$$

In the unbounded response case, we replace lemma A.1 with lemma A.2 and get

$$\begin{aligned}&\sum_{j\in T}\|\hat{\boldsymbol{\beta}}_{AGLj}-\boldsymbol{\beta}_j^0\|_2^2\\ =&O_p\left(s_n\gamma_2^{-2s_n}m_n^2a_n\frac{\log(s_nm_n)}{n}\right)+O(s_n^2\gamma_2^{-2s_n}m_n^{1-2d})+O(\lambda_{n2}^2m_n^2s_n\gamma_2^{-2s_n})\end{aligned}$$

for any diverging sequence a_n. Part (ii) is proved. □

Proof of Theorem 2.4

Proof. The proof is similar to the proof of theorem 2.2. □

Proof of Theorem 2.5

Proof. The idea of the proof is similar to the proofs in [52], but due to the group penalization structure, some changes have to be made. First, the GIC criterion has the solution of adaptive group lasso, which is not easy to study. So we use a proxy, the MLE on the nonzero components selected by the adaptive group lasso estimator. Let

$$\hat{\boldsymbol{\beta}}^*(A)=\underset{\{\boldsymbol{\beta}\in\mathbb{R}^{p_nm_n}:\text{supp}_B(\boldsymbol{\beta})=A\}}{\arg\max}\frac{1}{n}\sum_{i=1}^n\left[y_i\left(\boldsymbol{\beta}^T\Phi_i\right)-b\left(\boldsymbol{\beta}^T\Phi_i\right)\right] \tag{A.45}$$

for a given $A\subset\{1,...,p\}$, and the proxy of GIC is defined as

$$GIC_{a_n}^*(A)=\frac{1}{n}\{D(\hat{\mu}_A^*;\boldsymbol{Y})+a_n|A|\} \tag{A.46}$$

where $\hat{\mu}_A^* = b'(\Phi\hat{\boldsymbol{\beta}}^*(A))$. The first result is that the proxy $GIC_{an}^*(T)$ well approximates $GIC_{an}(\lambda_0)$. To prove this, observe by the definition of $\hat{\boldsymbol{\beta}}_0 = \hat{\boldsymbol{\beta}}^*(T)$, we have the first order necessary condition

$$\frac{\partial}{\partial\boldsymbol{\beta}} l_n(\hat{\boldsymbol{\beta}}_0) = \mathbf{0} \tag{A.47}$$

Use Taylor expansion and by assumptions 1 and 2, we have

$$\begin{aligned}
0 &\geq GIC_{an}^*(T) - GIC_{an}(\lambda_0) \\
&= \frac{1}{n}\left(l_n(\hat{\boldsymbol{\beta}}(\lambda_{n0})) - l_n(\hat{\boldsymbol{\beta}}_0)\right) \\
&= -\frac{1}{n}\left(\hat{\boldsymbol{\beta}}(\lambda_{n0}) - \hat{\boldsymbol{\beta}}_0\right)^T \Phi^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^*)\Phi\left(\hat{\boldsymbol{\beta}}(\lambda_{n0}) - \hat{\boldsymbol{\beta}}_0\right) \\
&\geq -c_1\gamma_0 \left\|\hat{\boldsymbol{\beta}}(\lambda_{n0}) - \hat{\boldsymbol{\beta}}_0\right\|_2^2
\end{aligned} \tag{A.48}$$

where $\boldsymbol{\beta}^*$ lies on the line segment joining $\hat{\boldsymbol{\beta}}(\lambda_{n0})$ and $\hat{\boldsymbol{\beta}}_0$. Then we need to bound $\left\|\hat{\boldsymbol{\beta}}(\lambda_{n0}) - \hat{\boldsymbol{\beta}}_0\right\|_2^2$. By the definition of $\hat{\boldsymbol{\beta}}(\lambda_{n0})$, we have

$$\Phi_T^T\left(\boldsymbol{y} - b'(\Phi_T\hat{\boldsymbol{\beta}}_T(\lambda_{n0}))\right) + n\lambda_{n0}\boldsymbol{\nu}_T = \mathbf{0} \tag{A.49}$$

where the elements of $\boldsymbol{\nu}_T$ are $w_{nj}\hat{\boldsymbol{\beta}}_j(\lambda_{n0})/\|\hat{\boldsymbol{\beta}}_j(\lambda_{n0})\|_2$ for $j \in T$. On the other hand, by the definition of $\hat{\boldsymbol{\beta}}_0$, we have

$$\Phi_T^T\left(\boldsymbol{y} - b'(\Phi_T\hat{\boldsymbol{\beta}}_{0T})\right) = \mathbf{0} \tag{A.50}$$

Together we have

$$\Phi_T^T\left(b'(\Phi_T\hat{\boldsymbol{\beta}}_{0T}) - b'(\Phi_T\hat{\boldsymbol{\beta}}_T(\lambda_{n0}))\right) + n\lambda_{n0}\boldsymbol{\nu}_T = \mathbf{0} \tag{A.51}$$

Use Taylor expansion on the left hand side of the equation, we have

$$\Phi_T^T \boldsymbol{\Sigma}(\boldsymbol{\beta}^{**}) \Phi_T \left(\hat{\boldsymbol{\beta}}_T(\lambda_{n0}) - \hat{\boldsymbol{\beta}}_{0T} \right) = n\lambda_{n0} \boldsymbol{\nu}_T \tag{A.52}$$

where $\boldsymbol{\beta}^{**}$ lies on the line segment joining $\hat{\boldsymbol{\beta}}_T(\lambda_{n0})$ and $\hat{\boldsymbol{\beta}}_{0T}$. Taking 2 norm and together with assumptions 1 and 2 and the results in theorem 2.1, we have

$$\left\| \hat{\boldsymbol{\beta}}_T(\lambda_{n0}) - \hat{\boldsymbol{\beta}}_{0T} \right\|_2 \leq C\lambda_{n0} \|\boldsymbol{w}_T\|_2 \leq C\lambda_{n0}\sqrt{s_n} \|\boldsymbol{w}_T\|_\infty \tag{A.53}$$

where $\boldsymbol{w}_T = (w_{nj}, j \in T)'$. Then we have

$$\|\hat{\boldsymbol{\beta}}(\lambda_{n0}) - \hat{\boldsymbol{\beta}}_0\|_2 = O(\lambda_{n0}\sqrt{s_n}) \tag{A.54}$$

Choose a_n to be any diverging sequence, then we have

$$\|\hat{\boldsymbol{\beta}}(\lambda_{n0}) - \hat{\boldsymbol{\beta}}_0\|_2 = o(\lambda_{n0}\sqrt{s_n a_n}) \tag{A.55}$$

Then by (A.48), we have

$$GIC_{a_n}(\lambda_0) - GIC^*_{a_n}(T) = o(\lambda_{n0}\sqrt{s_n a_n}) \tag{A.56}$$

As a direct result,

$$\begin{aligned} GIC_{a_n}(\lambda) - GIC_{a_n}(\lambda_{n0}) &\geq (GIC^*_{a_n}(\alpha_\lambda) - GIC^*_{a_n}(T)) + (GIC^*_{a_n}(T) - GIC_{a_n}(\lambda_{n0})) \\ &= (GIC^*_{a_n}(\alpha_\lambda) - GIC^*_{a_n}(T)) + o_p(\lambda_{n0}\sqrt{s_n a_n}) \end{aligned} \tag{A.57}$$

The using this proxy, next we prove that the proxy GIC^* is able to detect the distance between a selected model and the true model. Since the GIC^* depends only on the MLE and has nothing to do with the penalization, this is the same as the generalized linear model, but with the spline line approximation error being considered.

Due to the estimation problem, we are only interested in the models A such that $|A| \leq K$ where $Km_n = o(n)$. As the proof in [52], we consider the underfitted model and overfitted model (defined in their paper). Briefly, the underfitted models are A such that $A \not\supset T$ and the overfitted models are A such that $A \supsetneq T$. Also in the result of theorem 2.1, the model size $|A| = O(s_n) = o(n)$ and thus the KL divergence has a unique minimiser for every such model A, as discussed in [52].

Lemma A.3 implies that for all underfitted models

$$\begin{aligned} GIC^*_{A_n}(A) - GIC^*_{a_n}(T) &= 2|A|I(\boldsymbol{\beta}^*(A)) + (|A| - |T|)a_n n^{-1} + |A|O_P(R_n) \\ &\geq \delta_n - s_n a_n n^{-1} - O_P(KR_n) \\ &\geq \frac{\delta_n}{2} \end{aligned}$$

if $\delta_n K^{-1} R_n^{-1} \to \infty$ and $a_n = o(\delta_n s_n^{-1} n)$. This result states that there is a negligible increment on the GIC^* if one of the nonzero component is missed, when the parameters satisfy the conditions. Lemma A.4 implies that for all overfitted models

$$GIC^*_{a_n}(A) - GIC^*_{a_n}(T) = \frac{|A| - |T|}{n}[a_n - O_P(\psi_n)] > \frac{a_n}{2n}$$

if $a_n \psi_n \to \infty$. This result states that there is a negligible increment on the GIC^* if one of the zero component is selected along with the true model, when the parameters satisfy the

conditions. Therefore,

$$\mathbb{P}\left(\inf_{A\not\supset T} GIC^*_{a_n}(A) - GIC^*_{a_n}(T) > \frac{\delta_n}{2} \text{ and } \inf_{A\supsetneq T} GIC^*_{a_n}(A) - GIC^*_{a_n}(T) > \frac{a_n}{2n}\right) \to 1 \tag{A.58}$$

Combine this result with (A.57) and theorem assumptions, we have

$$\mathbb{P}\{\inf_{\lambda\in\Omega_-\cup\Omega_+} GIC_{a_n}(\lambda) > GIC_{a_n}(\lambda_{n0})\} \to 1$$

□

Lemma A.3. Under assumptions 2 and 3, as $n \to \infty$, we have

$$\sup_{\substack{|A|\le K\\ A\subset\{1,\ldots,p_n\}}} \frac{1}{n|A|}\left|D(\hat{\boldsymbol{\mu}}^*_A;\boldsymbol{Y}) - D(\hat{\boldsymbol{\mu}}^*_0;\boldsymbol{Y}) - 2I(\boldsymbol{\beta}^*(A))\right| = O_P(R_n)$$

where either a) the responses are bounded or Gaussian distributed, $R_n = \sqrt{\gamma_n m_n \log(p_n)/n}$, and $m_n \log(p_n) = o(n)$; or b) the responses are unbounded and non-Gaussian distributed, $R_n = \sqrt{\gamma_n m_n \log(p_n)/n} + \gamma_n^2 m_n M_n^2 \log(p_n)/n$ and $\log(p) = o(\min\{n(\log n)^{-1}K^{-2}m_n^{-1}\gamma_n^{-1}, nM_n^{-2}\})$.

Proof. lemma A.3 is a direct result from lemma A.7 and lemma A.8. □

Lemma A.4. Under assumption 2.1, 2 and 3, and suppose $\log p = O(n^\kappa)$ for some $0 < \kappa < 1$, as $n \to \infty$, we have

$$\frac{1}{|A| - |T|}\left(D(\hat{\boldsymbol{\mu}}^*_A;\boldsymbol{Y}) - D(\hat{\boldsymbol{\mu}}^*_0;\boldsymbol{Y})\right) = O_P(\psi_n)$$

uniformly for all $A \supsetneq T$ with $|A| < K$ and either a) $\psi_n = m_n\sqrt{\gamma_n \log(p_n)}$ when the responses

are bounded, $K = O(\min\{n^{(1-2\kappa)/6}, n^{(1-3\kappa)/8}\})$ and $\kappa \leq 1/2$; or b) $\psi_n = m_n\gamma_n \log(p_n)$ when the responses are Gaussian bounded; or when the response are unbounded and non-Gaussian distributed, and the last three terms in lemma A.10 are dominated by $m_n\gamma_n \log p_n$.

Proof. lemma A.4 is a direct result from lemma A.9 and A.10. □

Lemma A.5. Under assumptions 2-3, let γ_n be a slowly diverging sequence, if $\gamma_n L_n\sqrt{Km_n \log P_n/n} \to 0$ as $n \to \infty$, where $L_n = O(1)$ for the bounded case and $L_n = O(M_n + \sqrt{\log n})$ for the unbounded case, then we have

$$\sup_{|A|\leq K} \frac{1}{|A|} Z_A\left(\gamma_n L_n \sqrt{|A| m_n \frac{\log p_n}{n}}\right) = O_P\left(\gamma_n^2 L_n^2 \frac{m_n \log p_n}{n}\right)$$

where

$$Z_A(N) = \sup_{\boldsymbol{\beta}\in\mathcal{B}_A(N)} \frac{1}{n} \left|l_n(\boldsymbol{\beta}) - l_n(\boldsymbol{\beta}^*(A)) - E\left[l_n(\boldsymbol{\beta}) - l_n(\boldsymbol{\beta}^*(A))\right]\right|$$

and

$$\mathcal{B}_A(N) = \left\{\boldsymbol{\beta} \in \mathbb{R}^P : \|\boldsymbol{\beta} - \boldsymbol{\beta}^*(A)\|_2 \leq N, \text{supp}_B(\boldsymbol{\beta}) = A\right\} \cup \{\boldsymbol{\beta}^*(A)\}$$

Proof. Define

$$\Omega_n = \{\|\boldsymbol{\varepsilon}\|_\infty \leq \tilde{L_n}\}$$

If we take $\tilde{L_n} = C\sqrt{\log n}$, [52] has showed that $\mathbb{P}(\Omega_n) \to 1$. Let

$$\tilde{Z}_A(N) = \sup_{\boldsymbol{\beta}\in\mathcal{B}_A(N)} \frac{1}{n} \left|l_n(\boldsymbol{\beta}) - l_n(\boldsymbol{\beta}^*(A)) - E\left[l_n(\boldsymbol{\beta}) - l_n(\boldsymbol{\beta}^*(A))|\Omega_n\right]\right|$$

Then we have

$$\sup_{|A|\leq K)} \frac{1}{|A|} Z_A(N) \leq \sup_{|A|\leq K)} \frac{1}{|A|} \tilde{Z}_A(N) + \sup_{|A|\leq K, \boldsymbol{\beta}\in\mathcal{B}_A(N)} \frac{1}{|A|} R_A(\boldsymbol{\beta})$$

where

$$R_A(\boldsymbol{\beta}) = \frac{1}{n} \left| E\left[l_n(\boldsymbol{\beta}) - l_n(\boldsymbol{\beta}^*(A)) \right] - E\left[l_n(\boldsymbol{\beta}) - l_n(\boldsymbol{\beta}^*(A)) | \Omega_n \right] \right|$$

By the definition of l_n, we have

$$\begin{aligned} R_A(\boldsymbol{\beta}) &= \frac{1}{n} \left| E[\boldsymbol{\varepsilon}|\Omega_n]^T \Phi(\boldsymbol{\beta} - \boldsymbol{\beta}^*(A)) \right| \\ &\leq \frac{1}{n} \left\| E[\boldsymbol{\varepsilon}|\Omega_n] \right\|_2 \left\| \Phi(\boldsymbol{\beta} - \boldsymbol{\beta}^*(A) \right\|_2 \\ &= \sqrt{\frac{1}{n} \sum_{i=1}^{n} (E[\epsilon_i|\Omega_n])^2} \cdot \frac{1}{\sqrt{n}} \left\| \Phi(\boldsymbol{\beta} - \boldsymbol{\beta}^*(A) \right\|_2 \\ &\leq C\tilde{L}_n \exp(-C\tilde{L}_n) \|\boldsymbol{\beta} - \boldsymbol{\beta}^*(A)\|_2 \end{aligned}$$

where the first inequality is Cauchy-Schwartz inequality, and the second inequality is lemma 1 in [52] and assumption 2.1. Then we have

$$\sup_{|A| \leq K, \boldsymbol{\beta} \in \mathcal{B}_A(N)} \frac{1}{|A|} R_A(\boldsymbol{\beta}) = C\tilde{L}_n \exp(-C\tilde{L}_n) N$$

Taking $\tilde{L}_n = C\sqrt{\log n}$, $N = \gamma_n L_n \sqrt{|A| \log(p_n m_n)/n}$ and under the lemma assumption, we have

$$\sup_{|A| \leq K, \boldsymbol{\beta} \in \mathcal{B}_A(N)} \frac{1}{|A|} R_A(\boldsymbol{\beta}) = o(\log(p_n m_n)/n) \tag{A.59}$$

Then let's consider $\tilde{Z}_A(N)$. For any $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathcal{B}_A(N)$, by the mean value theorem, we have $b(\Phi_i^T \boldsymbol{\beta}_1) - B(\Phi_i^T \boldsymbol{\beta}_2) = b'(\Phi_i^T \tilde{\boldsymbol{\beta}}) \Phi_i^T (\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2)$, where $\tilde{\boldsymbol{\beta}}$ lies on the line segment joining $\boldsymbol{\beta}_1$

and $\boldsymbol{\beta}_2$. We have the likelihood function

$$\begin{aligned}
&|-y_i\Phi_i^T\boldsymbol{\beta}_1+b(\Phi_i^T\boldsymbol{\beta}_1)-(-y_i\Phi_i^T\boldsymbol{\beta}_2+b(\Phi_i^T\boldsymbol{\beta}_2))|\\
=&|(-y_i+b'(\Phi_i^T\tilde{\boldsymbol{\beta}}))|\Phi_i^T\boldsymbol{\beta}_1-\Phi_i^T\boldsymbol{\beta}_2|\\
\leq&(\tilde{L}_n+2M_n)|\Phi_i^T(\boldsymbol{\beta}_1-\boldsymbol{\beta}_2)|
\end{aligned}$$

to be Lipschitz continuous. Let $w_1,...,w_n$ be a Rademacher sequence independent of $\boldsymbol{\varepsilon}$. By the symmetrization theorem and the concentration inequality, see chapter 14 of [11], we have

$$\begin{aligned}
E[\tilde{Z}_A(N)|\Omega_n]\leq&2E\left[\sup_{\boldsymbol{\beta}\in\mathcal{B}_A(N)}\frac{1}{n}\left|\sum_{i=1}^n w_i[-y_i\Phi_i^T\boldsymbol{\beta}+b(\Phi_i^T\boldsymbol{\beta})\right.\right.\\
&\left.\left.-(-y_i\Phi_i^T\boldsymbol{\beta}^*(A)+b(\Phi_i^T\boldsymbol{\beta}^*(A)))]|\Omega_n\right|\right]\\
\leq&4L_nE\left[\sup_{\boldsymbol{\beta}\in\mathcal{B}_A(N)}\frac{1}{n}\left|\sum_{i=1}^n w_i[\Phi_i(\boldsymbol{\beta}-\boldsymbol{\beta}^*(A)]|\Omega_n\right|\right]\\
\leq&4L_nE\left[\left(\sup_{\boldsymbol{\beta}\in\mathcal{B}_A(N)}\|\boldsymbol{\beta}-\boldsymbol{\beta}^*(A)\|_2\right)\left(\sum_{j\in A}\sum_{i=1}^n\sum_{k=1}^{m_n}\left|\frac{1}{n^2}(w_i\phi_{ijk})^2\right|\right)^{1/2}\right]\\
\leq&4L_nN\sqrt{\frac{|A|m_n}{n}}
\end{aligned}$$

where the second last inequality is by Cauchy-Schwartz inequality, and the last inequality is by the definition of $\mathcal{B}_A(N)$ and w_i. Then since

$$\frac{1}{n}\sum_{i=1}^n(L_n\Phi_i^T(\boldsymbol{\beta}(A)-\boldsymbol{\beta}^0))^2\leq CL_n^2N^2$$

Apply Massart's inequality, see theorem 14.2 in [11], we have

$$\mathbb{P}\left(\tilde{Z}_A(N) \geq E[\tilde{Z}_A(N)|\Omega_n] + t\right) \leq \exp\left(-\frac{1}{CL_n^2N^2}\frac{nt^2}{2}\right)$$

Take $t = 4L_nNu\sqrt{|A|m_n/n}$ with $u > 0$, $N = L_n\sqrt{|A|m_n/n}(1+u)$, $u = \gamma_n\sqrt{\log p_n}$ and observe that $\binom{p_n}{k} \leq (pe/k)^k$, we have

$$\begin{aligned}
&\mathbb{P}\left(\sup_{|A|\leq K}\frac{1}{|A|}\tilde{Z}_A(N) \geq 4L_n^2\frac{m_n}{n}(1+u)^2|\Omega_n\right) \\
\leq &\sum_{|A|\leq K}\mathbb{P}\left(\tilde{Z}_A(N) \geq 4|A|L_n^2\frac{m_n}{n}(1+u)^2|\Omega_n\right) \\
\leq &\sum_{k\leq K}\left(\frac{pe}{k}\right)^k\exp(-CKm_nu^2) \\
\leq &\sum_{k\leq K}\left(\frac{pe}{k}\right)^k\exp(-CKm_n\gamma_n\log p_n) \to 0
\end{aligned}$$

Then we have

$$\mathbb{P}\left(\sup_{|A|\leq K}\frac{1}{|A|}\tilde{Z}_A(N) \geq \gamma_n^2L_n^2\frac{m_n}{n}\log p_n\right) = o(1) + \mathbb{P}(\Omega^c) \to 0$$

□

Lemma A.6. Under assumptions 1-3, we have

$$\sup_{|A|\leq K}\frac{1}{\sqrt{|A|}}\|\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A)\|_2 = O_P\left(\gamma_nL_n\sqrt{\frac{m_n\log p_n}{n}}\right)$$

Proof. Define the convex combination of $\hat{\boldsymbol{\beta}}^*(A)$ and $\boldsymbol{\beta}^*(A)$ to be the same way as we did in

proving theorem 2.1 as $\hat{\boldsymbol{\beta}}_u(A)$. Then is remains to show

$$\sup_{|A|\leq K} \frac{1}{\sqrt{|A|}} \|\hat{\boldsymbol{\beta}}_u(A) - \boldsymbol{\beta}^*(A)\|_2 = O_P\left(\gamma_n L_n \sqrt{\frac{m_n \log p_n}{n}}\right)$$

By the definition of $\hat{\boldsymbol{\beta}}^*(A)$ and the concavity of the likelihood function, we have

$$l_n(\hat{\boldsymbol{\beta}}_u(A)) - l_n(\boldsymbol{\beta}^*(A)) \geq 0$$

By the definition of $\boldsymbol{\beta}^*(A)$, we have

$$E[l_n(\boldsymbol{\beta}^*(A) - l_n(\hat{\boldsymbol{\beta}}_u(A)] \geq 0$$

Combine the two inequalities above, we have

$$0 \leq E[l_n(\boldsymbol{\beta}^*(A) - l_n(\hat{\boldsymbol{\beta}}_u(A)] \leq l_n(\hat{\boldsymbol{\beta}}_u(A)) - l_n(\boldsymbol{\beta}^*(A)) - E[l_n(\hat{\boldsymbol{\beta}}_u(A) - l_n(\boldsymbol{\beta}^*(A)] \leq nZ_A(N) \tag{A.60}$$

On the other hand, for any $\boldsymbol{\beta}_A \in \mathbb{B}_A(N)$, we have

$$\begin{aligned} E[l_n(\boldsymbol{\beta}_A) - l_n(\boldsymbol{\beta}^*(A))] &= E[\boldsymbol{y}^T \Phi \boldsymbol{\beta}_A - \mathbf{1}^T b(\Phi \boldsymbol{\beta}_A) - \boldsymbol{y}^T \Phi \boldsymbol{\beta}^*(A) + \mathbf{1}^T b(\Phi \boldsymbol{\beta}^*(A))] \\ &= b'(\sum_{j=1}^{pn} f_j)^T \Phi[\boldsymbol{\beta}_A - \boldsymbol{\beta}^*(A)] - \mathbf{1}^T[b(\Phi \boldsymbol{\beta}_A) - b(\Phi \boldsymbol{\beta}^*(A))] \end{aligned}$$

Observe that by the definition of $\boldsymbol{\beta}^*(A)$, we have

$$\Phi[b'(\sum_{j=1}^{pn} f_j) - b'(\Phi \boldsymbol{\beta}^*(A))] = \mathbf{0}$$

use Taylor expansion, we have

$$\begin{aligned} E[l_n(\boldsymbol{\beta}_A) - l_n(\boldsymbol{\beta}^*(A))] &= b'(\Phi\boldsymbol{\beta}^*(A))^T\Phi[\boldsymbol{\beta}_A - \boldsymbol{\beta}^*(A)] - \mathbf{1}^T[b(\Phi\boldsymbol{\beta}_A) - b(\Phi\boldsymbol{\beta}^*(A))] \\ &= -\frac{1}{2}(\boldsymbol{\beta}_A - \boldsymbol{\beta}^*(A))^T\Phi_A^T\tilde{\boldsymbol{\Sigma}}\Phi_A(\boldsymbol{\beta}_A - \boldsymbol{\beta}^*(A)) \\ &\leq Cn\|\boldsymbol{\beta}_A - \boldsymbol{\beta}^*(A)\|_2^2 \end{aligned}$$

where the last inequality is by assumptions 1 and 2. Then we have

$$\|\boldsymbol{\beta}_A - \boldsymbol{\beta}^*(A)\|_2^2 \leq CZ_A(N)$$

Take $N = \gamma_n L_n\sqrt{|A|m_n \log p_n/n}$ and by lemma A.5, we have

$$\sup_{|A|\leq K}\frac{1}{\sqrt{|A|}}\|\hat{\boldsymbol{\beta}}_u(A) - \boldsymbol{\beta}^*(A)\|_2 = O_P\left(\gamma_n L_n\sqrt{\frac{m_n\log p_n}{n}}\right)$$

Then lemma A.6 follows. □

Lemma A.7. Under assumptions 1-3, we have

$$\sup_{|A|\leq K}\frac{1}{n|A|}\left(l_n(\hat{\boldsymbol{\beta}}^*(A)) - l_n(\boldsymbol{\beta}^*(A))\right) \leq \frac{\gamma_n^2 L_n^2 m_n\log p_n}{n}$$

Proof. Define the event

$$\mathcal{E} = \left\{\sup_{|A|\leq K}\frac{1}{\sqrt{|A|}}\|\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A)\|_2 = O_P\left(\gamma_n L_n\sqrt{\frac{m_n\log p_n}{n}}\right)\right\}$$

By lemma A.6, we have $\mathbb{P}(\mathbb{E}) \to 1$. Using the same argument as in (A.60) in proving lemma

A.6, we have

$$0 \leq l_n(\hat{\boldsymbol{\beta}}^*(A)) - l_n(\boldsymbol{\beta}^*(A)) \leq l_n(\hat{\boldsymbol{\beta}}_u(A)) - l_n(\boldsymbol{\beta}^*(A)) - E[l_n(\hat{\boldsymbol{\beta}}_u(A) - l_n(\boldsymbol{\beta}^*(A)] \leq nZ_A(N)$$

By lemma A.5, conditioning on $\mathcal{E}$, we have

$$l_n(\hat{\boldsymbol{\beta}}^*(A)) - l_n(\boldsymbol{\beta}^*(A)) \leq nO_P\left(\gamma_n^2 L_n^2 \frac{|A| m_n \log p_n}{n}\right)$$

Then the lemma follow from $\mathbb{P}(A) \leq \mathbb{P}(A|\mathcal{E}) + \mathbb{P}(\mathcal{E}^c)$. □

Lemma A.8. Under assumption 2.1, 2.2 and 2.3, we have

$$\sup_{|A| \leq K} \frac{1}{n|A|} |l_n(\boldsymbol{\beta}^*(A)) - E[l_n(\boldsymbol{\beta}^*(A))]| = O_P\left(\sqrt{\frac{\gamma_n m_n \log p_n}{n}}\right)$$

where $\log p_n = o(n)$ for bounded response and $\gamma_n m_n K^2 \log p_n = o(n)$ for unbounded response.

Proof. By the definition, we have $l_n(\boldsymbol{\beta}^*(A)) - E[l_n(\boldsymbol{\beta}^*(A))] = \boldsymbol{\varepsilon}^T \Phi \boldsymbol{\beta}^*(A)$. For bounded response, by Hoeffding's inequality, we have

$$\begin{aligned}
\mathbb{P}(|\boldsymbol{\varepsilon}^T \Phi \boldsymbol{\beta}^*(A)| \geq t) &\leq C \exp\left(\frac{Ct^2}{\sum_{i=1}^n (\Phi_i^T \boldsymbol{\beta}^*(A))^2}\right) \\
&\leq C \exp\left(-\frac{Ct^2}{n|A|m_n}\right)
\end{aligned}$$

Take $t = |A|\sqrt{n\gamma_n m_n \log p_n}$, we have

$$\mathbb{P}(|\boldsymbol{\varepsilon}^T \Phi \boldsymbol{\beta}^*(A)| \geq |A|\sqrt{n\gamma_n m_n \log p_n}) \leq C \exp(-C|A|\gamma_n \log p_n)$$

Then we have

$$\sup_{|A|\leq K} \frac{1}{n|A|} |l_n(\boldsymbol{\beta}^*(A)) - E[l_n(\boldsymbol{\beta}^*(A))]| = O_P\left(\sqrt{\frac{\gamma_n m_n \log p_n}{n}}\right)$$

If the responses are unbounded, we use Bernstein's inequality. First check the condition

$$\begin{aligned}
E[|\Phi_i \boldsymbol{\beta}^*(A)\epsilon_i|^m] &= m\int_0^\infty x^{m-1}\mathbb{P}(|\Phi_i\boldsymbol{\beta}^*(A)\epsilon_i \geq x)dx \\
&= m|\Phi_i^T\boldsymbol{\beta}^*(A)|^m \int_0^\infty \left(\frac{x}{|\Phi_i^T\boldsymbol{\beta}^*(A)|}\right)^{m-1} \\
&\qquad \mathbb{P}\left(|\epsilon_i| \geq \frac{x}{|\Phi_i^T\boldsymbol{\beta}^*(A)|}\right) d\frac{x}{|\Phi_i^T\boldsymbol{\beta}^*(A)|} \\
&\leq m|\Phi_i^T\boldsymbol{\beta}^*(A)|^m \int_0^\infty t^{m-1} C\exp(-Ct^2)dt) \\
&\leq m|\Phi_i^T\boldsymbol{\beta}^*(A)|^m (\|\Phi\boldsymbol{\beta}^*(A)\|_\infty C)^{m-2}\frac{m!}{2}
\end{aligned}$$

Then by Bernstein's inequality, we have

$$\mathbb{P}(|\boldsymbol{\varepsilon}^T\Phi\boldsymbol{\beta}^*(A)| \geq \sqrt{n}t) \leq 2\exp\left(-\frac{1}{2}\frac{nt^2}{C\|\Phi_A\boldsymbol{\beta}^*(A)\|_2^2 + C\sqrt{n}\|\Phi_A\boldsymbol{\beta}^*(A)\|_\infty t}\right)$$

Taking $t = |A|\sqrt{\gamma_n m_n \log p_n}$, we have

$$\begin{aligned}
&\mathbb{P}(|\boldsymbol{\varepsilon}^T\Phi\boldsymbol{\beta}^*(A)| \geq \sqrt{n}|A|\sqrt{\gamma_n m_n \log p_n}) \\
\leq& 2\exp\left(-\frac{1}{2}\frac{n|A|^2\gamma_n m_n \log p_n}{C\|\Phi_A\boldsymbol{\beta}^*(A)\|_2^2 + C\sqrt{n}\|\Phi_A\boldsymbol{\beta}^*(A)\|_\infty |A|\sqrt{\gamma_n m_n \log p_n}}\right) \\
\to& 0
\end{aligned}$$

if $K^2\gamma_n m_n \log p_n/n \to 0$. □

Lemma A.9. Under assumptions 1-3, we have

$$\sup_{\substack{A\supset T\\ |A|\le K}} \frac{1}{|A|-|T|}(\boldsymbol{y}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{B}_A\boldsymbol{\Sigma}_0^{-1/2}(\boldsymbol{y}-\boldsymbol{\mu}_0) = O_P(m_n(\gamma_n\log p_n)^\xi)$$

where

$$\boldsymbol{B}_A = \boldsymbol{\Sigma}_0^{1/2}\Phi_A(\Phi_A^T\boldsymbol{\Sigma}_0\Phi_A)^{-1}\Phi_A^T\boldsymbol{\Sigma}_0^{1/2}$$

and $\xi = 1/2$ for bounded response and $\xi = 1$ for unbounded response.

Proof. Let $k = |A| - |T|$ and $\boldsymbol{P}_A = \boldsymbol{B}_A - \boldsymbol{B}_T$. It's easy to verify that $\boldsymbol{P}_A$ is a projection matrix, thus we have $tr(\boldsymbol{P}) = km_n$, $\sum_{i=1}^n P_{ii} = km_n$ and $\sum_{i,j} P_{ij} = km_n$. Let

$$\tilde{\boldsymbol{y}} = \boldsymbol{\Sigma}_0^{-1/2}(\boldsymbol{y}-\boldsymbol{\mu}_0)$$

We have the decomposition

$$\frac{1}{m_nk}\tilde{\boldsymbol{y}}^T\boldsymbol{P}_A\tilde{\boldsymbol{y}} = \frac{1}{m_nk}\sum_{i=1}^n P_{ii}\tilde{y}_i^2 + \frac{1}{m_nk}\sum_{i\neq j} P_{ij}\tilde{Y}_i\tilde{Y}_j \equiv I_1(A) + I_2(A)$$

Let $\tilde{y}_i^*$ be independent copies of $\tilde{y}_i$, then by the decoupling inequality, there exists a constant $C > 0$ such that

$$\mathbb{P}\left(\frac{1}{m_nk}|\sum_{i\neq j} P_{ij}\tilde{y}_i\tilde{y}_j| \ge t\right) \le C\mathbb{P}\left(\frac{1}{m_nk}|\sum_{i\neq j} P_{ij}\tilde{Y}_i\tilde{Y}_j^*| \ge C^{-1}t\right)$$

For bounded response, apply Hoeffding's inequality, we have

$$\mathbb{P}(I_1(A) \ge 1+x) \le 2\exp\left(2\frac{Cx^2}{\sum_{i=1}^n (m_nk)^{-2}P_{ii}^2}\right) \le 2\exp(-Cm_nkx^2)$$

Taking $x = \sqrt{\gamma \log p_n}$, use the inequality $\binom{p}{k} \leq (pe/k)^k$ and use the same technique as we used in proving lemma A.5, we have

$$\mathbb{P}\left(\sup_{|A|\leq K} I_1(A) \geq 1 + \sqrt{\gamma_n \log p_n}\right) \leq 2C \sum_{k=1}^{K} \left(\frac{(p_n - s_n)e}{k}\right)^k \exp(-Cm_n k\gamma_n \log p_n) \to 0$$

Then observe $\sum_{i\neq j} P_{ij}^2 = \sum_i (P_{ii} - P_{ii}^2) \leq m_n k$, we have following the decoupling inequality that

$$\begin{aligned}
\mathbb{P}(|I_2(A)| \geq t) &\leq C\mathbb{P}\left(\frac{1}{m_n k}|\sum_{i\neq j} P_{ij}\tilde{Y}_i\tilde{Y}_j^*| \geq C^{-1}t\right) \\
&\leq C\exp\left(-\frac{C^{-2}(m_n k)^2 t^2}{\sum_{i\neq j} P_{ij}^2}\right) \\
&\leq C\exp(-Cm_n k t^2)
\end{aligned}$$

Taking $t = \sqrt{\gamma_n \log p_n}$ and use the same technique as in the previous step, we have

$$\mathbb{P}\left(\sup_{|A|\leq K} I_2(A) \geq 1 + \sqrt{\gamma_n \log p_n}\right) \to 0$$

In the unbounded case, we apply the Bernstein's inequality. In the same way as we did in proving lemma A.8, we check the condition

$$E|P_{ii}\tilde{Y}_i^2|^m \leq m! C^{m-2}\frac{P_{ii}^2}{2}$$

By Bernstein's inequality, we have

$$\mathbb{P}(I_1(A) \geq x^2) \leq 2\exp(-Cm_n k x^2)$$

Taking $x = \sqrt{\gamma_n \log p_n}$, we have

$$\sup_{|A| \leq K} I_1(A) = O_P(\gamma_n \log p_n)$$

For $I_2(A)$, we have

$$\sum_{i \neq j} |P_{ij}|^m E[|\tilde{\boldsymbol{y}}_i \tilde{\boldsymbol{y}}_j^*|^m] \leq m! C^{m-2} \frac{P_{ij}^2}{2}$$

Then by Berstein's inequality and taking $x = \sqrt{\gamma_n \log p_n}$, we have

$$\mathbb{P}(|I_2(A)| \geq \gamma_n \log p_n) \to 0$$

□

Lemma A.10. Under assumptions 1-3, for all $A \supset T$ and $|A| \leq K$, we have

$$\begin{aligned} l_n(\hat{\boldsymbol{\beta}}^*(A)) - l_n(\boldsymbol{\beta}^(A)) = & \frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1/2} \boldsymbol{B}_A \boldsymbol{\Sigma}_0^{-1/2} (\boldsymbol{y} - \boldsymbol{\mu}_0) \\ & + |A|^{5/2} O_P\left(m_n^{5/2} \gamma_n^{5/2} L_n^2 \frac{(\log p_n)^{1+\xi/2}}{\sqrt{n}} \right) \\ & + |A|^4 O_P\left(m_n^4 \gamma_n^4 L_n^4 \frac{(\log p_n)^2}{n} \right) + |A|^3 O_P\left(m_n^3 \gamma_n^3 L_n^3 \frac{(\log p_n)^{3/2}}{\sqrt{n}} \right) \end{aligned}$$

Proof. Use Taylor's expansion, we have

$$\begin{aligned} & l_n(\hat{\boldsymbol{\beta}}^*(A)) - l_n(\boldsymbol{\beta}^*(A)) \\ =& (\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A))^T \Phi^T (\boldsymbol{y} - b'(\Phi \boldsymbol{\beta}^*(A)) - \frac{1}{2} (\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A))^T \Phi^T \boldsymbol{\Sigma}_0 \Phi (\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A)) \\ & + \text{Remainder} \\ \equiv & I_1(A) + I_2(A) + I_3(A) \end{aligned}$$

First, by the definition of $\hat{\boldsymbol{\beta}}^*(A)$, we have

$$\Phi_A^T[\boldsymbol{y} - b'(\Phi\hat{\boldsymbol{\beta}}^*(A))] = 0$$

Then by Taylor expansion, we have

$$\begin{aligned}\Phi_A^T\boldsymbol{y} &= \Phi_A^T b'(\Phi\hat{\boldsymbol{\beta}}^*(A)) \\ &= \Phi_A^T b'(\Phi\boldsymbol{\beta}^*(A)) + \Phi_A^T\boldsymbol{\Sigma}_0\Phi(\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A)) + \Phi_A^T\boldsymbol{\nu}_A\end{aligned}$$

where $\nu_{Ai} = b'''(\Phi_i^T\tilde{\boldsymbol{\beta}}^*(A))(\Phi_i^T(\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A)))^2/2$ and $\tilde{\boldsymbol{\beta}}^*(A)$ lies on the line segment joining $\hat{\boldsymbol{\beta}}^*(A)$ and $\boldsymbol{\beta}^*(A)$. By the definition of $\boldsymbol{\beta}^*(A)$, we have

$$\Phi_A^T[b'(\sum_{j=1}^{p_n} f_j) - b'(\Phi_A\boldsymbol{\beta}^*(A))] = 0$$

we have

$$\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A) = (\Phi_A^T\boldsymbol{\Sigma}_0\Phi_A)^{-1}\Phi_A^T(\boldsymbol{y} - b'(\sum_{j=1}^{p_n} -\boldsymbol{\nu}_A))$$

Therefore, we have

$$I_1(A) = (\boldsymbol{y} - \boldsymbol{\mu}_y)^T\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{B}_A\boldsymbol{\Sigma}_0^{-1/2}(\boldsymbol{y} - \boldsymbol{\mu}_y) + R_{1,A}$$

where $R_{1,A} = -\boldsymbol{\mu}_A^T\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{B}_A\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\varepsilon}$. By Cauchy-Schwartz inequality, we have

$$\begin{aligned}|R_{1,A}| &\leq \|\boldsymbol{B}_A\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\varepsilon}\|_2\|\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\nu}_A\|_2 \\ &\leq (\|\boldsymbol{B}_T\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\varepsilon}\|_2 + \|\tilde{R}_{1,A}\|_2)\|\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\nu}_A\|_2\end{aligned}$$

where $\tilde{R}_{1,A} = (\boldsymbol{B}_A - \boldsymbol{B}_0)\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\varepsilon}$. Observe that $\boldsymbol{\Sigma}_0 = E[\epsilon\epsilon^T]$ and $tr(\boldsymbol{B}_T\boldsymbol{B}_T) = m_n s_n$, take $\gamma_n \to \infty$, by Markov's inequality, we have

$$\begin{aligned}
\mathbb{P}\left(\|\boldsymbol{B}_T\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\varepsilon}\|_2 \geq \sqrt{m_n s_n \gamma_n}\right) &\leq \frac{1}{m_n s_n \gamma_n} E[\|\boldsymbol{B}_T\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\varepsilon}\|_2^2] \\
&= \frac{1}{m_n s_n \gamma_n} tr\{\boldsymbol{B}_T\boldsymbol{\Sigma}_0^{-1/2} E[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T]\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{B}_T\} \\
&= \frac{1}{\gamma_n} \to 0
\end{aligned}$$

Then we have

$$\|\boldsymbol{B}_T\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\varepsilon}\|_2 = O_P(\sqrt{m_n s_n \gamma_n}) \tag{A.61}$$

By lemma A.9, we have

$$(|A| - |T|)^{-1/2}\|\tilde{R}_{1,A}\|_2 = O_P(m_n^{1/2}(\gamma_n \log p_n)^{\xi}) \tag{A.62}$$

Finally, we have

$$\begin{aligned}
\|\boldsymbol{\Sigma}_0^{-1/2}\boldsymbol{\nu}_A\|_2 &\leq C\|\boldsymbol{\nu}_A\|_2 \\
&\leq C\left(\sum_{i=1}^{n} |\Phi_i^T(\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta})^*(A))|^4\right)^{1/2} \\
&\leq C\left(\sum_{i=1}^{n} \|\Phi_{iA}\|_2^4 \|\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta})^*(A)\|_2^4\right)^{1/2} \\
&\leq C m_n |A| n^{1/2} \|\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta})^*(A)\|_2^2 \\
&= m_n^2 |A|^2 O_P\left(\gamma_n^2 L_n^2 \frac{\log p_n}{\sqrt{n}}\right)
\end{aligned} \tag{A.63}$$

Combining (A.61), (A.62) and (A.63), we have

$$I_1(A) = (\boldsymbol{y} - \boldsymbol{\mu}_y)^T \boldsymbol{\Sigma}_0^{-1/2} \boldsymbol{B}_A \boldsymbol{\Sigma}_0^{-1/2} (\boldsymbol{y} - \boldsymbol{\mu}_y) + O_P\left(|A|^{5/2} m_n^{5/2} \gamma_n^{5/2} L_n^2 \frac{(\log p_n)^{1+\xi/2}}{\sqrt{n}} \right)$$

Then we look at $I_2(A)$. We have

$$\begin{aligned} I_2(A) &= \frac{1}{2} (\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A))^T \Phi^T \boldsymbol{\Sigma}_0 \Phi (\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A)) \\ &= \frac{1}{2} (\boldsymbol{y} - \boldsymbol{\mu}_y)^T \boldsymbol{\Sigma}_0^{-1/2} \boldsymbol{B}_A \boldsymbol{\Sigma}_0^{-1/2} (\boldsymbol{y} - \boldsymbol{\mu}_y) + \frac{1}{2} R_{2,A} - R_{1,A} \end{aligned}$$

where

$$\begin{aligned} R_{2,A} &= \boldsymbol{\nu}_A \boldsymbol{\Sigma}_0^{-1/2} \boldsymbol{B}_A \boldsymbol{\Sigma}_0^{-1/2} \boldsymbol{\mu}_A \\ &\leq C \|\boldsymbol{\nu}_A\|_2^2 \\ &\leq C m_n^2 |A|^2 n \|\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A)\|_2^4 \\ &= O\left(m_n^2 |A|^4 \gamma_n^4 L_n^4 \frac{m_n^2 (\log p_n)^2}{n^2} n \right) \\ &= O\left(|A|^4 m_n^4 \gamma_n^4 L_n^4 \frac{(\log p_n)^2}{n} \right) \end{aligned}$$

Therefore,

$$\begin{aligned} I_2(A) =& \frac{1}{2} (\boldsymbol{y} - \boldsymbol{\mu}_y)^T \boldsymbol{\Sigma}_0^{-1/2} \boldsymbol{B}_A \boldsymbol{\Sigma}_0^{-1/2} (\boldsymbol{y} - \boldsymbol{\mu}_y) + O_P\left(|A|^{5/2} m_n^{5/2} \gamma_n^{5/2} L_n^2 \frac{(\log p_n)^{1+\xi/2}}{\sqrt{n}} \right) \\ &+ O\left(|A|^4 m_n^4 \gamma_n^4 L_n^4 \frac{(\log p_n)^2}{n} \right) \end{aligned}$$

Finally, we have for $I_3(A)$ that

$$\begin{aligned} |I_3(A)| &\leq Cn|A|^{3/2} m_n^{3/2} \|\hat{\boldsymbol{\beta}}^*(A) - \boldsymbol{\beta}^*(A)\|_2^3 \\ &= O_P\left(|A|^3 m_n^3 \gamma_n^3 L_n^3 \frac{(\log p_n)^{3/2}}{\sqrt{n}} \right) \end{aligned}$$

Combining the three results for $I_1(A)$, $I_2(A)$ and $I_3(A)$, we get the desired result. □

Proof of Theorem 2.6

Proof. Part (i). Because $\hat{\boldsymbol{\beta}}$ is the minimizer, we have

$$\|\boldsymbol{y} - \boldsymbol{\phi}\hat{\boldsymbol{\beta}}\|_2^2 + \lambda \sum_{j=1}^{p} \mathbf{1}_{\|\hat{\boldsymbol{\beta}}_j\|_2 \neq 0} \leq \|\boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}^0\|_2^2 + \lambda \sum_{j=1}^{p} \mathbf{1}_{\|\boldsymbol{\beta}_j^0\|_2 \neq 0} \tag{A.64}$$

i.e.

$$\|\boldsymbol{y} - \boldsymbol{\phi}\hat{\boldsymbol{\beta}}\|_2^2 + \lambda \hat{k} \leq \|\boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}^0\|_2^2 + \lambda s_n \tag{A.65}$$

Observe that

$$\|\boldsymbol{y} - \boldsymbol{\phi}\hat{\boldsymbol{\beta}}\|_2^2 = \|\boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}^0 + \boldsymbol{\phi}\boldsymbol{\beta}^0 - \boldsymbol{\phi}\hat{\boldsymbol{\beta}}\|_2^2 = \|\boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}^0\|_2^2 + \boldsymbol{\varepsilon}^{*T}\boldsymbol{\phi}(\boldsymbol{\beta}^0 - \hat{\boldsymbol{\beta}}) + \|\boldsymbol{\phi}(\boldsymbol{\beta}^0 - \hat{\boldsymbol{\beta}})\|_2^2$$

where

$$\boldsymbol{\varepsilon}^* = \boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}^0 = \boldsymbol{y} - \boldsymbol{f} + \boldsymbol{f} - \boldsymbol{\phi}\boldsymbol{\beta}^0 := \boldsymbol{\varepsilon} + \boldsymbol{\delta}$$

We have

$$\|\boldsymbol{\phi}(\boldsymbol{\beta}^0 - \hat{\boldsymbol{\beta}})\|_2^2 \leq |\boldsymbol{\varepsilon}^{*T}\boldsymbol{\phi}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)| + \lambda(s_n - \hat{k}) \leq |\boldsymbol{\varepsilon}^{*T}\boldsymbol{\phi}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)| \tag{A.66}$$

By assumptions on eigenvalues of $\boldsymbol{\phi}$, we have

$$\|\boldsymbol{\beta}^0 - \hat{\boldsymbol{\beta}}\|_2^2 \geq cn\gamma_2^{2s_n} m_n^{-1} \|\boldsymbol{\beta}_0 - \hat{\boldsymbol{\beta}}\|_2^2$$

since both $\boldsymbol{\beta}^0$ and $\hat{\boldsymbol{\beta}}$ are sparse. On the other hand, we have

$$\begin{aligned}
|\boldsymbol{\varepsilon}^{*T}\boldsymbol{\phi}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)| \leq & |(\boldsymbol{y} - \boldsymbol{f})^T\boldsymbol{\phi}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)| + |(\boldsymbol{f} - \boldsymbol{\phi}\boldsymbol{\beta}^0)^T\boldsymbol{\phi}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)| \\
\leq & \frac{cn\gamma_2^{2s_n} m_n^{-1}}{2}\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 + \frac{1}{cn\gamma_2^{2s_n} m_n^{-1}}\|\boldsymbol{\phi}(\boldsymbol{y} - \boldsymbol{f})\|_2^2 \\
& + \frac{cn\gamma_2^{2s_n} m_n^{-1}}{4}\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 + \frac{2}{c\gamma_2^{2s_n} m_n^{-1}}\|\boldsymbol{f} - \boldsymbol{\phi}\boldsymbol{\beta}^0\|_2^2
\end{aligned}$$

Combine this with the previous argument, we have

$$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 \leq \frac{4}{c^2 n^2 \gamma_2^{2s_n} m_n^{-1}}\|\boldsymbol{\phi}(\boldsymbol{y} - \boldsymbol{f})\|_2^2 + \frac{8}{c^2 n\gamma_2^{2s_n} m_n^{-1}}\|\boldsymbol{f} - \boldsymbol{\phi}\boldsymbol{\beta}^0\|_2^2 \tag{A.67}$$

By the same arguments in the proof of theorem 2.1, we have

$$\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 = O_p\left(s_n\gamma_2^{-2s_n} m_n^2 \frac{\log(s_n m_n)}{n}\right) + O(s_n^2\gamma_2^{-2s_n} m_n^{1-2d}) \tag{A.68}$$

Part (ii). From (A.66), we have

$$\lambda(\hat{k} - s_n) \leq |\boldsymbol{\varepsilon}^{*T}\boldsymbol{\phi}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)| \leq \frac{1}{n}\|\boldsymbol{\phi}(\boldsymbol{y} - \boldsymbol{f})\|_2^2 + \|\boldsymbol{f} - \boldsymbol{\phi}\boldsymbol{\beta}^0\|_2^2 + n\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 \tag{A.69}$$

Since $\hat{k} \geq s_n$, we have

$$|\hat{k} - s_n| \leq O(\frac{s_n m_n \log(p_n m_n)}{\lambda}) + O(\frac{n s_n^2 m_n^{-2d}}{\lambda}) \to\ 0\ as\ n \to \infty \tag{A.70}$$

if $s_n m_n \log(p_n m_n)/\lambda \to 0$ and $n s_n^2 m_n^{-2d}/\lambda \to 0$. □

Proof of Theorem 2.7

Proof. Part (i). Because $\hat{\boldsymbol{\beta}}$ is the minimiser, we have

$$\|\boldsymbol{y} - \boldsymbol{\phi}\hat{\boldsymbol{\beta}}\|_2^2 + \lambda_1 \sum_{j=1}^{p} \|\hat{\boldsymbol{\beta}}_j\|_2 + \lambda_2 \sum_{j=1}^{p} \mathbf{1}_{\|\hat{\boldsymbol{\beta}}_j\|_2 \neq 0} \leq \|\boldsymbol{y} - \boldsymbol{\phi}\boldsymbol{\beta}^0\|_2^2 + \lambda_1 \sum_{j=1}^{p} \|\boldsymbol{\beta}_j^0\|_2 + \lambda_2 \sum_{j=1}^{p} \mathbf{1}_{\|\boldsymbol{\beta}_j^0\|_2 \neq 0} \tag{A.71}$$

Rearranging the terms and use the theorem conditions, we have

$$\lambda_1 \sum_{j=1}^{p} (\|\hat{\boldsymbol{\beta}}_j\|_2 - \|\boldsymbol{\beta}_j^0\|_2) \leq |\boldsymbol{\varepsilon}^{*T} \boldsymbol{\phi}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)| + \|\boldsymbol{\phi}(\boldsymbol{\beta}^0 - \hat{\boldsymbol{\beta}}\|_2^2$$

Following the same arguments as in (A.26), we have the desired results.

Part (ii). From part (i), we have

$$\begin{aligned} \lambda(\hat{k} - s_n) &\leq |\boldsymbol{\varepsilon}^{*T} \boldsymbol{\phi}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0)| + \lambda_1 \sum_{j=1}^{p} (\|\boldsymbol{\beta}_j^0\|_2 - \|\hat{\boldsymbol{\beta}}_j\|_2) \\ &\leq \frac{1}{n} \|\boldsymbol{\phi}(\boldsymbol{y} - \boldsymbol{f})\|_2^2 + \|\boldsymbol{f} - \boldsymbol{\phi}\boldsymbol{\beta}^0\|_2^2 + n\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2^2 + \lambda_1 \sqrt{s_n} \|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0\|_2 \end{aligned}$$

Then the results follows from the theorem conditions. □

APPENDIX B

Technical Details and Supplementary Materials for Chapter 3.

Proof of Theorem 3.1

In this section, we give the proof for theorem 3.1. To prove the theorem, we need the following lemmas.

Lemma B.1. For a general classifier $\hat{C}(\boldsymbol{x})$ of y, denote $C^*(\boldsymbol{x})$ the Bayes classifier. Then we have

$$R(\hat{C}) - R(C^*) \leq 2\mathbb{E}_{\boldsymbol{X}}[|\sigma(\eta(\boldsymbol{X})) - \sigma(\hat{\eta}(\boldsymbol{X}))|] \leq 2\mathbb{E}_{\boldsymbol{X}}[|\eta(\boldsymbol{X}) - \hat{\eta}(\boldsymbol{X})|]$$

Proof. By the definition of $R(C)$ and $R(C^*)$, we have

$$
\begin{aligned}
& R(\hat{C}) - R(C^*) \\
=& \mathbb{E}_{\boldsymbol{X},Y}\left[\mathbb{1}_{\{\hat{C}(\boldsymbol{X})\neq Y\}}\right] - \mathbb{E}_{\boldsymbol{X},Y}\left[\mathbb{1}_{\{C^*(\boldsymbol{X})\neq Y\}}\right] \\
=& \mathbb{E}_{\boldsymbol{X}}\mathbb{E}_{Y|\boldsymbol{X}}\left[\mathbb{1}_{\{\hat{C}(\boldsymbol{X})\neq Y\}} - \mathbb{1}_{\{C^*(\boldsymbol{X})\neq Y\}}\right] \\
=& \mathbb{E}_{\boldsymbol{X}}\Big[\mathbb{1}_{\{\hat{C}(\boldsymbol{X})=0\}}\eta(\boldsymbol{X}) + \mathbb{1}_{\{\hat{C}(\boldsymbol{X})=1\}}(1-\eta(\boldsymbol{X})) \\
& -\mathbb{1}_{\{C^*(\boldsymbol{X})=0\}}\eta(\boldsymbol{X}) - \mathbb{1}_{\{C^*(\boldsymbol{X})=1\}}(1-\eta(\boldsymbol{X}))\Big] \\
=& \mathbb{E}_{\boldsymbol{X}}\left[\mathbb{1}_{\{\hat{C}(\boldsymbol{X})\neq C^*(\boldsymbol{X})\}}|2\eta(\boldsymbol{X})-1|\right] \\
=& \mathbb{E}_{\boldsymbol{X}}\left[\mathbb{1}_{\{\hat{C}(\boldsymbol{X})=1, C^*(\boldsymbol{X})=0 \text{ or } \hat{C}(\boldsymbol{X})=0, C^*(\boldsymbol{X})=1\}}|2\eta(\boldsymbol{X})-1|\right] \\
=& 2\mathbb{E}_{\boldsymbol{X}}\Big[\mathbb{1}_{\{\sigma(\hat{\eta}(\boldsymbol{X}))\geq 1/2, \sigma(\eta(\boldsymbol{X}))<1/2 \text{ or } \{\sigma(\hat{\eta}(\boldsymbol{X}))<1/2, \sigma(\eta(\boldsymbol{X}))\geq 1/2\}} \\
& |\eta(\boldsymbol{X})-1/2|\Big] \\
\leq& 2\mathbb{E}_{\boldsymbol{X}}\left[|\sigma(\eta(\boldsymbol{X})) - \sigma(\hat{\eta}(\boldsymbol{X}))|\right]
\end{aligned}
$$

For the second inequality, consider the Taylor expansion of $\sigma(\eta(\boldsymbol{X}))$ at $\hat{\eta}(\boldsymbol{X})$, we have

$$
\begin{aligned}
& E_{\boldsymbol{X}}\left[|\sigma(\eta(\boldsymbol{X})) - \sigma(\hat{\eta}(\boldsymbol{X}))|\right] \\
=& E_{\boldsymbol{X}}\left[|\sigma'(\eta^*(\boldsymbol{X}))(\eta(\boldsymbol{X}) - \hat{\eta}(\boldsymbol{X})|\right] \\
\leq& \mathbb{E}_{\boldsymbol{X}}[|\eta(\boldsymbol{X}) - \hat{\eta}(\boldsymbol{X})|]
\end{aligned}
$$

where $\eta^*(\boldsymbol{X})$ lies on the line jointing $\eta(\boldsymbol{X})$ and $\hat{\eta}(\boldsymbol{X})$, and the second inequality follows from the fact that $\sigma'(x) = \exp(x)/(1+\exp(x))^2 \leq 1$. □

Lemma B.2. Under assumptions, we have

$$\frac{1}{n}\|\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}^0\|_2^2 = O_P\left(\frac{\log(n)}{n\tilde{\epsilon}^2}\right) + O\left(\frac{1}{n\tilde{\epsilon}^2 m}\right)$$
$$+O\left(\frac{s^2 m\lambda^2}{n\tilde{\epsilon}^4}\right) + O_P\left(n^{-1}m^{9/2}s^{5/2}\sqrt{\log p}\right)$$

where $\hat{\boldsymbol{\eta}}$ and $\boldsymbol{\eta}^0$ are the vectors of predictions for the sample $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ using the estimated parameters and the true parameters, respectively. Here the four terms come from the estimation, the neural network approximation, the regularization and the excess loss error by the sparse group lasso regularization on $\boldsymbol{\theta}$, respectively.

Proof. By the definition of $\hat{\boldsymbol{\eta}}$, we have

$$\begin{aligned}
&-\frac{1}{n}\sum_{i=1}^{n}\left[y_i\hat{\eta}(\boldsymbol{x}_i) - \log\left(1+\exp(\hat{\eta}(\boldsymbol{x}_i))\right)\right] \\
&+\lambda\alpha\sum_{j=1}^{p}\|\hat{\boldsymbol{\theta}}_{(j)}\|_2 + \lambda(1-\alpha)\|\hat{\boldsymbol{\theta}}\|_1 \\
\leq &-\frac{1}{n}\sum_{i=1}^{n}\left[y_i\eta^0(\boldsymbol{x}_i) - \log\left(1+\exp(\eta^0(\boldsymbol{x}_i))\right)\right] \\
&+\lambda\alpha\sum_{j=1}^{p}\|\boldsymbol{\theta}^0_{(j)}\|_2 + \lambda(1-\alpha)\|\boldsymbol{\theta}^0\|_1
\end{aligned}$$

Rearrange the terms and by Taylor expansion, we have

$$
\begin{aligned}
&-\frac{1}{n}(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0)^T(\boldsymbol{y}-\boldsymbol{\mu}^0)+\frac{1}{2n}(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0)^T\boldsymbol{\Sigma}^0(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0)\\
&-\frac{1}{n}\sum_{i=1}^{n}\frac{\partial^3 l}{\partial\eta^*(\boldsymbol{x}_i)}(\hat{\eta}_i-\eta_i^0)^3\\
\leq&\lambda\sum_{j=1}^{s_n}\Big[\alpha(\|\boldsymbol{\theta}^0_{(j)}\|_2-\|\hat{\boldsymbol{\theta}}_{(j)}\|_2)+(1-\alpha)(\|\hat{\boldsymbol{\theta}}_{(j)}\|_1\\
&-\|\boldsymbol{\theta}^0_{(j)}\|_1)\Big]-\lambda\sum_{j=s_n+1}^{p}\Big[\alpha\|\hat{\boldsymbol{\theta}}_{(j)}\|_2+(1-\alpha)(\|\hat{\boldsymbol{\theta}}^0_{(j)}\|_1\Big]\\
\leq&\lambda\sum_{j=1}^{s_n}\Big[\alpha(\|\boldsymbol{\theta}^0_{(j)}\|_2-\|\hat{\boldsymbol{\theta}}_{(j)}\|_2)+(1-\alpha)(\|\hat{\boldsymbol{\theta}}_{(j)}\|_1-\|\boldsymbol{\theta}^0_{(j)}\|_1)\Big] \qquad \text{(B.1)}
\end{aligned}
$$

where $\boldsymbol{\Sigma}^0$ is diagonal matrix with $\boldsymbol{\Sigma}^0_{ii}=\exp(\eta^0(\boldsymbol{x}_i))/[1+\exp(\eta^0(\boldsymbol{x}_i))]^2$, $\boldsymbol{\mu}^0$ is the conditional expectation of $\boldsymbol{y}$ given $\boldsymbol{X}$ in the neural network approximation space, η^* lies on the line joining $\hat{\eta}$ and η^0. Consider the second order term in (B.1), by assumption 3.2, we have

$$
\frac{1}{2n}(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0)^T\boldsymbol{\Sigma}^0(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0)\geq\frac{\tilde{\epsilon}^2}{2n}\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0\|_2^2
$$

Then the first term on the LHS of (B.1), by $\boldsymbol{u}^T\boldsymbol{v}\leq\|\boldsymbol{u}\|_2^2/4+\|\boldsymbol{v}\|_2^2$, norm inequality, maximal inequality (see proof in [65]) and result in [113],

$$
\begin{aligned}
&\frac{1}{n}|(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0)^T(\boldsymbol{y}-\boldsymbol{\mu}^0)|\\
\leq&\frac{1}{n}|(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0)^T(\boldsymbol{y}-\boldsymbol{\mu})|+\frac{1}{n}|(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0)^T(\boldsymbol{\mu}-\boldsymbol{\mu}^0)|\\
\leq&\frac{\tilde{\epsilon}^2}{4n}\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0\|_2^2+\frac{1}{n\tilde{\epsilon}^2}\|\boldsymbol{y}-\boldsymbol{\mu}\|_2^2+\frac{\tilde{\epsilon}^2}{8n}\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0\|_2^2\\
&+\frac{2}{n\tilde{\epsilon}^2}\|\boldsymbol{\mu}-\boldsymbol{\mu}^0\|_2^2\\
=&\frac{3\tilde{\epsilon}^2}{8n}\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0\|_2^2+O_P\left(\frac{\log(n)}{n\tilde{\epsilon}^2}\right)+O\left(\frac{1}{n\tilde{\epsilon}^2 m}\right)
\end{aligned}
$$

Combine this result and (B.1), we have

$$\begin{aligned}
&\frac{\tilde{\epsilon}^2}{8n}\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0\|_2^2-\frac{1}{n}\sum_{i=1}^{n}\frac{\partial^3 l}{\partial\eta^*(\boldsymbol{x}_i)^3}(\hat{\eta}_i-\eta_i^0)^3\\
\leq&\lambda\sum_{j=1}^{s_n}\Big[\alpha(\|\boldsymbol{\theta}_{(j)}^0\|_2-\|\hat{\boldsymbol{\theta}}_{(j)}\|_2)\\
&+(1-\alpha)(\|\hat{\boldsymbol{\theta}}_{(j)}\|_1-\|\boldsymbol{\theta}_{(j)}^0\|_1)\Big]
\end{aligned}\tag{B.2}$$

Note that

$$\left|\frac{\partial^3 l}{\partial\eta^*(\boldsymbol{x}_i)^3}\right|=\left|\frac{\exp(\eta^*(\boldsymbol{x}_i))[1-\exp(\eta^*(\boldsymbol{x}_i))]}{[1+\exp(\eta^*(\boldsymbol{x}_i))]^3}\right|\leq 1$$

Then apply norm inequality, (B.2) becomes

$$\begin{aligned}
&\frac{\tilde{\epsilon}^2}{8n}\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0\|_2^2-\frac{1}{n}\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0\|_2^3\\
\leq&\lambda\sum_{j=1}^{s_n}\Big[\alpha(\|\boldsymbol{\theta}_{(j)}^0\|_2-\|\hat{\boldsymbol{\theta}}_{(j)}\|_2)+(1-\alpha)(\|\hat{\boldsymbol{\theta}}_{(j)}\|_1\\
&-\|\boldsymbol{\theta}_{(j)}^0\|_1)\Big]+O_P\left(\frac{\log(n)}{n\tilde{\epsilon}^2}\right)+O\left(\frac{1}{n\tilde{\epsilon}^2 m}\right)
\end{aligned}\tag{B.3}$$

Apply the auxiliary lemma in [118], see also [53], we have

$$\begin{aligned}
&\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}^0\|_2^2/nC_0^2\\
\leq&\lambda\sum_{j=1}^{s_n}\Big[\alpha(\|\boldsymbol{\theta}_{(j)}^0\|_2-\|\hat{\boldsymbol{\theta}}_{(j)}\|_2)+(1-\alpha)(\|\hat{\boldsymbol{\theta}}_{(j)}\|_1\\
&-\|\boldsymbol{\theta}_{(j)}^0\|_1)\Big]+O_P\left(\frac{\log(n)}{n\tilde{\epsilon}^2}\right)+O\left(\frac{1}{n\tilde{\epsilon}^2 m}\right)
\end{aligned}\tag{B.4}$$

where

$$C_0 = \max\left(\frac{1}{\epsilon_0\sqrt{n}}, \frac{R^2}{\alpha_{\epsilon_0}\sqrt{n}}\right)$$

$$\epsilon_0 = \frac{\tilde{\epsilon}^2}{16}$$

for some constant R, and some a_{ϵ_0} that depends on ϵ_0, s_n and K. Then by norm inequalities, the first term of RHS of (B.4) becomes

$$\begin{aligned}
&\lambda\sum_{j=1}^{s_n}\Big[\alpha(\|\boldsymbol{\theta}^0_{(j)}\|_2 - \|\hat{\boldsymbol{\theta}}_{(j)}\|_2) + (1-\alpha)(\|\hat{\boldsymbol{\theta}}_{(j)}\|_1 \\
&-\|\boldsymbol{\theta}^0_{(j)}\|_1)\Big] \\
\leq&\lambda\sum_{j=1}^{s_n}\Big[\alpha\|\boldsymbol{\theta}^0_{(j)} - \hat{\boldsymbol{\theta}}_{(j)}\|_2 + (1-\alpha)\|\boldsymbol{\theta}^0_{(j)} - \hat{\boldsymbol{\theta}}_{(j)}\|_1\Big] \\
\leq&\lambda\sum_{j=1}^{s_n}[\alpha + \sqrt{m}(1-\alpha)]\|\boldsymbol{\theta}^0_{(j)} - \hat{\boldsymbol{\theta}}_{(j)}\|_2 \\
\leq&\frac{1}{2}\lambda^2 s[\alpha + \sqrt{m}(1-\alpha)]^2 C_0^2 + \frac{1}{2C_0^2}\|\boldsymbol{\theta}^0_S - \hat{\boldsymbol{\theta}}_S\|_2^2
\end{aligned} \tag{B.5}$$

According to [53], when we choose $\lambda \geq 2T\tilde{\lambda}$ for some constant $T \geq 1$ and $\tilde{\lambda} = c\sqrt{m\log n/n}(\sqrt{\log Q} + \sqrt{m\log p}\log(nm)/(1-\alpha+\alpha/\sqrt{m}))$, we have

$$\begin{aligned}
\frac{1}{2C_0^2}\|\boldsymbol{\theta}^0_S - \hat{\boldsymbol{\theta}}_S\|_2^2 \leq& \left(\tilde{\lambda} \vee \varepsilon(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{t}}, \hat{b})\right) \\
=&O\left(n^{-1}m^{9/2}s^{5/2}\sqrt{\log p}\right)
\end{aligned} \tag{B.6}$$

with probability at least

$$1 - \frac{\sqrt{m}}{n^2} - c \log n \exp\Big(- \frac{T^2(\sqrt{\log Q} + \sqrt{m \log p}\log(nm)/(1 - \alpha + \alpha/\sqrt{m}))^2}{c_1} \Big)$$
$$:= 1 - P_1$$

Combining the results in (B.4), (B.5) and (B.6), we have

$$\frac{1}{n}\|\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}^0\|_2^2 = O_P\left(\frac{\log(n)}{n\tilde{\epsilon}^2}\right) + O\left(\frac{1}{n\tilde{\epsilon}^2 m}\right) + O\left(\frac{s^2 m \lambda^2}{n\tilde{\epsilon}^4}\right) + O_P\left(n^{-1} m^{9/2} s^{5/2}\sqrt{\log p}\right)$$

□

Proof of theorem 3.1

Proof. By lemma B.1, we have

$$R(\hat{C}) - R(C^*) \leq 2\mathbb{E}_{\boldsymbol{X}}[|\eta(\boldsymbol{X}) - \hat{\eta}(\boldsymbol{X})|]$$

Thus it suffices to bound $E_{\boldsymbol{X}}[|\eta(\boldsymbol{X}) - \hat{\eta}(\boldsymbol{X})|]$, or equivalently, $|\eta(\boldsymbol{X}) - \hat{\eta}(\boldsymbol{X})|$ in probability. Let W be the random variable $|\eta(\boldsymbol{X}) - \hat{\eta}(\boldsymbol{X})|$ according to $P_{\boldsymbol{X}}$, then $|\boldsymbol{\eta}(\boldsymbol{X}) - \hat{\boldsymbol{\eta}}(\boldsymbol{X})|$ is a vector of n i.i.d. copies of W, denoted $W_1, ..., W_n$. By lemma B.2, we have

$$\frac{1}{n}\sum_{i=1}^{n} W_i^2 = O\left(\frac{\log(n)}{n\tilde{\epsilon}^2}\right) + O\left(\frac{1}{n\tilde{\epsilon}^2 m}\right) + O\left(\frac{s^2 m \lambda^2}{n\tilde{\epsilon}^4}\right) + O\left(n^{-1} m^{9/2} s^{5/2}\sqrt{\log p}\right)$$

with probability at least $1 - P_2$ for some $P_2 \to 0$ as $n \to \infty$. With proper choice of n, p and other hyper-parameters, we have

$$\frac{1}{n}\sum_{i=1}^{n} W_i^2 \xrightarrow{\mathrm{P}} 0 \; as \; n \to \infty \tag{B.7}$$

Since $\boldsymbol{X} \in \mathcal{X}$ for some bounded space $\mathcal{X}$, by the weak law of large numbers, we have

$$\frac{1}{n}\sum_{i=1}^{n} W_i^2 \xrightarrow{\mathrm{P}} E_{\boldsymbol{X}}[W^2] \; as \; n \to \infty$$

By definition, we have for any $\epsilon > 0$,

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} W_i^2 - E_{\boldsymbol{X}}[W^2]\right| > \epsilon\right) \to \; 0 \; as \; n \to \infty$$

Combine this with (B.7), we have

$$E_{\boldsymbol{X}}[W^2] \to 0 \; as \; n \to \infty$$

Then by Jensen's inequality, we have

$$E_{\boldsymbol{X}}[W] \leq \left(E_{\boldsymbol{X}}[W^2]\right)^{1/2} \to 0 \; as \; n \to \infty$$

Therefore, we have

$$R(\hat{C}) - R(C^*) \leq 2\mathbb{E}_{\boldsymbol{X}}[W] \to 0 \; as \; n \to \infty$$

□

APPENDIX C

Technical Details and Supplementary Materials for Chapter 4

.

Proof of Results in Chapter 4

In this section, we will provide the proof of the theorems in section 4.3.

Proof of Proposition 4.1

Proof. Consider independent observations $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$. Assume

$$x_{(j)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \quad j = 1, ..., p$$

In the regression set up where $\boldsymbol{y}$ is centered, we have

$$y|x_1, ..., x_p \sim \mathcal{N}(\beta_1 x_1 + ... + \beta_p x_p, \sigma^2).$$

Without loss of generality, we assume that

$$|\beta_1| \geq |\beta_2| \geq ... \geq |\beta_s|$$

otherwise, we may re-arrange the order of columns of the design matrix. Furthermore, without loss of generality, we may assume all coefficients are positive, otherwise, we may multiply the corresponding column of the design matrix by -1. Since $s < n$, we may without loss of generality consider an orthogonal design on the matrix $\boldsymbol{x}_{(S)}$, which can be achieved by re-parametrization. Let $\hat{S}$ be the set of variables included in the current model. The algorithm computes

$$\boldsymbol{G}_{0j} = \frac{\partial}{\partial \boldsymbol{W}_{0j}} l(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) := (G_{0j1}, ..., G_{0jK})$$

where K is the size of the first hidden layer. Without loss of generality, we may consider a shallow network in this part, since there isn't any predictor x involved in this section, all estimates can be treated as constants, which are universal for all $j's$. We have

$$G_{0jk} = -\frac{2}{n} \sum_{i=1}^{n} y_i \hat{a}_k \sigma'(\sum_{j=1}^{p} x_{ij} \hat{\theta}_{jk} + \hat{t}_k) x_{ij}, \quad k = 1, ..., K$$

where $\hat{a}_k, \hat{t}_k$ are estimated parameters for the initial model and $\hat{\theta}_{jk}$ is set to zero for all input variables at the very beginning. Thus we have

$$\begin{aligned}\|\boldsymbol{G}_{0j}\|_2 &= \sqrt{\sum_{k=1}^{K}[-\frac{2}{n}\sum_{i=1}^{n} y_i \hat{a}_k \sigma'(\sum_{j=1}^{p} x_{ij}\hat{\theta}_{jk} + \hat{t}_k)x_{ij}]^2} \\ &= \frac{2}{n}\sqrt{\sum_{k=1}^{K} \hat{a}_k^2 \sigma'(\hat{t}_k)}|\boldsymbol{x}_{(j)}^T \boldsymbol{y}|\end{aligned}$$

Since the leading constant is independent of j, it's easier to consider the different part, denoted

$$c_j = |\boldsymbol{x}_{(j)}^T \boldsymbol{y}|$$

for $j \in \mathcal{C}$, where $\mathcal{C}$ is the candidate set. The first variable selected is

$$j_+ = \arg\max_{j \in \mathcal{C}} c_j.$$

At the very beginning, we have for $x \geq 0$ that

$$\begin{aligned}\mathbb{P}(c_1 \leq x) &= \mathbb{P}\left(|\boldsymbol{x}_{(1)}^T \boldsymbol{y}| \leq x\right) \\ &= \mathbb{P}\left(-x \leq \boldsymbol{x}_{(1)}^T \boldsymbol{y} \leq x\right) \\ &= \mathbb{P}\left(-x \leq \beta_1 + \boldsymbol{x}_{(1)}^T \boldsymbol{\varepsilon} \leq x\right) \\ &= \Phi\left(\frac{x - \beta_1}{\sigma\|\boldsymbol{x}_{(1)}\|_2}\right) - \Phi\left(\frac{-x - \beta_1}{\sigma\|\boldsymbol{x}_{(1)}\|_2}\right) \\ &= \Phi\left(\frac{x - \beta_1}{\sigma}\right) - \Phi\left(\frac{-x - \beta_1}{\sigma}\right) \qquad \text{(C.1)}\end{aligned}$$

This result implies that greater β leads to higher probability of large c_1. Then

$$\mathbb{P}(c_1 > c_2) = \mathbb{P}\left(|\boldsymbol{x}_{(1)}^T \boldsymbol{y}| > |\boldsymbol{x}_{(2)}^T \boldsymbol{y}|\right) \tag{C.2}$$

Let

$$W_1 = \boldsymbol{x}_{(1)}^T \boldsymbol{y} \qquad \textit{and} \qquad W_2 = \boldsymbol{x}_{(2)}^T \boldsymbol{y} \tag{C.3}$$

which are both normally distributed. Therefore, c_1 and c_2 follow folded normal distribution

$$c_1 \sim FN(\beta_1, \sigma^2) \qquad \textit{and} \qquad c_2 \sim FN(\beta_2, \sigma^2) \tag{C.4}$$

We can calculate

$$Cov(W_1, W_2) = Cov(\beta_1 + \boldsymbol{x}_{(1)}^T \boldsymbol{y}, \beta_2 + \boldsymbol{x}_{(2)}^T \boldsymbol{y}) = \sigma^2 \boldsymbol{x}_{(1)}^T \boldsymbol{x}_{(2)} = 0 \tag{C.5}$$

Because both W_1 and W_2 are normally distributed, W_1 and W_2 are independent. Therefore, c_1 and c_2 are independent. Since both c_1 and c_2 are positive, the probability is equivalent to

$$\mathbb{P}(c_1 > c_2) = \mathbb{P}\left(\frac{c_1}{c_2} > 1\right) \tag{C.6}$$

Let

$$c_{12} = \frac{c_1}{c_2}$$

Then we have

$$c_{12} \sim RN(\beta_1, \beta_2, \sigma^2, \sigma^2) \tag{C.7}$$

where RN stands for the ratio of folded normal distributions. By theorem 3.1 in [71], we have the CDF of c_{12}

$$\begin{aligned} F_{12}(x) =& 2L\left(\frac{\beta_1-\beta_2 x}{\sigma\sqrt{1+x^2}}, -\frac{\beta_2}{\sigma}, \frac{x}{\sqrt{1+x^2}}\right) + 2L\left(\frac{\beta_1+\beta_2 x}{\sigma\sqrt{1+x^2}}, \frac{\beta_2}{\sigma}, \frac{x}{\sqrt{1+x^2}}\right) + \\ & \Phi\left(\frac{\beta_1-\beta_2 x}{\sigma\sqrt{1+x^2}}\right) + \Phi\left(\frac{\beta_1+\beta_2 x}{\sigma\sqrt{1+x^2}}\right) - 2 \end{aligned} \tag{C.8}$$

where

$$L(a,b,\rho) = \mathbb{P}(X_1 > a, X_2 > b) \tag{C.9}$$

with

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\right)$$

Then we have

$$\begin{aligned} \mathbb{P}(c_1 < c_2) &= F_{12}(1) \\ &= 2L\left(\frac{\beta_1-\beta_2}{\sqrt{2}\sigma}, -\frac{\beta_2}{\sigma}, \frac{1}{\sqrt{2}}\right) + 2L\left(\frac{\beta_1+\beta_2}{\sqrt{2}\sigma}, \frac{\beta_2}{\sigma}, \frac{1}{\sqrt{2}}\right) + \\ &\Phi\left(\frac{\beta_1-\beta_2}{\sqrt{2}\sigma}\right) + \Phi\left(\frac{\beta_1+\beta_2}{\sqrt{2}\sigma}\right) - 2 \end{aligned} \tag{C.10}$$

Release the general assumption of $\beta_j > 0$ by multiply -1 to those which are negative, we have the absolute values back on $|\beta_j|$. This is also true for different β_i and β_j, since we did not use the difference between the nonzero predictors and zero predictors. By the exchangeability

of predictors, the result holds for all i and j. Therefore, we have

$$\begin{aligned}\mathbb{P}(c_j < c_k) =& 2L\left(\frac{|\beta_j| - |\beta_k|}{\sqrt{2}\sigma}, -\frac{|\beta_k|}{\sigma}, \frac{1}{\sqrt{2}}\right) + 2L\left(\frac{|\beta_j| + |\beta_k|}{\sqrt{2}\sigma}, \frac{|\beta_k|}{\sigma}, \frac{1}{\sqrt{2}}\right) + \\ & \Phi\left(\frac{|\beta_j| - |\beta_k|}{\sqrt{2}\sigma}\right) + \Phi\left(\frac{|\beta_j| + |\beta_k|}{\sqrt{2}\sigma}\right) - 2\end{aligned}$$

□

Proof of Proposition 4.2

Proof. Consider independent observations $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$. Assume

$$x_{(j)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \quad j = 1, ..., p$$

In the regression set up where $\boldsymbol{y}$ is centered, we have

$$y|x_1, ..., x_p \sim \mathcal{N}(\beta_1 x_1 + ... + \beta_p x_p, \sigma^2).$$

Without loss of generality, we assume that

$$|\beta_1| \geq |\beta_2| \geq ... \geq |\beta_s|$$

otherwise, we may re-arrange the order of columns of the design matrix. Furthermore, without loss of generality, we may assume all coefficients are positive, otherwise, we may multiply the corresponding column of the design matrix by -1. Since $s < n$, we may without loss of generality consider an orthogonal design on the matrix $\boldsymbol{x}_{(S)}$, which can be achieved by re-parametrization. Let $\hat{S}$ be the set of variables included in the current model.

At the very beginning, we have proved in the proof of proposition 4.1 that

$$c_j \sim FN(\beta_j, \sigma^2) \qquad j = 1, ..., p \tag{C.11}$$

and that c_i and c_j are independent for $i \neq j$. Denote event

$$E_k = \{c_k > \max_{i \neq k} c_i\}, \ k = 1, ..., s \tag{C.12}$$

It's easy to observe that E_k's are mutually exclusive. Therefore, we have

$$\begin{aligned} & Pr(\textit{At least one of } c_1, ..., c_s \textit{ is greater than all of } c_{s+1}, ..., c_p) \\ =& Pr\left(\bigcup_{k=1}^{s} E_k\right) \\ =& \sum_{k=1}^{s} Pr(E_k) \end{aligned} \tag{C.13}$$

We may calculate

$$Pr(E_k) = Pr(c_k > c_{(-k,p-1)})$$

where $c_{(-k,p-1)}$ is the largest order statistic of $c_1, ...c_{(k-1)}, c_{(k+1)}, c_{(p)}$, which is independent of c_k. Let $F_{(-k,p-1)}$ and $f_{(-k,p-1)}$ be the CDF and PDF of $c_{(-k,p-1)}$, respectively, we have

$$F_{(-k,p-1)}(x) = \prod_{j \neq k}^{p} F_j(x)$$

and

$$f_{(-k,p-1)}(x) = \frac{\partial}{\partial x} \prod_{j \neq k}^{p} F_j(x)$$

where from the properties of folded normal distribution we have

$$F_k(x) = \frac{1}{2}\left[erf\left(\frac{x+|\beta_k|}{\sqrt{2\sigma^2}}\right) + erf\left(\frac{x-|\beta_k|}{\sqrt{2\sigma^2}}\right)\right]$$

and

$$f_k(x) = \frac{\partial}{\partial x}F_k(x) = \sqrt{\frac{2}{\pi\sigma^2}}e^{-\frac{x^2+\beta_k^2}{2\sigma^2}}\cosh\frac{\beta_k x}{\sigma^2}$$

Then we have

$$\begin{aligned}
&Pr(E_k)\\
=&Pr(c_k > c_{(-k,p-1)})\\
=&\int_0^\infty Pr(c_k > x) f_{(-k,p-1)}(x)dx\\
=&\int_0^\infty [1-F_k(x)]\frac{\partial}{\partial x}\prod_{j\neq k}^{p} F_j(x)dx\\
=&\left[[1-F_k(x)]\prod_{j\neq k}^{p} F_j(x)\right]\Bigg|_0^\infty + \int_0^\infty f_k(x)\prod_{j\neq k}^{p} F_j(x)dx\\
=&\int_0^\infty f_k(x)\prod_{j\neq k}^{p} F_j(x)dx
\end{aligned} \tag{C.14}$$

where the second equality is by the convolution formula, the fourth equality is by integration by parts. Therefore,

$$\begin{aligned}
&Pr(\textit{At least one of } c_1,...,c_m \textit{ is greater than all of } c_{s+1},...,c_p)\\
=&\sum_{k=1}^{s}\int_0^\infty f_k(x)\prod_{j\neq k}^{p} F_j(x)dx
\end{aligned} \tag{C.15}$$

Next we will show that this probability is actually a very high probability. Let

$$p_k = \int_0^\infty f_k(x) \prod_{j \neq k}^{p} F_j(x) dx = \mathbb{E}_k \left[\prod_{j \neq k}^{p} F_j(X) \right]$$

By the formulas for F_k and f_k, we have

$$\begin{aligned} p_k &= \int_0^\infty \sqrt{\frac{2}{\pi\sigma^2}} e^{-\frac{x^2+\beta_k^2}{2\sigma^2}} \cosh \frac{\beta_k x}{\sigma^2} \prod_{j \neq k} \frac{1}{2} \left[erf\left(\frac{x+\beta_j}{\sqrt{2\sigma^2}} \right) + erf\left(\frac{x-\beta_j}{\sqrt{2\sigma^2}} \right) \right] dx \\ &= \int_0^\infty \sqrt{\frac{1}{2\pi\sigma^2}} \left[e^{-\frac{(x+\beta_k)^2}{2\sigma^2}} + e^{-\frac{(x-\beta_k)^2}{2\sigma^2}} \right] \prod_{j \neq k} \frac{1}{2} \left[erf\left(\frac{x+\beta_j}{\sqrt{2\sigma^2}} \right) + erf\left(\frac{x-\beta_j}{\sqrt{2\sigma^2}} \right) \right] dx \end{aligned} \tag{C.16}$$

Do change of variable $z = x/\sigma$, we have

$$p_k = \int_0^\infty \frac{1}{\sqrt{2\pi}} \left[e^{-\frac{(z+\frac{\beta_k}{\sigma})^2}{2}} + e^{-\frac{(z-\frac{\beta_k}{\sigma})^2}{2}} \right] \prod_{j \neq k} \frac{1}{2} \left[erf\left(\frac{z+\frac{\beta_j}{\sigma}}{\sqrt{2}} \right) + erf\left(\frac{z-\frac{\beta_j}{\sigma}}{\sqrt{2}} \right) \right] dz \tag{C.17}$$

Let $\tilde{\beta}_k = \beta_k/\sigma$, without loss of generality, assume that $\infty = \beta_0 \geq \beta_1 \geq ... \geq \beta_p \geq \beta_{p+1} = 0$, we have

$$\begin{aligned} p_k =& \int_0^\infty \frac{1}{\sqrt{2\pi}} \left[e^{-\frac{(z+\tilde{\beta}_k)^2}{2}} + e^{-\frac{(z-\tilde{\beta}_k)^2}{2}} \right] \prod_{j \neq k} \frac{1}{2} \left[erf\left(\frac{z+\tilde{\beta}_j}{\sqrt{2}} \right) + erf\left(\frac{z-\tilde{\beta}_j}{\sqrt{2}} \right) \right] dz \\ =& \sum_{i=0}^{p} \int_{\beta_{i+1}}^{\beta_i} \frac{1}{\sqrt{2\pi}} \left[e^{-\frac{(z+\tilde{\beta}_k)^2}{2}} + e^{-\frac{(z-\tilde{\beta}_k)^2}{2}} \right] \\ & \prod_{j \neq k} \frac{1}{2} \left[erf\left(\frac{z+\tilde{\beta}_j}{\sqrt{2}} \right) + \mathbb{1}_{\{j \geq i+1\}} erf\left(\frac{z-\tilde{\beta}_j}{\sqrt{2}} \right) - \mathbb{1}_{\{j \leq i\}} erf\left(\frac{\tilde{\beta}_j - z}{\sqrt{2}} \right) \right] dz \end{aligned} \tag{C.18}$$

By the exponential approximation of the error function, see for example [131], there exist c_1 and c_2 such that

$$\sup_{x>0} |erf(x) - (1 - \exp[-c_1 x - c_2 x^2])|$$

can be arbitrarily small, where approximately $c_1 \approx 1.095$ and $c_2 \approx 0.7565$. Consider this approximation, we have

$$\begin{aligned} p_k =& \sum_{i=0}^{p} \int_{\beta_{i+1}}^{\beta_i} \frac{1}{\sqrt{2\pi}} \left[e^{-\frac{(z+\tilde{\beta}_k)^2}{2}} + e^{-\frac{(z-\tilde{\beta}_k)^2}{2}} \right] \\ & \prod_{j \neq k} \frac{1}{2} \left\{ 1 + \mathbb{1}_{\{j \geq i+1\}} - \mathbb{1}_{\{j \leq i\}} - e^{\frac{c_1^2}{4c_2}} \left[e^{-\frac{c_2}{2}\left[z + \left(\tilde{\beta}_j + \frac{c_1}{\sqrt{2}c_2} \right) \right]^2} \right. \right. \\ & \left. \left. + \mathbb{1}_{\{j \geq i+1\}} e^{-\frac{c_2}{2}\left[z + \left(-\tilde{\beta}_j + \frac{c_1}{\sqrt{2}c_2} \right) \right]^2} - \mathbb{1}_{\{j \leq i\}} e^{-\frac{c_2}{2}\left[z - \left(\tilde{\beta}_j + \frac{c_1}{\sqrt{2}c_2} \right) \right]^2} \right] \right\} \end{aligned} \tag{C.19}$$

Here

$$e^{\frac{c_1^2}{4c_2}} \approx 1.48 >> 1$$

Observe that as when $i = s$, also observe that $\tau_n \to 0$ indicates $\max_{j=s+1,\ldots,p} \beta_j \to 0$, we have

$$\prod_{j=1,j\neq k}^{s} \frac{1}{2} e^{\frac{c_1^2}{4c_2}} \left[e^{-\frac{c_2}{2}\left[z-\left(\beta_j+\frac{c_1}{\sqrt{2}c_2}\right)\right]^2} - e^{-\frac{c_2}{2}\left[z+\left(\beta_j+\frac{c_1}{\sqrt{2}c_2}\right)\right]^2} \right] \to 0 \; as \; s \to \infty$$

Therefore, the formula of p_k can be simplified to

$$\begin{aligned}
p_k &= o\left(\frac{1}{2^s} e^{\frac{sc_1^2}{4c_2}}\right) + \sum_{i=0}^{s} \int_{\beta_{i+1}}^{\beta_i} \frac{1}{\sqrt{2\pi}} \left[e^{-\frac{(z+\tilde{\beta}_k)^2}{2}} + e^{-\frac{(z-\tilde{\beta}_k)^2}{2}} \right] \\
&\prod_{j\neq k} \frac{1}{2} \left\{ 1 + \mathbb{1}_{\{j\geq i+1\}} - \mathbb{1}_{\{j\leq i\}} - e^{\frac{c_1^2}{4c_2}} \left[e^{-\frac{c_2}{2}\left[z+\left(\tilde{\beta}_j+\frac{c_1}{\sqrt{2}c_2}\right)\right]^2} \right.\right. \\
&\left.\left. + \mathbb{1}_{\{j\geq i+1\}} e^{-\frac{c_2}{2}\left[z+\left(-\tilde{\beta}_j+\frac{c_1}{\sqrt{2}c_2}\right)\right]^2} - \mathbb{1}_{\{j\leq i\}} e^{-\frac{c_2}{2}\left[z-\left(\tilde{\beta}_j+\frac{c_1}{\sqrt{2}c_2}\right)\right]^2} \right] \right\} \\
&\leq o\left(\frac{1}{2^s} e^{\frac{sc_1^2}{4c_2}}\right) \\
&+ \sum_{i=0}^{s} \left(\frac{1}{2} e^{\frac{c_1^2}{4c_2}}\right)^{s-i} \frac{1}{2s} \left[\Phi(\tilde{\beta}_i - \tilde{\beta}_k) - \Phi(\tilde{\beta}_{i+1} - \tilde{\beta}_k) + \Phi(\tilde{\beta}_i + \tilde{\beta}_k) - \Phi(\tilde{\beta}_{i+1} + \tilde{\beta}_k) \right]
\end{aligned} \tag{C.20}$$

where Φ is the normal CDF and the inequality is by observing

$$e^{-x^2} \leq 1$$

and the term in the bracket is less than 2 when $j \geq i+1$. Then summing up $p_k's$ and observing the double sum is not converging to zero since it consists of a geometric component, when

β_{max} is not big enough and let $s \to \infty$, we have

$$
\begin{aligned}
& 1 - \sum_{k=1}^{s} p_k \\
\geq & 1 - o\left(s\frac{1}{2^s}e^{\frac{sc_1^2}{4c_2}}\right) - \sum_{k=1}^{s}\sum_{i=0}^{s}\left(\frac{1}{2}e^{\frac{c_1^2}{4c_2}}\right)^{s-i} \\
& \frac{1}{2s}\left[\Phi(\tilde{\beta}_i - \tilde{\beta}_k) - \Phi(\tilde{\beta}_{i+1} - \tilde{\beta}_k) + \Phi(\tilde{\beta}_i + \tilde{\beta}_k) - \Phi(\tilde{\beta}_{i+1} + \tilde{\beta}_k)\right] \\
\geq & \sum_{i=1}^{s}\left(\frac{1}{2}e^{\frac{c_1^2}{4c_2}}\right)^{s-i}\sum_{k=1}^{s}\frac{1}{2s}(1 - \Phi(\beta_{max})) - o\left(s\frac{1}{2^s}e^{\frac{sc_1^2}{4c_2}}\right) \\
\geq & c - o(1) \qquad \text{(C.21)}
\end{aligned}
$$

□

Proof of Theorem 4.1

Proof. In this proof, we will show the probability that the same zero predictor appears in k bagging rounds tends to zero as k increases. At the first step, we have $\mathcal{C} = \{1, ..., p\}$ and $\mathcal{S} = \{\}$. By proposition 4.2 we know that the probability that the first variable belongs to $\mathcal{S}_0$ converges to one under the conditions.

At the m^{th} step, denote the candidate set $\mathcal{C}^m$ and the selected set $\mathcal{S}^m$. Assume that $\mathcal{S} \subset \mathcal{S}_0$. Without loss of generality, consider $\sigma^2 = 1$. If not, we may divide the response and coefficients by σ. Consider the first case that

$$\mathcal{C}^m \cap \mathcal{S}_0 \neq \emptyset$$

Let $\mathcal{C}^m \cap \mathcal{S}_0 = \{j_1, ..., j_{s'}\}$. By the proof of proposition 4.2, the probability that a zero

variable is selected is at most

$$\begin{aligned}
\mathbb{P}(\text{select zero variable}) &\leq 1 - \frac{\sum_{j \in \mathcal{C}^m \cap \mathcal{S}_0} e^{\beta_j}}{\sum_{j \in \mathcal{C}^m} e^{\beta_j}} \\
&= 1 - \frac{\sum_{j \in \mathcal{C}^m \cap \mathcal{S}_0} e^{\beta_j}}{\sum_{j \in \mathcal{C}^m \cap \mathcal{S}_0} e^{\beta_j} + \sum_{j \in \mathcal{C}^m \cap \mathcal{S}_0^C} e^{\beta_j}} \\
&= \frac{\sum_{j \in \mathcal{C}^m \cap \mathcal{S}_0^C} e^{\beta_j}}{\sum_{j \in \mathcal{C}^m \cap \mathcal{S}_0} e^{\beta_j} + \sum_{j \in \mathcal{C}^m \cap \mathcal{S}_0^C} e^{\beta_j}} \\
&\leq \frac{(|\mathcal{C}^m| - s')e^{\tau_n}}{s' e^{\gamma_n} + (|\mathcal{C}^m| - s')}
\end{aligned}$$

where $|\mathcal{C}^m| = O(p)$ is the cardinality of $\mathcal{C}^m$ by theorem condition, $\beta_{min} = \min_{j=1,\ldots,s} \beta_j$, $\beta_{max} = \max_{j=1,\ldots,s} \beta_j$ and by assumption 4.1

$$\tau_n = o(\gamma_n) \leq o(\beta_{min})$$

If we have

$$\frac{|\mathcal{C}^m| - s'}{s'} e^{\tau_n - \gamma_n} \to 0 \; as \; n \to \infty \tag{C.22}$$

Then we have

$$\mathbb{P}(\text{select zero variable}) \to 0 \; as \; n \to \infty$$

In this case, the probability of false positive in the ENNS algorithm goes to zero. However, it is not always that equation C.22 is satisfied. It happens that the signal strength of nonzero variable is not big enough. This case can be combined with the other case that

$$\mathcal{C}^m \cap \mathcal{S}_0 = \emptyset$$

In this case, it is (almost) guaranteed that a zero variable will be selected in the next step. However, we will show that though a zero variable is selected, as long as the number of zero variable in $\mathcal{S}$ is not too big, which is guaranteed by the theorem condition

$$s_0 \leq Cs = o(p)$$

the selected zero variables in different rounds of the bagging algorithm together with the neural network random initialization make the probability that the same zero variables appears more than the threshold number of times converges to zero. Now we have

$$\mathcal{C}^m = \{j_1, ..., j_{p'}\} \subset \{s+1, ..., p\}$$

Consider the scenario that all bagging rounds are independent. The residual

$$\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{\mathcal{S}^m}$$

is not related to $x_{j_1}, ..., x_{j'_p}$ by assumption 4.2, where $\mathcal{S}^m$ is the selected set at the m^{th} step and $\hat{\boldsymbol{\mu}}_{\mathcal{S}^m}$ is the estimated conditional expectation of $\boldsymbol{y}$ given $\boldsymbol{x}_{\mathcal{S}^m}$. Therefore, the variables

$\boldsymbol{x}_j$, $j \in \mathcal{C}^m$ are exchangeable. We have

$$\begin{aligned}
\mathbb{P}(j \in \mathcal{S}^{m+1} \cap \mathcal{C}^m) &= \mathbb{P}\left(\frac{1}{B_2}\sum_{b=1}^{B_2} \mathbb{1}_{\{c_j \geq c_{(s_0-|\mathcal{S}^m|)}\}} \geq p_s\right) \\
&= \sum_{k=[B_2 p_s]}^{B_2} \binom{B_2}{k} \frac{(s_0-|\mathcal{S}^m|)^k\,(p-s_0)^{B_2-k}}{(p-|\mathcal{S}^m|)^{B_2}} \\
&\leq \exp\left(-B_2\left[(1-p_s)\log\left(\frac{(1-p_s)(p-|\mathcal{S}^m|)}{p-s_0}\right)\right.\right. \\
&\left.\left.+p_s \log\left(\frac{p_s(p-|\mathcal{S}^m|)}{s_0-|\mathcal{S}^m|}\right)\right]\right) \qquad \text{(C.23)}
\end{aligned}$$

where the last inequality is by [6]. Since we have

$$s_0 \leq Cs = o(p)$$

then we have

$$\mathbb{P}\left(j \in \mathcal{S}^{m+1} \cap \mathcal{C}^m\right) \to 0 \quad as \quad s \; and \; B_2 \to \infty$$

Consider the last case that there is at least one variable in $\mathcal{C}^m$ that is also in $\{1, ..., s\}$ and equation C.22 does not hold. Also consider the truth that the bagging rounds are not fully independent in practice. Consider variable j and the estimator

$$\hat{s}_j^m = \mathbb{1}_{\{\|\boldsymbol{G}_{mj}\|_2 \leq t^m\}}$$

Conditioning on the observations, there exist a fixed t^m such that $\hat{s}_j^m$ indicates whether variable j is not selected $(= 1)$ or selected $(= 0)$. The bagged estimator is defined as

$$\hat{s}_{j,B}^m = \mathbb{E}\left[\mathbb{1}_{\{\|\boldsymbol{G}_{mj}^*\|_2 \leq t_m\}}\right]$$

where $\boldsymbol{G}^*_{mj}$ is $\boldsymbol{G}_{mj}$ evaluated on a bootstrap sample. By the uniform law of large numbers, see for example [61], we have

$$\sup_{\boldsymbol{x},y}\left|\frac{1}{B_2}\sum_{b=1}^{B_2}\mathbb{1}_{\{\|\boldsymbol{G}^*_{mj,b}\|_2\leq t_m\}}-\mathbb{E}\left[\mathbb{1}_{\{\|\boldsymbol{G}^*_{mj}\|_2\leq t_m\}}\right]\right|\to 0 \quad as\ B_2\to\infty \tag{C.24}$$

Let $\mathbb{P}_n^B$ be the empirical measure of the bootstrap sample. It's easy to verify that $\hat{s}_j^m$ is a smooth function evaluated at $\mathbb{P}_n^B$. By assumption 4.3, we have independent observations. Then according to [58], see also [19], there exist an increasing sequence $\{b_n\}_{n\in\mathcal{N}}$ such that

$$b_n(\|\boldsymbol{G}_{mj}\|_2-c_0)\to\mathcal{N}(0,\sigma_\infty^2)$$

for some constant $c_0<\infty$ and $\sigma_\infty^2<\infty$. By algebra, in the m^{th} step, we have

$$\begin{aligned}\|\boldsymbol{G}_{mj}\|_2&=\sqrt{\sum_{k=1}^{K}[-\frac{2}{n}\sum_{i=1}^{n}(y_i-\hat{\mu}_i)\hat{a}_k\sigma'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k)x_{ij}]^2}\\&=\frac{2}{n}\sqrt{\sum_{k=1}^{K}\hat{a}_k^2\left[\hat{\boldsymbol{\varepsilon}}^T\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k)\boldsymbol{x}_{(j)}\right]^2}\end{aligned} \tag{C.25}$$

where in $\hat{\boldsymbol{\theta}}_k$, $\hat{\theta}_{jk}$ is estimated from data for $j\in\mathcal{S}^m$ and $\hat{\theta}_{jk}$ equals zero for $j\in\mathcal{C}^m$, $\hat{\mu}_i$ is the neural network estimate of y_i based on $\boldsymbol{x}_{\mathcal{S}^m}$, $\hat{\boldsymbol{\varepsilon}}$ is the prediction error based on $\boldsymbol{x}_{\mathcal{S}^m}$, $\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k)$ is a diagonal matrix consists of $\sigma'(\cdot)$ evaluated at $\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k$ and $\boldsymbol{x}_{(j)}$ is the

j^{th} column of $\boldsymbol{x}$. We have

$$\begin{aligned}
\hat{\boldsymbol{\varepsilon}} &= \boldsymbol{y} - \hat{f}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{a}}, \hat{b}, \boldsymbol{x}_{\mathcal{S}^m}) \\
&= \sum_{j \in \mathcal{C}^m \cap \{1,...,s\}} \beta_j \boldsymbol{x}_{(j)} + \left[\sum_{j \in \mathcal{S}^m \cap \{1,...,s\}} \beta_j \boldsymbol{x}_{(j)} - \hat{f}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{a}}, \hat{b}, \boldsymbol{x}_{\mathcal{S}^m}) \right] + \boldsymbol{\varepsilon} \\
&= \sum_{j \in \mathcal{C}^m \cap \{1,...,s\}} \beta_j \boldsymbol{x}_{(j)} + \boldsymbol{\varepsilon} + O\left(K_n^2 \sqrt{\frac{\log(nK_n)}{n}} \right) \qquad \text{(C.26)}
\end{aligned}$$

Therefore, for $j \in \mathcal{C}^m \cap \{1, ..., s\}$, since $\boldsymbol{x}_{(j)}$ is normalized and $\sigma'(\cdot) \leq 1$, by norm inequality, we have

$$\begin{aligned}
&\mathbb{E}\|\boldsymbol{G}_{mj}\|_2 \\
\approx &\mathbb{E}\left[\frac{2}{n} \sqrt{ \sum_{k=1}^{K} \hat{a}_k^2 \left[\sum_{j' \in \mathcal{C}^m \cap \{1,...,s\}} \beta_{j'} \boldsymbol{x}_{(j')}^T \boldsymbol{\Sigma}'(\boldsymbol{x}_i^T \hat{\boldsymbol{\theta}}_k + \hat{t}_k) \boldsymbol{x}_{(j)} + \boldsymbol{\varepsilon}^T \boldsymbol{\Sigma}'(\boldsymbol{x}_i^T \hat{\boldsymbol{\theta}}_k + \hat{t}_k) \boldsymbol{x}_{(j)} \right]^2 } \right] \\
\geq &\mathbb{E}\left[\frac{2}{nK} \sum_{k=1}^{K} |\hat{a}_k| \left| \sum_{j' \in \mathcal{C}^m \cap \{1,...,s\}} \beta_{j'} \boldsymbol{x}_{(j')}^T \boldsymbol{\Sigma}'(\boldsymbol{x}_i^T \hat{\boldsymbol{\theta}}_k + \hat{t}_k) \boldsymbol{x}_{(j)} + \boldsymbol{\varepsilon}^T \boldsymbol{\Sigma}'(\boldsymbol{x}_i^T \hat{\boldsymbol{\theta}}_k + \hat{t}_k) \boldsymbol{x}_{(j)} \right| \right] \\
\geq &\frac{c \cdot |\mathcal{C}^m \cap \{1, ..., s\}|}{nK} \gamma_n \qquad \text{(C.27)}
\end{aligned}$$

For $j \in \mathcal{C}^m \cap \{s+1, ..., p\}$, by Jensen's inequality, we have

$$\begin{aligned}
&\mathbb{E}\|\boldsymbol{G}_{mj}\|_2 \\
\approx &\mathbb{E}\left[\frac{2}{n}\sqrt{\sum_{k=1}^{K}\hat{a}_k^2\left[\sum_{j'\in\mathcal{C}^m\cap\{1,...,s\}}\beta_{j'}\boldsymbol{x}_{(j')}^T\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k)\boldsymbol{x}_{(j)}+\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k)\boldsymbol{x}_{(j)}\right]^2}\right] \\
\leq &\frac{2}{n}\sqrt{\mathbb{E}\left\{\sum_{k=1}^{K}\hat{a}_k^2\left[\sum_{j'\in\mathcal{C}^m\cap\{1,...,s\}}\beta_{j'}\boldsymbol{x}_{(j')}^T\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k)\boldsymbol{x}_{(j)}+\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k)\boldsymbol{x}_{(j)}\right]^2\right\}} \\
\leq &\frac{c'}{n\sqrt{K}} \qquad \text{(C.28)}
\end{aligned}$$

If $\gamma_n \geq \frac{c'\sqrt{K}}{c}$, we have

$$\mathbb{P}\left(\min_{j\in\mathcal{C}^m\cap\{1,...,s\}}\mathbb{E}\|\boldsymbol{G}_{mj}\|_2 \geq \max_{j\in\mathcal{C}^m\cap\{s+1,...,p\}}\mathbb{E}\|\boldsymbol{G}_{mj}\|_2\right) \to 1$$

Since we have $s_0 \leq Cs = o(p)$, taking t^m to be the $(|\mathcal{C}^m| - |\mathcal{S}^m|)^{th}$ smallest value of $\|\boldsymbol{G}_{mj}\|_2,\ j \in \mathcal{C}^m$, combine this with equation C.24, for $j \in \mathcal{C}^m \cap \{s+1, ..., p\}$, we have

$$\begin{aligned}
&\frac{1}{B_2}\sum_{b=1}^{B_2}\mathbb{1}_{\{\|\boldsymbol{G}^*_{mj,b}\|_2\leq t_m\}} \leq \mathbb{E}\left[\mathbb{1}_{\{\|\boldsymbol{G}^*_{mj}\|_2\leq t_m\}}\right] + \epsilon \to \Phi\left(\frac{b_n(t^m - c_0)}{\sigma_\infty} - Z\right) \\
&\textit{as } n \textit{ and } B_2 \to \infty \qquad \text{(C.29)}
\end{aligned}$$

where the result is by [19], Z is standard normal random variable and $\Phi(\cdot)$ is the standard normal CDF. Observe that b_n is a diverging sequence and $s_0 \leq Cs = o(p)$, then we have the

probability that a zero variable is selected

$$
\begin{aligned}
&\mathbb{P}\left(j \in \mathcal{C}^m \cap \mathcal{S}^{m+1^c} \cap \{s+1, ..., p\}\right) \\
=&\mathbb{P}\left(j \in \mathcal{C}^m \cap \mathcal{S}^{m+1^c} \cap \{s+1, ..., p\} \middle| \mathbb{E}\|\boldsymbol{G}_{mj}\|_2 \leq t_m\right) \mathbb{P}\left(\mathbb{E}\|\boldsymbol{G}_{mj}\|_2 \leq t_m\right) \\
&+ \mathbb{P}\left(j \in \mathcal{C}^m \cap \mathcal{S}^{m+1^c} \cap \{s+1, ..., p\} \middle| \mathbb{E}\|\boldsymbol{G}_{mj}\|_2 \geq t_m\right) \mathbb{P}\left(\mathbb{E}\|\boldsymbol{G}_{mj}\|_2 \geq t_m\right) \\
\leq&\mathbb{P}\left(j \in \mathcal{C}^m \cap \mathcal{S}^{m+1^c} \cap \{s+1, ..., p\} \middle| \mathbb{E}\|\boldsymbol{G}_{mj}\|_2 \leq t_m\right) + \mathbb{P}\left(\mathbb{E}\|\boldsymbol{G}_{mj}\|_2 \geq t_m\right) \\
\approx&1 - \Phi\left(\frac{b_n(t^m - \mathbb{E}\|\boldsymbol{G}_{mj}\|_2)}{\sigma_\infty} - Z\right) + \frac{s_0 - |\mathcal{S}^m| - |\mathcal{C}^m \cap \{1, ..., s\}|}{p - |\mathcal{S}^m|} \\
\to&0 \quad as\ n \to \infty\ and\ B_2 \to \infty
\end{aligned} \tag{C.30}
$$

Therefore, the false positive rate of the ENNS algorithm goes to zero.

In the classification set up, we have similarly for equation C.25 that

$$
\begin{aligned}
\|\boldsymbol{G}_{mj}\|_2 &= \sqrt{\sum_{k=1}^{K}[-\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\mu}_i)\hat{a}_k\sigma'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k + \hat{t}_k)x_{ij}]^2} \\
&= \frac{1}{n}\sqrt{\sum_{k=1}^{K}\hat{a}_k^2\left[\hat{\boldsymbol{\varepsilon}}^T\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k + \hat{t}_k)\boldsymbol{x}_{(j)}\right]^2}
\end{aligned} \tag{C.31}
$$

where in $\hat{\boldsymbol{\theta}}_k$, $\hat{\theta}_{jk}$ is estimated from data for $j \in \mathcal{S}^m$ and $\hat{\theta}_{jk}$ equals zero for $j \in \mathcal{C}^m$, $\hat{\mu}_i$ is the neural network estimate of the mean of y_i based on $\boldsymbol{x}_{\mathcal{S}^m}$, i.e.

$$
\hat{\mu}_i = \sigma\left(\sum_{k=1}^{K}\hat{\alpha}_k\sigma(\boldsymbol{\theta}_k^T\boldsymbol{x}_i + t_k) + b\right),
$$

$\hat{\boldsymbol{\varepsilon}}$ is the prediction error based on $\boldsymbol{x}_{\mathcal{S}^m}$, $\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k + \hat{t}_k)$ is a diagonal matrix consists of $\sigma'(\cdot)$ evaluated at $\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k + \hat{t}_k$ and $\boldsymbol{x}_{(j)}$ is the j^{th} column of $\boldsymbol{x}$. The only difference between the regression set up and the classification set up is the formula for the mean. Use Taylor

expansion with Lagrange remainder, we have

$$\begin{aligned}\sigma\left(\sum_{j=1}^{s}\beta_j x_j\right) =&\sigma\left(\sum_{j\in\mathcal{C}^m\cap\{1,\ldots,s\}}\beta_j x_j\right)+\sigma'\left(\sum_{j\in\mathcal{S}^m\cap\{1,\ldots,s\}}\beta_j x_j+\xi\sum_{j\in\{1,\ldots,s\}/\mathcal{S}^m}\beta_j x_j\right)\\ &\sum_{j\in\{1,\ldots,s\}/\mathcal{S}^m}\beta_j x_j\end{aligned}$$

for some $\xi\in(0,1)$ and

$$\begin{aligned}0<&\sigma'\left(\sum_{j\in\mathcal{S}^m\cap\{1,\ldots,s\}}\beta_j x_j+\xi\sum_{j\in\{1,\ldots,s\}/\mathcal{S}^m}\beta_j x_j\right)\\ <&\sigma\left(\sum_{j\in\mathcal{S}^m\cap\{1,\ldots,s\}}\beta_j x_j+\xi\sum_{j\in\{1,\ldots,s\}/\mathcal{S}^m}\beta_j x_j\right)<1\end{aligned}$$

Then

$$\begin{aligned}\hat{\varepsilon} =&\boldsymbol{\varepsilon}+\sigma'\left(\sum_{j\in\mathcal{S}^m\cap\{1,\ldots,s\}}\beta_j x_j+\xi\sum_{j\in\{1,\ldots,s\}/\mathcal{S}^m}\beta_j x_j\right)\\ &\sum_{j\in\{1,\ldots,s\}/\mathcal{S}^m}\beta_j x_j+O\left(K_n^2\sqrt{\frac{\log(nK_n)}{n}}\right)\end{aligned}$$

where $\boldsymbol{\varepsilon}$ is the theoretical error of Bernoulli distribution with their means. We don't have a direct control on $\boldsymbol{\varepsilon}$, but by Cauchy-Schwarz inequality we have for any $\delta>0$ that

$$\begin{aligned}&\mathbb{P}\left(\frac{1}{n}\left|\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k)\boldsymbol{x}_{(j)}\right|>\delta\right)\\ \leq&\mathbb{P}\left(\frac{1}{n}\max_i\boldsymbol{\Sigma}'(\boldsymbol{x}_i^T\hat{\boldsymbol{\theta}}_k+\hat{t}_k)\left\|\boldsymbol{\varepsilon}\right\|_2\|\boldsymbol{x}_{(j)}\|_2\right|>\delta\Big)\\ \leq&\mathbb{P}\left(\frac{1}{n}\left\|\boldsymbol{\varepsilon}\right\|_2>\delta\right)\\ \leq&e^{-n\delta^2/8}\end{aligned}\tag{C.32}$$

Therefore, the only difference between classification and regression is the first order approximation term

$$\sigma'\left(\sum_{j\in\mathcal{S}^m\cap\{1,...,s\}}\beta_j x_j+\xi\sum_{j\in\{1,...,s\}/\mathcal{S}^m}\beta_j x_j\right)$$

Observe that this term only depends on the true relationship and is independent of any $j\in\mathcal{C}^m$, therefore can be treated as a constant when comparing $\|G_{mj}\|_2$. This finishes the proof for the classification case. □

Proof of Theorem 4.2

Proof. In the proof of theorem 4.1, we have proved that with probability tending to 1, the algorithm won't select any zero variables. Therefore, here it suffices to show that the model will be able to include all nonzero variables in the model. Though it looks complicated, we only need to consider the worst case:

$$\mathcal{S}^m=\{1,...,s-1\}\quad and\quad \mathcal{C}^m=\{s,s+1,...,p\}$$

and prove that variable s will be selected in the next step, since variable s has the smallest true coefficient β_s among $\{1,...,s\}$ and thus all other cases have greater probability to selected a nonzero variable. Note variable s will be selected

$$s\in\mathcal{S}^{m+1}\quad\Longleftrightarrow\quad\|\boldsymbol{G}_{ms}\|_2=\max_{j\in\mathcal{C}^m}\|\boldsymbol{G}_{mj}\|_2$$

Now we have

$$\|\boldsymbol{G}_{mj}\|_2=\frac{2}{n}\sqrt{\sum_{k=1}^{K}\hat{a}_k^2\left[\hat{\boldsymbol{\varepsilon}}^T\boldsymbol{\Sigma}'(\boldsymbol{x}_i^{s-1^T}\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k)\boldsymbol{x}_{(j)}\right]^2}$$

where ${\boldsymbol{x}^{s-1}}_i$ is the first $s-1^{th}$ elements in $\boldsymbol{x}_i$, $\hat{\boldsymbol{\theta}}_k^{s-1}$ is estimated from data as the coefficient of $\boldsymbol{x}_i^{s-1}$, $\hat{\mu}_i$ is the neural network estimate of y_i based on $\boldsymbol{x}^{s-1}$, $\hat{\boldsymbol{\varepsilon}}$ is the prediction error based on $\boldsymbol{x}^{s-1}$, $\boldsymbol{\Sigma}'({\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k)$ is a diagonal matrix consists of $\sigma'(\cdot)$ evaluated at ${\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k$ and $\boldsymbol{x}_{(j)}$ is the j^{th} column of $\boldsymbol{x}$.

Here we need the probability that $\|\boldsymbol{G}_{ms}\|_2$ being the greatest among all candidates to be very big, so that it will not be missed in the ensemble filtering. For $j \in \{s+1, ..., p\}$, we have

$$
\begin{aligned}
&\mathbb{P}\left(\|\boldsymbol{G}_{ms}\|_2 > \|\boldsymbol{G}_{mj}\|_2\right) \\
=&\mathbb{P}\left(\sum_{k=1}^{K}\hat{\alpha}_k^2\left[\hat{\boldsymbol{\varepsilon}}^T\boldsymbol{\Sigma}'({\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k)\boldsymbol{x}_{(s)}\right]^2 > \sum_{k=1}^{K}\hat{\alpha}_k^2\left[\hat{\boldsymbol{\varepsilon}}^T\boldsymbol{\Sigma}'({\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k)\boldsymbol{x}_{(j)}\right]^2\right) \\
=&\mathbb{P}\left(\sum_{k=1}^{K}\left[\left(\hat{\boldsymbol{\varepsilon}}^T\hat{\alpha}_k\boldsymbol{\Sigma}'({\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k)\boldsymbol{x}_{(s)}\right)^2 - \left(\hat{\boldsymbol{\varepsilon}}^T\hat{\alpha}_k\boldsymbol{\Sigma}'({\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k)\boldsymbol{x}_{(s)}\right)^2\right] > 0\right)
\end{aligned}
\tag{C.33}
$$

In the regression set up, observe that

$$
\max_i \boldsymbol{\Sigma}'_{ii}({\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k) = \max_i \frac{\sigma({\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k)}{1+\exp({\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k)} \leq \max_i \sigma({\boldsymbol{x}_i^{s-1}}^T\hat{\boldsymbol{\theta}}_k^{s-1}+\hat{t}_k) \leq 1
$$

and

$$
\hat{\boldsymbol{\varepsilon}} = \beta_s\boldsymbol{x}_{(s)} + \boldsymbol{\varepsilon} + O\left(K_n^2\sqrt{\frac{\log(nK_n)}{n}}\right)
$$

Also by the fact that

$$
A \implies B \quad \implies \quad \mathbb{P}(A) \leq \mathbb{P}(B)
$$

we have for regression that

$$
\begin{aligned}
&\mathbb{P}\left(\|\boldsymbol{G}_{ms}\|_2 > \|\boldsymbol{G}_{mj}\|_2\right) \\
=&\mathbb{P}\left(\sum_{k=1}^{K}\left[\left(\left(\beta_s \boldsymbol{x}_{(s)}+\boldsymbol{\varepsilon}\right)^T \boldsymbol{\Sigma}_k \boldsymbol{x}_{(s)}\right)^2 - \left(\left(\beta_s \boldsymbol{x}_{(s)}+\boldsymbol{\varepsilon}\right)^T \boldsymbol{\Sigma}_k \boldsymbol{x}_{(j)}\right)^2\right]\right. \\
&\left.\geq O\left(KK_n^2\sqrt{\frac{\log(nK_n)}{n}}\right)\right) \\
=&\mathbb{P}\left(\sum_{k=1}^{K}\left[\beta_s^2\left[\left(\boldsymbol{x}_{(s)}^T\boldsymbol{\Sigma}_k\boldsymbol{x}_{(s)}\right)^2 - \left(\boldsymbol{x}_{(s)}^T\boldsymbol{\Sigma}_k\boldsymbol{x}_{(j)}\right)^2\right] + 2\beta_s\left[\boldsymbol{x}_{(s)}^T\boldsymbol{\Sigma}\boldsymbol{x}_{(s)}\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}(\boldsymbol{x}_{(s)}-\boldsymbol{x}_{(j)})\right]\right. \\
&\left.+\left[\left(\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}\boldsymbol{x}_{(s)}\right)^2 - \left(\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}\boldsymbol{x}_{(j)}\right)^2\right]\right] \geq O\left(KK_n^2\sqrt{\frac{\log(nK_n)}{n}}\right)\right) \\
\geq&\mathbb{P}\left(\sum_{k=1}^{K}\left[c'\beta_s\left[\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}(\boldsymbol{x}_{(s)}-\boldsymbol{x}_{(j)})\right] + \left[\left(\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}\boldsymbol{x}_{(s)}\right)^2 - \left(\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}\boldsymbol{x}_{(j)}\right)^2\right]\right] \geq\right. \\
&\left.-cK\beta_s^2 + O\left(KK_n^2\sqrt{\frac{\log(nK_n)}{n}}\right)\right)
\end{aligned}
\tag{C.34}
$$

Observe by assumption 4.3 that $x_{(s)}$ and $x_{(j)}$ are independent and identically distributed, we have

$$
\mathbb{P}\left(\left(\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}\boldsymbol{x}_{(s)}\right)^2 > \left(\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}\boldsymbol{x}_{(j)}\right)^2\right) = \frac{1}{2}
$$

Therefore, we have

$$
\begin{aligned}
&\mathbb{P}\left(\|\boldsymbol{G}_{ms}\|_2 > \|\boldsymbol{G}_{mj}\|_2\right) \\
\geq &\mathbb{P}\left(c'\beta_s\left[\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}(\boldsymbol{x}_{(s)}-\boldsymbol{x}_{(j)})\right]+\left[\left(\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}\boldsymbol{x}_{(s)}\right)^2-\left(\boldsymbol{\varepsilon}^T\boldsymbol{\Sigma}\boldsymbol{x}_{(j)}\right)^2\right]\right. \\
&\left.\geq -c\beta_s^2+O\left(K_n^2\sqrt{\frac{\log(nK_n)}{n}}\right)\right)^K \\
\to &\Phi\left(\frac{c\beta_s}{\|\boldsymbol{\Sigma}(\boldsymbol{x}_{(s)}-\boldsymbol{x}_{(j)})\|_2}\right)^K \geq (1-\delta_n)^{1/(p-s)} \quad as\ n\to\infty
\end{aligned}
\tag{C.35}
$$

under the theorem conditions for some asymptotically negligible sequence $\delta_n > 0$. Then consider the bagging process, similar to C.30, according to theorem 6 in [13], we have

$$
\begin{aligned}
&\mathbb{P}\left(s\notin\mathcal{C}^m\cap\mathcal{S}^{m+1}\cap\mathcal{C}^{m+1^c}\right) \\
=&\mathbb{P}\left(\frac{1}{B_2}\sum_{b=1}^{B_2}\mathbb{1}_{\{\|\boldsymbol{G}_{ms}\|_2\neq\max_{j\in\mathcal{C}^m}\|\boldsymbol{G}_{mj}\|_2\}}\geq 1-p_r\right) \\
\leq&\frac{1}{1-p_r}\mathbb{E}\left[\frac{1}{B_2}\sum_{b=1}^{B_2}\mathbb{1}_{\{\|\boldsymbol{G}_{ms}\|_2\neq\max_{j\in\mathcal{C}^m}\|\boldsymbol{G}_{mj}\|_2\}}\right] \\
\leq&\frac{\delta_n}{1-p_r}\to 0 \quad as\ n\to\infty\ and\ B_2\to\infty
\end{aligned}
\tag{C.36}
$$

where the first inequality is by Markov's inequality, and the second inequality is by C.35. Therefore, the probability that variable s will not enter the model in the next step tends to zero, thus with probability tending to 1, all nonzero variables are selected in the regression set up.

Consider the classification case, we have the same as in regression but

$$\begin{aligned}\hat{\boldsymbol{\varepsilon}} =& \boldsymbol{\varepsilon} + \sigma' \left(\sum_{j \in \mathcal{S}^m \cap \{1,\ldots,s\}} \beta_j x_j + \xi \sum_{j \in \{1,\ldots,s\}/\mathcal{S}^m} \beta_j x_j \right) \sum_{j \in \{1,\ldots,s\}/\mathcal{S}^m} \beta_j x_j \\ & + O\left(K_n^2 \sqrt{\frac{\log(nK_n)}{n}} \right)\end{aligned}$$

by the proof of theorem 4.1, where $\boldsymbol{\varepsilon}$ is the theoretical error of Bernoulli distribution with their means. Here we no longer have the normality and have an extra term σ' which can be treated as constant in this step, but by the central limit theorem we have

$$\sqrt{n}(\boldsymbol{x}_{(s)} - \boldsymbol{x}_{(j)})^T \boldsymbol{\Sigma} \boldsymbol{\varepsilon}^T \Rightarrow N(0, \boldsymbol{V})$$

where $\boldsymbol{V}$ is bounded by assumption 4.4 and the fact that $\boldsymbol{\Sigma}$ is diagonal with the largest element less than 1. Feeding this back into C.35, we have

$$\mathbb{P}(\|\boldsymbol{G}_{ms}\|_2 > \|\boldsymbol{G}_{mj}\|_2) \geq (1 - \delta' n)^{1/(p-s)}$$

where δ'_n is greater than δ_n up to a factor of constant but still converges to zero as $n \to \infty$, under theorem conditions. Then similar to C.36, we have

$$\mathbb{P}\left(s \notin \mathcal{C}^m \cap \mathcal{S}^{m+1} \cap \mathcal{C}^{m+1^c} \right) \leq \frac{\delta'_n}{1 - p_r} \to 0 \quad as\ n \to \infty\ and\ B_2 \to \infty$$

This finishes the proof for the classification case. □

Proof of Theorem 4.3

Proof. In this subsection, we prove the estimation and prediction of regression and classification, respectively. In the regression set up, under assumption 4.2, we have

$$\boldsymbol{y} = f(\boldsymbol{x}) + \epsilon = f(\boldsymbol{x}_{\mathcal{S}}) + \epsilon$$

We have

$$\begin{aligned}
&\mathbb{P}\left(\mathbb{E}\int |f_n(\boldsymbol{x}_{\hat{\mathcal{S}}}) - f(\boldsymbol{x}_{\mathcal{S}})|^2 \mu(dx) \to 0\right) \\
=&\mathbb{P}\left(\mathbb{E}\int |f_n(\boldsymbol{x}_{\hat{\mathcal{S}}}) - f(\boldsymbol{x}_{\mathcal{S}})|^2 \mu(dx) \to 0 \middle| \hat{\mathcal{S}} = \mathcal{S}\right)\mathbb{P}\left(\hat{\mathcal{S}} = \mathcal{S}\right) \\
&+ \mathbb{P}\left(\mathbb{E}\int |f_n(\boldsymbol{x}_{\hat{\mathcal{S}}}) - f(\boldsymbol{x}_{\mathcal{S}})|^2 \mu(dx) \to 0 \middle| \hat{\mathcal{S}} \neq \mathcal{S}\right)\mathbb{P}\left(\hat{\mathcal{S}} \neq \mathcal{S}\right) \\
\geq&\mathbb{P}\left(\mathbb{E}\int |f_n(\boldsymbol{x}_{\hat{\mathcal{S}}}) - f(\boldsymbol{x}_{\mathcal{S}})|^2 \mu(dx) \to 0 \middle| \hat{\mathcal{S}} = \mathcal{S}\right)\mathbb{P}\left(\hat{\mathcal{S}} = \mathcal{S}\right) \\
=&\mathbb{P}\left(\mathbb{E}\int |f_n(\boldsymbol{x}_{\mathcal{S}}) - f(\boldsymbol{x}_{\mathcal{S}})|^2 \mu(dx) \to 0\right)\mathbb{P}\left(\hat{\mathcal{S}} = \mathcal{S}\right)
\end{aligned} \tag{C.37}$$

Observe that

$$\left|\hat{\boldsymbol{\beta}}\right| \leq |\boldsymbol{\theta}|_1 \leq K_n$$

According to [61], when we perform a neural network estimation on the true subset of variables, we have that the total error is bounded by the approximation error, which is bounded according to [43], plus the estimation error, which is bounded by the covering number, then by the packing number, then by the Vapnik-Chervonenkis dimension, and

finally by the space dimension, i.e.

$$\begin{aligned}&\mathbb{E}\int |f_n(\boldsymbol{x}_{\mathcal{S}}) - f(\boldsymbol{x}_{\mathcal{S}})|^2 \mu(dx)\\ =&O\left(L\sqrt{\frac{k_n}{n-1}}\right) + \delta_n\end{aligned} \tag{C.38}$$

where L is the Lipshitz continuity coefficient, k_n is the first hidden layer size, and by [61] δ_n satisfies

$$\mathbb{P}\left\{\sup \delta_n > \epsilon\right\} \leq 8\left(\frac{384K_n^2(k_n+1)}{\epsilon}\right)^{(2s+5)k_n+1} e^{-n\epsilon^2/128\cdot 2^4 K_n^4}$$

Under theorem assumptions, the probability above is summable, thus we have the first probability in C.37 converges to 1. On the other hand, by theorem 4.2, we have the second probability in C.37 converges to 1. Therefore, the result for regression set up is proved.

In the classification set up, similarly, we have

$$\begin{aligned}&\mathbb{P}\left(R(f_{n,\hat{\mathcal{S}}}) - R(f^*_{\mathcal{S}}) \to 0\right)\\ =&\mathbb{P}\left(R(f_{n,\hat{\mathcal{S}}}) - R(f^*_{\mathcal{S}}) \to 0 \middle| \hat{\mathcal{S}} = \mathcal{S}\right)\mathbb{P}\left(\hat{\mathcal{S}} = \mathcal{S}\right)\\ &+\mathbb{P}\left(R(f_{n,\hat{\mathcal{S}}}) - R(f^*_{\mathcal{S}}) \to 0 \middle| \hat{\mathcal{S}} \neq \mathcal{S}\right)\mathbb{P}\left(\hat{\mathcal{S}} \neq \mathcal{S}\right)\\ \geq&\mathbb{P}\left(R(f_{n,\hat{\mathcal{S}}}) - R(f^*_{\mathcal{S}}) \to 0 \middle| \hat{\mathcal{S}} = \mathcal{S}\right)\mathbb{P}\left(\hat{\mathcal{S}} = \mathcal{S}\right)\\ =&\mathbb{P}\left(R(f_{n,\mathcal{S}}) - R(f^*_{\mathcal{S}}) \to 0\right)\mathbb{P}\left(\hat{\mathcal{S}} = \mathcal{S}\right)\end{aligned} \tag{C.39}$$

By [37], we have

$$R(f_n) - R(f^*) \to 0 \quad as\ n \to \infty$$

and from theorem 4.2, we have the second probability in equation C.39 tends to 1. Combine these two results, the consistency of classification case is proved. □

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical programming*, 95(1):3–51, 2003.

[2] Umberto Amato, Anestis Antoniadis, and Italia De Feis. Additive model selection. *Statistical Methods & Applications*, 25(4):519–564, 2016.

[3] David F Andrews. Plots of high-dimensional data. *Biometrics*, pages 125–136, 1972.

[4] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

[5] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in neural information processing systems*, pages 41–48, 2007.

[6] Richard Arratia and Louis Gordon. Tutorial on large deviations for the binomial distribution. *Bulletin of mathematical biology*, 51(1):125–131, 1989.

[7] Sergey Bakin. *Adaptive regression and model selection in data mining problems*. PhD thesis, School of Mathematical Sciences, Australian National University, 1999.

[8] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

[9] A Belloni and V Chernozhukov. Least squares after model selection in high-dimensional sparse models. bernoulli 19 521–547. *Mathematical Reviews (MathSciNet): MR3037163 Digital Object Identifier: doi*, 10, 2013.

[10] Alexandre Belloni, Victor Chernozhukov, and Lie Wang. Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.

[11] Peter Bhlmann and Sara van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[12] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565, 2014.

[13] GÃŠrard Biau, Luc Devroye, and GÃĄbor Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(Sep):2015–2033, 2008.

[14] Peter J Bickel, Elizaveta Levina, et al. Some theory for fisher's linear discriminant function,naive bayes', and some alternatives when there are many more variables than observations. *Bernoulli*, 10(6):989–1010, 2004.

[15] Peter J Bickel, Ya'acov Ritov, and Alexandre B Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, pages 1705–1732, 2009.

[16] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

[17] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[18] Peter Bühlmann and Sara Van De Geer. *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011.

[19] Peter Bühlmann, Bin Yu, et al. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002.

[20] Emmanuel Candes, Terence Tao, et al. The dantzig selector: Statistical estimation when p is much larger than n. *The annals of Statistics*, 35(6):2313–2351, 2007.

[21] Timothy I Cannings and Richard J Samworth. Random-projection ensemble classification. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(4):959–1035, 2017.

[22] A Chatterjee and SN Lahiri. Rates of convergence of the adaptive lasso estimators to the oracle distribution and higher order refinements by the bootstrap. *The Annals of Statistics*, 41(3):1232–1259, 2013.

[23] Kamalika Chaudhuri and Sanjoy Dasgupta. Rates of convergence for nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 3437–3445, 2014.

[24] Jiahua Chen and Zehua Chen. Extended bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008.

[25] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[26] François Chollet et al. keras, 2015.

[27] Alexandra Chouldechova and Trevor Hastie. Generalized additive model selection. *arXiv preprint arXiv:1506.03850*, 2015.

[28] Charles K Chui, Xin Li, and Hrushikesh Narhar Mhaskar. Limitations of the approximation capabilities of neural networks with one hidden layer. *Advances in Computational Mathematics*, 5(1):233–243, 1996.

[29] CK Chui, Xin Li, and Hrushikesh Narhar Mhaskar. Neural networks for localized approximation. *Mathematics of Computation*, 63(208):607–623, 1994.

[30] Vasek Chvatal, Vaclav Chvatal, et al. *Linear programming*. Macmillan, 1983.

[31] Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

[32] John B Copas. Regression, prediction and shrinkage. *Journal of the Royal Statistical Society: Series B (Methodological)*, 45(3):311–335, 1983.

[33] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[34] Debraj Das, Karl Gregory, and SN Lahiri. Perturbation bootstrap in adaptive lasso. *arXiv preprint arXiv:1703.03165*, 2017.

[35] C De Boor. A practical guide to splines (revised ed.) springer. *New York*, 2001.

[36] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *Advances in neural information processing systems*, pages 666–674, 2011.

[37] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[38] David L Donoho. For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(6):797–829, 2006.

[39] David L Donoho and Jain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994.

[40] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[41] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[42] Paul HC Eilers and Brian D Marx. Flexible smoothing with b-splines and penalties. *Statistical science*, pages 89–102, 1996.

[43] Fenglei Fan, Jinjun Xiong, and Ge Wang. Universal approximation with quadratic deep networks. *Neural Networks*, 124:383–392, 2020.

[44] Jianqing Fan and Yingying Fan. High dimensional classification using features annealed independence rules. *Annals of statistics*, 36(6):2605, 2008.

[45] Jianqing Fan, Yang Feng, and Rui Song. Nonparametric independence screening in sparse ultra-high-dimensional additive models. *Journal of the American Statistical Association*, 106(494):544–557, 2011.

[46] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[47] Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):849–911, 2008.

[48] Jianqing Fan and Jinchi Lv. Nonconcave penalized likelihood with np-dimensionality. *IEEE Transactions on Information Theory*, 57(8):5467–5484, 2011.

[49] Jianqing Fan, Heng Peng, et al. Nonconcave penalized likelihood with a diverging number of parameters. *The Annals of Statistics*, 32(3):928–961, 2004.

[50] Jianqing Fan, Rui Song, et al. Sure independence screening in generalized linear models with np-dimensionality. *The Annals of Statistics*, 38(6):3567–3604, 2010.

[51] Qingliang Fan and Wei Zhong. Nonparametric additive instrumental variable estimator: A group shrinkage estimation perspective. *Journal of Business & Economic Statistics*, 36(3):388–399, 2018.

[52] Yingying Fan and Cheng Yong Tang. Tuning parameter selection in high dimensional penalized likelihood. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):531–552, 2013.

[53] Jean Feng and Noah Simon. Sparse-input neural networks for high-dimensional nonparametric regression and classification. *arXiv preprint arXiv:1711.07592*, 2017.

[54] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[55] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[56] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[57] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[58] Evarist Giné and Joel Zinn. Bootstrapping general empirical measures. *The Annals of Probability*, pages 851–869, 1990.

[59] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[60] Eitan Greenshtein, Ya'Acov Ritov, et al. Persistence in high-dimensional linear predictor selection and the virtue of overparametrization. *Bernoulli*, 10(6):971–988, 2004.

[61] László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

[62] Trevor Hastie and Robert Tibshirani. [generalized additive models]: Rejoinder. *Statist. Sci.*, 1(3):314–318, 08 1986.

[63] Trevor J Hastie. Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge, 2017.

[64] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[65] Jian Huang, Joel L Horowitz, and Fengrong Wei. Variable selection in nonparametric additive models. *Annals of statistics*, 38(4):2282, 2010.

[66] Bing'er Jiang, Tim O'Donnell, and Meghan Clayards. A deep neural network approach to investigate tone space in languages. *The Journal of the Acoustical Society of America*, 145(3):1913–1913, 2019.

[67] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[68] Ian T Jolliffe, Nickolay T Trendafilov, and Mudassir Uddin. A modified principal component technique based on the lasso. *Journal of computational and Graphical Statistics*, 12(3):531–547, 2003.

[69] Sungkyu Jung, J Stephen Marron, et al. Pca consistency in high dimension, low sample size context. *The Annals of Statistics*, 37(6B):4104–4130, 2009.

[70] Andre I Khuri. *Linear model methodology*. CRC Press, 2009.

[71] Hea-Jung Kim. On the ratio of two folded normal distributions. *Communications in Statistics-Theory and Methods*, 35(6):965–977, 2006.

[72] Stephen Cole Kleene. Representation of events in nerve nets and finite automata. Technical report, RAND PROJECT AIR FORCE SANTA MONICA CA, 1951.

[73] Keith Knight and Wenjiang Fu. Asymptotics for lasso-type estimators. *Annals of statistics*, pages 1356–1378, 2000.

[74] Matthieu Kowalski. Sparse regression using mixed norms. *Applied and Computational Harmonic Analysis*, 27(3):303–324, 2009.

[75] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[76] Neil D Lawrence. A unifying probabilistic perspective for spectral dimensionality reduction: Insights and new models. *Journal of Machine Learning Research*, 13(May):1609–1638, 2012.

[77] Gee Y Lee, Scott Manski, and Tapabrata Maiti. Actuarial applications of word embedding models. *ASTIN Bulletin: The Journal of the IAA*, 50(1):1–24, 2020.

[78] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[79] Yingjie Li and Taps Maiti. High dimensional discriminant analysis for spatially dependent data. 2018.

[80] You L Li, Jessica Ducey-Wysling, Aurélie D'Hondt, Dongwoon Hyun, Bhavik Patel, and Jeremy J Dahl. Vector flow imaging using a deep neural network. *The Journal of the Acoustical Society of America*, 146(4):2901–2902, 2019.

[81] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[82] Bo Liu, Ying Wei, Yu Zhang, and Qiang Yang. Deep neural networks for high dimension, low sample size data. In *IJCAI*, pages 2287–2293, 2017.

[83] Rong Liu, Lijian Yang, and Wolfgang K Härdle. Oracally efficient two-step estimation of generalized additive model. *Journal of the American Statistical Association*, 108(502):619–631, 2013.

[84] Yang Liu, Quanxue Gao, Xinbo Gao, and Ling Shao. $l_{\{2,1\}}$-norm discriminant manifold learning. *IEEE Access*, 6:40723–40734, 2018.

[85] Yufeng Liu and Yichao Wu. Variable selection via a combination of the l 0 and l 1 penalties. *Journal of Computational and Graphical Statistics*, 16(4):782–798, 2007.

[86] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. A provably efficient algorithm for training deep networks. *CoRR, vol. abs/1304.7045*, 2013.

[87] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

[88] Jinchi Lv and Yingying Fan. A unified approach to model selection and sparse recovery using regularized least squares. *The Annals of Statistics*, pages 3498–3528, 2009.

[89] Li Ma, Melba M Crawford, and Jinwen Tian. Local manifold learning-based k-nearest-neighbor for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 48(11):4099–4109, 2010.

[90] Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[91] Giampiero Marra and Simon N Wood. Practical variable selection for generalized additive models. *Computational Statistics & Data Analysis*, 55(7):2372–2387, 2011.

[92] Rahul Mazumder, Peter Radchenko, and Antoine Dedieu. Subset selection with shrinkage: Sparse linear modeling when the snr is low. *arXiv preprint arXiv:1708.03288*, 2017.

[93] Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[94] Lukas Meier, Sara Van de Geer, Peter Bühlmann, et al. High-dimensional additive modeling. *The Annals of Statistics*, 37(6B):3779–3821, 2009.

[95] Nicolai Meinshausen, Peter Bühlmann, et al. High-dimensional graphs and variable selection with the lasso. *The annals of statistics*, 34(3):1436–1462, 2006.

[96] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[97] Mitsunori Mizumachi and Maya Origuchi. Superdirective non-linear beamforming with deep neural network. *The Journal of the Acoustical Society of America*, 140(4):3167–3167, 2016.

[98] Siddhartha Nandy, Chae Young Lim, and Tapabrata Maiti. Additive model building for spatial regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):779–800, 2017.

[99] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[100] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[101] Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

[102] Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of machine learning research*, 3(Mar):1333–1356, 2003.

[103] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

[104] Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

[105] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[106] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[107] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[108] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[109] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[110] LL Schumaker. Spline functions: basic theory. 1981. *John Wiley&Sons, New York*, 1981.

[111] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[112] Uri Shaham, Alexander Cloninger, and Ronald R Coifman. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537–557, 2018.

[113] Jonathan W Siegel and Jinchao Xu. On the approximation properties of neural networks. *arXiv preprint arXiv:1904.02311*, 2019.

[114] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

[115] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[116] Donald F Specht et al. A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576, 1991.

[117] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[118] Nicolas Städler, Peter Bühlmann, and Sara Van de Geer. Rejoinder: l1-penalization for mixture regression models. *Test*, 19(2):209–256, 2010.

[119] Eduardo D Stonag. Critical points for least-squares problems involving certain analytic functions, with applications to sigmoidal nets. *Advances in Computational Mathematics*, 5(1):245–268, 1996.

[120] Charles J Stone. Additive regression and other nonparametric models. *The annals of Statistics*, pages 689–705, 1985.

[121] Charles J Stone. The dimensionality reduction principle for generalized additive models. *The Annals of Statistics*, pages 590–606, 1986.

[122] Wesley Tansey and James G Scott. A fast and flexible algorithm for the graph-fused lasso. *arXiv preprint arXiv:1505.06475*, 2015.

[123] Gülşen Taşkin and Melba M Crawford. An out-of-sample extension to manifold learning via meta-modeling. *IEEE Transactions on Image Processing*, 28(10):5227–5237, 2019.

[124] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[125] Ambuj Tewari, Pradeep K Ravikumar, and Inderjit S Dhillon. Greedy algorithms for structurally constrained high dimensional problems. In *Advances in Neural Information Processing Systems*, pages 882–890, 2011.

[126] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[127] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[128] Ryan J Tibshirani. A general framework for fast stagewise algorithms. *The Journal of Machine Learning Research*, 16(1):2543–2588, 2015.

[129] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[130] Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[131] Wen-Jen Tsay, Cliff J Huang, Tsu-Tan Fu, and I-Lin Ho. A simple closed-form approximation for the cumulative distribution function of the composite error of stochastic frontier models. *Journal of Productivity Analysis*, 39(3):259–269, 2013.

[132] Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

[133] John W Tukey. Non-parametric estimation ii. statistically equivalent blocks and tolerance regions–the continuous case. *The Annals of Mathematical Statistics*, pages 529–539, 1947.

[134] Gerhard Tutz and Harald Binder. Generalized additive modeling with implicit variable selection by likelihood-based boosting. *Biometrics*, 62(4):961–971, 2006.

[135] Sara A Van de Geer. High-dimensional generalized linear models and the lasso. *The Annals of Statistics*, pages 614–645, 2008.

[136] AW van der Vaart and J. Wellner. *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer Series in Statistics. Springer, 1996.

[137] Hansheng Wang, Bo Li, and Chenlei Leng. Shrinkage tuning parameter selection with a diverging number of parameters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):671–683, 2009.

[138] Mingqiu Wang and Guo-Liang Tian. Adaptive group lasso for high-dimensional generalized linear models. *Statistical Papers*, 60(5):1469–1486, 2019.

[139] Rongrong Wang and Xiaopeng Zhang. Capacity preserving mapping for high-dimensional data visualization. *arXiv preprint arXiv:1909.13322*, 2019.

[140] Fengrong Wei and Jian Huang. Consistent group selection in high-dimensional linear regression. *Bernoulli: official journal of the Bernoulli Society for Mathematical Statistics and Probability*, 16(4):1369, 2010.

[141] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[142] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[143] Tong Tong Wu, Kenneth Lange, et al. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

[144] Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P Xing, and Masashi Sugiyama. High-dimensional feature selection by feature-wise kernelized lasso. *Neural computation*, 26(1):185–207, 2014.

[145] Kaixu Yang and Tapabrata Maiti. Ultra high-dimensional generalized additive model: consistency and tuning parameter selection. *Technical report, Michigan State University*, 2018.

[146] Kaixu Yang and Tapabrata Maiti. Statistical aspects of high-dimensional sparse artificial neural network models. *Machine learning and knowledge extraction*, 2(1):1–19, 2020.

[147] Yi Yang and Hui Zou. A fast unified algorithm for solving group-lasso penalize learning problems. *Statistics and Computing*, 25(6):1129–1141, 2015.

[148] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[149] Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *The Annals of Statistics*, pages 1567–1594, 2008.

[150] Nancy R Zhang and David O Siegmund. A modified bayes information criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63(1):22–32, 2007.

[151] Yiyun Zhang, Runze Li, and Chih-Ling Tsai. Regularization parameter selections via generalized information criterion. *Journal of the American Statistical Association*, 105(489):312–323, 2010.

[152] Peng Zhao and Bin Yu. On model selection consistency of lasso, 2006.

[153] S Zhou, X Shen, DA Wolfe, et al. Local asymptotics for regression splines and confidence regions. *The annals of statistics*, 26(5):1760–1782, 1998.

[154] Bo Zhu, Jeremiah Z Liu, Stephen F Cauley, Bruce R Rosen, and Matthew S Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487–492, 2018.

[155] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

[156] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

[157] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.