INFERENTIAL GANS AND DEEP FEATURE SELECTION WITH APPLICATIONS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Yao Chen

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Xiao Wang, Chair

Department of Statistics

Dr. Chuanhai Liu

Department of Statistics

Dr. Lingsong Zhang

Department of Statistics

Dr. Anindya Bhadra

Department of Statistics

Approved by:

Dr. Jun Xie

Graduate Chair of Department of Statistics

I would like to dedicate this dissertation to my parents, Ling Chen and Lei Chen, and to my wife Ran Bi, for their endless support and unconditional love.

ACKNOWLEDGMENTS

First of all, I would like to express my most sincere gratitude to my advisor Dr. Xiao Wang, who constantly inspires me to explore the fantasy of machine learning and always provides insightful suggestions on my research. I would also like to thank Dr. Chuanhai Liu, Dr. Lingsong Zhang, and Dr. Anindya Bhadra for their guidance as my advisory committee members. In addition, special thanks goes to Dr. Hao Zhang and Dr. Jun Xie for their priceless support for my study, research, and teaching in the past years. I would like to express my gratitude to my collaborators (in collaborating time favor): Qingyi Gao, Dr. Faming Liang, Yonghan Jung, Dr. Vida Abedi, Dr. Ramin Zand, Marvi Bikak, Dr. Mohammad Adibuzzaman, Dr. Linglong Kong, Dr. Hongtu Zhu. I would like to thank my friends from the Department of Statistics at Purdue University, who always support and encourage me in their own ways. Last but not the least, I am grateful to my parents and wife for their unconditional love and support throughout my life.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Chen, Yao Ph.D., Purdue University, August 2020. Inferential GANs and Deep Feature Selection with Applications. Major Professor: Xiao Wang Professor.

Deep nueral networks (DNNs) have become popular due to their predictive power and flexibility in model fitting. In unsupervised learning, variational autoencoders (VAEs) and generative adverarial networks (GANs) are two most popular and successful generative models. How to provide a unifying framework combining the best of VAEs and GANs in a principled way is a challenging task. In supervised learning, the demand for high-dimensional data analysis has grown significantly, especially in the applications of social networking, bioinformatics, and neuroscience. How to simultaneously approximate the true underlying nonlinear system and identify relevant features based on high-dimensional data (typically with the sample size smaller than the dimension, a.k.a. small-n-large-p) is another challenging task. In this dissertation, we have provided satisfactory answers for these two challenges. In addition, we have illustrated some promising applications using modern machine learning methods.

In the first chapter, we introduce a novel inferential Wasserstein GAN (iWGAN) model, which is a principled framework to fuse auto-encoders and WGANs. GANs have been impactful on many problems and applications but suffer from unstable training. The Wasserstein GAN (WGAN) leverages the Wasserstein distance to avoid the caveats in the minmax two-player training of GANs but has other defects such as mode collapse and lack of metric to detect the convergence. The iWGAN model jointly learns an encoder network and a generator network motivated by the iterative primal dual optimization process. The encoder network maps the observed samples to the latent space and the generator network maps the samples from the latent space to the data space. We establish the generalization error bound of iWGANs to theoretically justify the performance of iWGANs. We further provide a rigorous probabilistic interpretation of our model under the framework of maximum likelihood estimation. The iWGAN, with a clear stopping criteria, has many advantages over other autoencoder GANs. The empirical experiments show that the iWGAN greatly mitigates the symptom of mode collapse, speeds up the convergence, and is able to provide a measurement of quality check for each individual sample. We illustrate the ability of iWGANs by obtaining a competitive and stable performance with state-of-the-art for benchmark datasets.

In the second chapter, we present a general framework for high-dimensional nonlinear variable selection using deep neural networks under the framework of supervised learning. The network architecture includes both a selection layer and approximation layers. The problem can be cast as a sparsity-constrained optimization with a sparse parameter in the selection layer and other parameters in the approximation layers. This problem is challenging due to the sparse constraint and the nonconvex optimization. We propose a novel algorithm, called Deep Feature Selection, to estimate both the sparse parameter and the other parameters. Theoretically, we establish the algorithm convergence and the selection consistency when the objective function has a Generalized Stable Restricted Hessian. This result provides theoretical justifications of our method and generalizes known results for high-dimensional linear variable selection. Simulations and real data analysis are conducted to demonstrate the superior performance of our method.

In the third chapter, we develop a novel methodology to classify the electrocardiograms (ECGs) to normal, atrial fibrillation and other cardiac dysrhythmias as defined by the Physionet Challenge 2017. More specifically, we use piecewise linear splines for the feature selection and a gradient boosting algorithm for the classifier. In the algorithm, the ECG waveform is fitted by a piecewise linear spline, and morphological features related to the piecewise linear spline coefficients are extracted. XGBoost is used to classify the morphological coefficients and heart rate variability features. The performance of the algorithm was evaluated by the PhysioNet Challenge database

(3658 ECGs classified by experts). Our algorithm achieves an average F_1 score of 81% for a 10-fold cross validation and also achieved 81% for F_1 score on the independent testing set. This score is similar to the top 9th score (81%) in the official phase of the Physionet Challenge 2017.

In the fourth chapter, we introduce a novel region-selection penalty in the framework of image-on-scalar regression to impose sparsity of pixel values and extract active regions simultaneously. This method helps identify regions of interest (ROI) associated with certain disease, which has a great impact on public health.Our penalty combines the Smoothly Clipped Absolute Deviation (SCAD) regularization, enforcing sparsity, and the SCAD of total variation (TV) regularization, enforcing spatial contiguity, into one group, which segments contiguous spatial regions against zero-valued background. Efficient algorithm is based on the alternative direction method of multipliers (ADMM) which decomposes the non-convex problem into two iterative optimization problems with explicit solutions. Another virtue of the proposed method is that a divide and conquer learning algorithm is developed, thereby allowing scaling to large images. Several examples are presented and the experimental results are compared with other state-of-the-art approaches.

1. INTRODUCTION

Human has long dreamed of creating machines that can think. Scientists today are applying machine learning and deep learning to create machines that help us do certain task. In cases like facial recognition, machines have already outperformed humans. In particular, deep learning technology, a specific branch of machine learning based on artificial neural networks, has made a great progress during past decade. Different neural network models have been applied to both supervised and unsupervised tasks. For example, fully connected neural networks (FNN) have performed well on regular classification and regression tasks, convolution neural networks (CNN) have shown great advantages in image processing, and recurrent neural networks (RNN) have been widely used in sequential tasks. Besides supervised learning models, deep generative architectures like variational autoencoders (VAEs) and generative adversarial nets (GANs) have achieved great successes in unsupervised learning. Deep Learning has been implement in various artificial intelligence tasks, like driverless car, language translation, etc. In the field of healthcare, deep learning and machine learning are also playing more and more important roles. In this dissertation, we introduce several novel ideas in deep learning and their implement on medical care data. More detailed background knowledge, notation and model structures, in deep learning and healthcare terms are introduced in chapter 2.

1.1 Deep Generative Models

Generative models are one of the breakthroughs of modern deep learning technology. To train a generative model we first collect a large amount of data in some domain (e.g., images, sentences, or sounds, etc.) and then train a model to generate data whose distribution is similar to it. The intuition behind this approach follows a famous quote from Richard Feynman: ”What I cannot create, I do not understand.”

VAEs and GANs are two well developed frameworks in this field. Especially in image generating tasks, both generate either nice inferential model or state-of-art samples, but also have their own disadvantages. VAEs framework includes encoder, which maps the real sample to the low-dimensional latent space, and decoder, which reconstruct the sample from the latent variable. The generative model is trained by minimize the reconstruction error of samples plus a regularization terms. VAEs have built nice theoretical results, but tend to generate blurry images in practice. GANs architecture consists of a generator and a discriminator, which generator generates samples while discriminator tries to distinguish them from the real samples. Essentially, GANs is playing a minmax two-player problem. Empirically, GANs can generate vivid and sharp images compare with VAEs, but it is also well known that GANs suffers from unstable training and mode collapse problem. During the training, GANs usually require a delicate balance between generator and discriminator. One comment thought would be choosing different metric of difference between two distributions, as the architecture offers such flexibility. In [1], Wasserstein distance is applied to calculate the distance between two distribution instead of the Jensen-Shannon divergence in vanilla GANs. Wasserstein GAN (WGAN) has improved the stability training of GANs. However, challenges still remain as WGAN also suffers mode collapse problem and lacks metrics to monitor the convergence of training. How to build a framework that fuses the best of VAEs and GANs, i.e. stably generating vivid, diverse images and detecting the convergence of the training simultaneously remain to be a challenging problem.

In chapter 3, the deep generative models are studied and a novel framework, Inferential WGAN (iWGAN), that unifying the best of VAEs and GANs has been proposed. The iWGAN architecture includes encoder network as well as generator under WGAN framework. We establish theoretical generalization error bound of iWGANs and rigorous probabilistic interpretation of the model. We also raised an algorithm based on the model with clear stopping criteria.

1.2 High-dimensional Data Analysis

High-dimensional data analysis has always been a important and challenging question in the field of statistics and machine learning. How to approximate the true underlying system and select the relevant features with high-dimensional data, i.e. sample size smaller than the variable dimension, is challenging. Deep neural networks have been widely used in solving supervised learning problems. Deep feedforward networks, and their derivatives, like convolutional neural networks and recurrent neural networks, have significantly improved the performance of many tasks. They even outperformed human in tasks such as image classification and disease diagnosis. Most successful cases using deep neural networks require enormously large dataset, in order to obtain a good estimation of millions of parameters in the model. However, collecting data can be difficult for many domains in practice, particularly the aread such as social networking, bioinformatics, and neuroscience. Whether deep neural networks can performance well in high-dimensional data analysis becomes both interesting and challenging.

In chapter 4, a general framework for high-dimensional nonlinear variable selection via deep neural networks is proposed. The framework includes a selection layer in other neural network structure. The variable selection is achieved by sparse control of the selection layer's parameter. A novel algorithm is raised to estimate the sparse parameter and other parameters. We also build the theoretical results on the approximation error of such framework and the selection consistency.

1.3 Machine Learning for Medical Diagnosis

With the rapid development of varieties of machine learning frameworks, state-of-the-art results spring up like mushrooms. How to take advantages of them for real world problems and benefit humankind in the area of healthcare has become the next challenge for researchers. As statisticians, we are very curious whether these models can be applied to solve healthcare difficulties or improve the efficiency of healthcare system. For example, can we incorporate human knowledge into machine and further help improve the diagnosis accuracy and efficiency? Many more questions are waiting to be answered and machine learning techniques are highly expected to play an important role.

Although machine learning has demonstrated its power on many tasks, the state-of-the-art results are mostly based on huge cleaned data or well known datasets that are carefully studied. However, in reality, collecting data to answer scientific questions is hard and people often facing a raw, unaligned, and incomplete data. How to start from zero to build a pipeline to solve the real problem is always challenging and often requires lots of cross discipline cooperation.

In chapter 5, we demonstrate a case study of machine learning in medical care problem. We developed a new automatic approach of classifying the electrocardiograms (ECGs) to normal, atrial fibrillation, other cardiac disease along with noise signal. Our method achieves top 9th score in the Physionet Challenge 2017.

1.4 Local Region Regression

We are now experiencing an age of data explosion. Tremendous data with different data types are collected. It is easy to collect both image data and other scalar variable data in the same field. For example, fMRI image and other variables such as cognitive scores can be easily collected at the same time when patients enter the hospital. Building connection between them has been a significant research activity in the area of neuroscience. One of the urgent tasks is to discover the Region of Interest (ROI) of image data that may help identify subjects with high risk. Machine learning techniques with penalized optimization is often used. However, requirements of spatially heterogeneous smoothness and local region selection are the most significant challenges to be faced by researchers. How to achieve sparsity of the selection and smoothness of selected region simultaneously poses a lot of challenges to modern machine learning methods.

In chapter 6, we introduce a novel region-selection penalty in the framework of image-on-scalar regression to impose sparsity of pixel values and extract active regions simultaneously. We define a novel penalty, SCAD2TV, which combines the SCAD regularization, enforcing sparsity, and the SCAD of TV regularization, enforcing spatial contiguity, into one group, which segments contiguous spatial regions against zero-valued background. This method helps identify regions of interest (ROI) associated with certain disease, which has a great impact on public health.

2. BACKGROUND

In this chapter, preliminary knowledge and terms of deep learning and medical care that will be used in latter chapters are defined.

2.1 Deep Learning

Deep learning has a long history and many aspirations. Several approaches have been proposed that have yet to entirely bear fruit. Several ambitious goals have yet to be realized. Modern deep learning provides a very powerful framework for both supervised learning and unsupervised learning. We introduce several basic frameworks which are used in latter chapters in the following.

2.1.1 Deep feedforward network

Deep feedforward networks, also often called feedforward neural networks, or multilayer perceptrons (MLPs), are the most essential deep learning models. The models are defined to approximate some function f^*. For example, the general prediction problem:

$$y^* = f^*(\mathbf{x}) \tag{2.1}$$

$$y = t(y^*) \tag{2.2}$$

where t is a fixed function related to the prediction problem, and f^* is an unknown function. For regression, t is the identity mapping. For classification, t is the function to maps y^* to different number of classes. A feedforward network defines a mapping $y = f(\mathbf{x}; \theta)$ and learns the value of the parameters θ that result in the best approximation of the true function f^*.

Deep feedforward networks are represented by composing together many different functions. For example, $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ are three functions that connected in a chain to form the feedforward model $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. The number of chain functions are called the depth of the model. The final chain function or layer of the feedforward network is called the output layer, and the functions before the output layer is called hidden layers. The output of each hidden layer is vector-valued and the dimensionality of these hidden layer is called the width of the model. Figure 2.1 is example of feedforward neural networks.

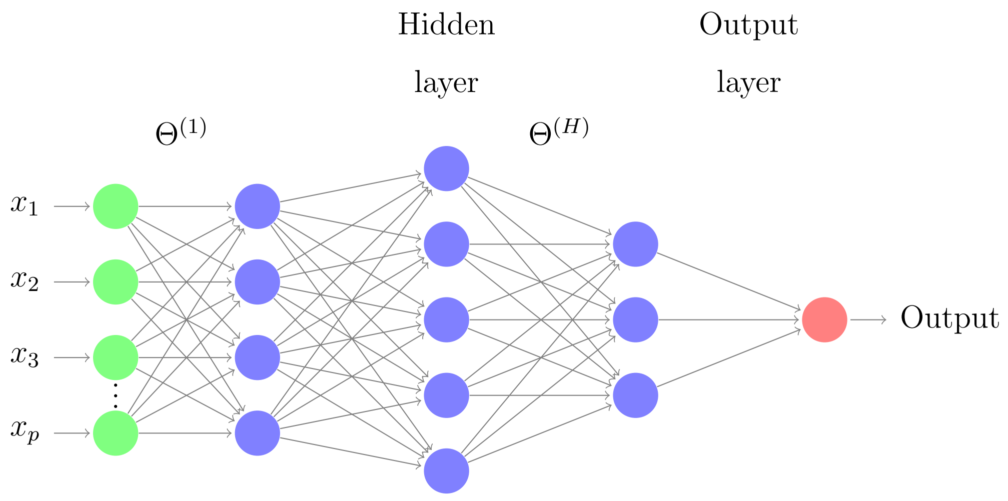


Fig. 2.1. Feedforward Neural Networks Architecture

The function represented by Figure 2.1 can be written as

$$f(\boldsymbol{x}) = T_{H+1} \circ \sigma \circ T_H \circ \sigma \circ \cdots \sigma \circ T_1 \circ (\boldsymbol{x}), \tag{2.3}$$

where $T_i(u) = \Theta^{(i)}u + b^{(i)}$, $i = 1, \ldots, H+1$, are affine transformations with unknown parameters $(\Theta^{(i)}, b^{(i)})$ with $\Theta^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$ and $b^{(i)} \in \mathbb{R}^{d_i}$, and $\sigma(\cdot)$ is the activation function. Popular choice of activation functions includes rectified linear unit (ReLU), sigmoid, and tanh.

2.1.2 Variational AutoEncoders (VAEs)

Variational AutoEncoders (VAEs) have become the most popular generative models. The major task of generative models is studying the distribution, $P_{\boldsymbol{x}}$, of data $\boldsymbol{x} \in \mathcal{X}$. In order to be able to sample from $P_{\boldsymbol{x}}$, we first define a latent variable $\boldsymbol{z}$, which follows distribution that is easy to sample from. Then we define a decoder $p_\theta(x|z)$ to model the conditional distribution $P_{\boldsymbol{x}|\boldsymbol{z}}$. If we can estimate the conditional distribution well, then we can sample $\boldsymbol{x}$ from given latent variable $\boldsymbol{z}$ and approximate $P_{\boldsymbol{x}}$ by:

$$p_\theta(x) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z}) \cdot p(\boldsymbol{z}) d\boldsymbol{z} \tag{2.4}$$

we can then estimate θ by maximize the likelihood function of given data. However, Equation 2.4 is intractable, which means the $P_{z|x} = p_\theta(\boldsymbol{z}|\boldsymbol{x})p\boldsymbol{z}/p_\theta(x)$ is also intractable. Markov Chain Monte Carlo (MCMC) is one solution for estimating the integration, but it is very inefficient. So we define an encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to approximate $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. Then the log likelihood function can be rewritten as:

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}^{(i)}) &= \mathbb{E}_{\boldsymbol{z}\sim q_\phi(z|x^{(i)})}\Big[\log p_\theta(x^{(i)})\Big] \\
&= \mathbb{E}_{\boldsymbol{z}}[\log \frac{p_\theta(x^{(i)}|z)p(\boldsymbol{z})}{p_\theta(z|x^{(i)})} \frac{q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})}] \\
&= \mathbb{E}_{\boldsymbol{z}}[\log p_\theta(x^{(i)}|z)] - D_{KL}(q_\phi(z|x^{(i)}) \| p(z)) \\
&+ D_{KL}(q_\phi(z|x^{(i)}) \| p_\theta(z|x^{(i)}))
\end{aligned} \tag{2.5}$$

The first two terms of Equation 2.5 have tractable lower bound, which we can take gradient and optimize. The first term is always greater than 0. Putting together, we can maximize the lower bound of the likelihood function.

VAEs are principled approach in generative models. It allows we do inference of $q(z|x)$, which can be useful feature representation in other tasks. However, instead of optimize the likelihood function, we maximize its lower bound. This makes this approach has intrinsic bias even with growing sample size. Also, VAEs generate blurrier and low quality samples compared to the state-of-the-art generated by Generative Adversarial Nets (GANs).

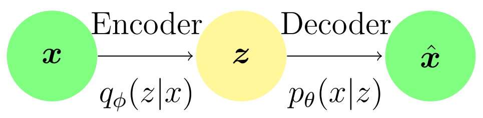


Fig. 2.2. Autoencoder Architecture

2.1.3 Generative Adversarial Nets (GANs)

Generative adversarial nets (GANs) are a class of machine learning framework that is used to generate samples from target distribution or learning the target distribution. GANs consists of two networks: generative model G and discriminative model D. The generative model G learns to map a latent variable $\boldsymbol{z}$, typically follows a standard multivariate normal distribution or a uniform distribution, to the target distribution where the sample $\boldsymbol{x}$ come from. While the discriminative model D are trained to distinguish true sample $\boldsymbol{x}$ and the generated samples $G(\boldsymbol{z})$. The training object of G is to increase the error rate of D and the goal of D is to identify true data from synthesized ones. The success of G is aimed to fool arbitrary D, in which sense, we conclude $G(\boldsymbol{z})$ follows the true distribution of $\boldsymbol{x}$.

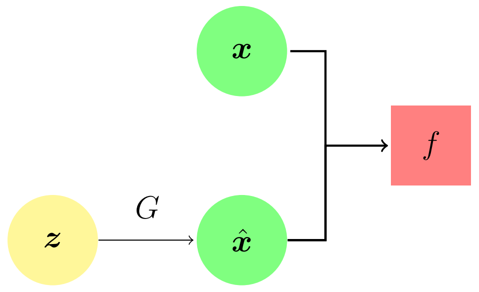


Fig. 2.3. Generative Adversarial Nets Architecture

The general loss function of GANs is defined in Equation 2.6, where ϕ_1 and ϕ_2 can be flexibly defined. The loss function can be also regarded as the measurement of two distribution $P_{\boldsymbol{x}}$ and $P_{G\boldsymbol{z}}$. With different choice of measurement, the ϕ_1 and ϕ_2 will be defined correspondingly. For example, the vanilla GANs choose Jensen-Shannon Divergence with $\phi_1(x) = \log x$ and $\phi_2(x) = -\log(1-x)$. Wasserstein GANs (WGANs) use Wasserstein distance to measure the difference of $P_{\boldsymbol{x}}$ and $P_{G(\boldsymbol{z})}$, so $\phi_1(x) = \phi_2(x) = x$ is adopted with constrain of f to be 1-Lipschitz. In practice, GANs generate state-of-art samples, but they always suffer unstable training and mode collapse.

$$\min_G \max_D V(G, D) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\phi_1((f(\boldsymbol{x})))] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}} \boldsymbol{z}}[\phi_2(f(G(\boldsymbol{z})))] \tag{2.6}$$

2.2 Healthcare

With the rapid development of Machine Learning, its applications in medical care has also become popular. AI for medicine has been wide used in diagnosis and interpretation of medical data. In latter chapters, we will implement two applications of machine learning on different disease. In this section, the background knowledge of Alzheimer's Disease and Atrial Fibrillation is introduced.

2.2.1 Alzheimer's Disease

Alzheimer's disease (AD), also simply called as Alzheimer's is a chronic neurodegenerative disease. The most common symptom is difficulty in remembering recent events, and problems with language, disorientation, mood swings, loss of motivation, not managing self-care and behavioural issues may come after. Patients often gradually withdraw from family and society, lose bodily function and are lead to death. In 2015, there were approximately 29.8 million people worldwide with AD. It most often begins in people over 65 years of age. AD has become one of the top 10 leading causes of death in the United States.

2.2.2 Atrial Fibrillation

Atrial Fibrillation(AF) is an abnormal heart rhythm characterized by rapid and irregular heartbeat. AF, usually associated with significant mortality and morbidity, is the most common sustained cardiac arrhythmia, occurring in 1-2% of the general population [2, 3]. At least 2.7 million Americans are living with AF and, more than 12 million Europeans and North Americans are estimated to suffer from AF [3, 4]. The incidence of AF increases with age, from less than 0.5% at 40-50 years, to 5-15% for 80 years of age [5]. Its prevalence will likely triple in the next 30-50 years, particularly, in the United States and other western countries with aging population demographics [6]. This growth may also be influenced by extended survival outcomes for patients with congestive heart failure (CHF), valvular heart disease, and coronary artery disease, as AF is common among patients with other forms of structural heart disease.

3. INFERENTIAL WASSERSTEIN GENERATIVE ADVERSARIAL NETS

3.1 Introduction

One of the goals of generative modeling is to match the model distribution $P_\theta(x)$ with parameters θ to the true data distribution P_X for a random variable $X \in \mathcal{X}$. For latent variable models, the data point X is generated from a latent variable $Z \in \mathcal{Z}$ through a conditional distribution $P(X|Z)$. Here $\mathcal{X}$ denotes the support for P_X and $\mathcal{Z}$ denotes the support for P_Z. In this chapter, we consider models with $Z \sim \mathcal{N}(0, I)$. There has been a surge of research on deep generative networks in recent years and the literature is too vast to summarize here [7–9]. These models have provided a powerful framework for modeling complex high dimensional datasets. We start introducing two main approaches for generative modeling. The first one is called *variational auto-encoders* (VAEs) [7], which use variational inference to learn a model by maximizing the lower bound of the likelihood function. VAEs have elegant theoretical foundations but the drawback is that they tend to produce blurry images. The second approach is called *generative adversarial networks* (GANs) [8], which learn a model by using a powerful discriminator to distinguish between real data points and generative data points. GANs produce more visually realistic images but suffer from the unstable training and the mode collapse problem. Although there are many variants of generative models trying to take advantages of both VAEs and GANs [10, 11], to the best of our knowledge, the model which provides a unifying framework combining the best of VAEs and GANs in a principled way is yet to be discovered.

3.1.1 Related work

The generative model is to learn a mapping, denoted by G, from $\mathcal{Z}$ to $\mathcal{X}$ to approximate the conditional distribution $P_G(X|Z)$ of the data point $X \in \mathcal{X}$ given latent code $Z \in \mathcal{Z}$. We consider the deterministic mapping G in this work. Both the vanilla GAN [8] and the Wasserstein GAN (WGAN) [1] can be viewed as minimizing certain divergence between the data distribution P_X and the generative model distribution $P_{G(Z)}$. For example, the Jensen-Shannon (JS) divergence is implicitly used in vanilla GANs [8]. The 1-Wasserstein distance between P_X and $P_{G(Z)}$, denoted by $W_1(P_X, P_{G(Z)})$, is employed in WGANs. Empirical experiments suggest that the Wasserstein distance is a more sensible measure to differentiate probability measures supported in low-dimensional manifold. In terms of training, it turns out that it is hard or even impossible to compute these standard divergences in probability, especially when P_X is unknown and $P_{G(Z)}$ is parameterized by deep neural networks (DNNs). The training of GANs is converted into playing a game between two competing networks: the *generator* and the *discriminator*. The generator is to fool the discriminator and the discriminator is to distinguish between true data samples and generated samples. Instead, the training of WGANs is to study its dual problem because of the elegant form of Kantorovich-Rubinstein duality [12]. Analogous to GANs, the discriminator is now a real-valued 1-Lipschitz function. Many techniques such as weight clipping [1] and gradient penalty [13] are used to enforce the Lipschitz constraint. For autoencoder GANs, [14] first introduced the VAE-GAN, which is a hybrid of VAEs and GANs. The VAE-GAN uses a GAN discriminator to replace a VAE's decoder to learn the loss function. The motivation behind this modification is that VAEs tend to produce blurry outputs during the reconstruction phase. More recent VAE-GAN variants, such as Adversarial Generator Encoders (AGE) [15] and Auto-encoding GANs (α-GAN) [11], use a separate encoder to stabilize GAN training. The main difference with standard GANs is that, besides the generator G, there is an encoder $Q : \mathcal{X} \to \mathcal{Z}$ which maps the data points into the latent space. This encoder is to approximate the conditional distribution $Q(Z|X)$ of the latent variable Z given the data point X. Other encoder-decoder GANs are introduced in Adversarially Learned Inference (ALI) [16] and Bidirectional Generative Adversarial Networks (BiGAN) [17]. The objective of both ALI and BiGAN is to match two joint distributions under the framework of vanilla GANs, the joint distribution of $(X, Q(X))$ and the joint distribution of $(G(Z), Z)$. When the algorithm achieves equilibrium, these two joint distributions roughly match. It is expected to obtain more meaningful latent codes by $Q(X)$, and this should improve the quality of the generator as well. Adversarial Variational Bayes (AVB) [18] presented a more flexible latent distribution to train Variational Autoencoders. [19] provided new interpretations of GANs and VAEs and revealed strong connections between them which are linked by the classic wake-sleep algorithm. Regarding the optimization perspectives of GANs, [20, 21] studied duality-based methods for improving algorithm performance for training. Primal-dual Wasserstein GANs (PD-GANs) are introduced in [22], which proposed a new penalty term whose evaluation samples are obtained from the encoder Q. [23] developed a convex duality framework to address the case when the discriminator is constrained into a smaller class. [24] developed an evaluation metric to detect the non-convergence behavior of vanilla GANs, which is the duality gap defined as the difference between the primal and the dual objective functions.

3.1.2 Our Contributions

Although there are many interesting works on autoencoder GANs, it remains unclear what the principles are underlying the fusion of auto-encoders and GANs. For example, do there even exist these two mappings, the encoder Q and the decoder G, for any high-dimensional random variable X, such that $Q(X)$ has the same distribution as Z and $G(Z)$ has the same distribution as X? Is there any probabilistic interpretation such as the maximum likelihood principle on autoencoder GANs? What is the generalization performance of autoencoder GANs? We introduce inferential WGANs (iWGANs), which provide satisfying answers for these questions. We will mainly focus on the 1-Wasserstein distance, instead of the Kullback-Leibler divergence. We borrow the strength from both the primal and the dual problems and demonstrate the synergistic effect between these two optimizations. The encoder component tends out to be a natural consequence from our algorithm. Furthermore, the iWGAN has a rigorous probabilistic interpretation under the maximum likelihood principle, and our learning algorithm is equivalent to the maximum likelihood estimation when our model is defined as an energy-based model based on an autoencoder. Our main contributions are listed as follows: (a) We propose a novel framework, called iWGAN, to learn both an encoder and a decoder simultaneously. We prove the existence of meaningful encoder and decoder, establish an equivalence between WGAN and iWGAN, and develop a generalization error bound for iWGAN. (b) We establish a rigorous probability interpretation for iWGANs and our training process is exactly the same as the maximum likelihood estimation. As a byproduct, this interpretation allows us to perform the quality check at the individual sample level. (c) We demonstrate the natural use of the duality gap as a measure of convergence for iWGANs, and show its effectiveness for various numerical settings. Our experiments do not experience any mode collapse problem.

The rest of the chapter is organized as follows. Section 2 presents the new iWGAN framework, and its extension to general inferential f-GANs. Section 3 establishes the generalization error bound and introduces the algorithm for iWGANs. The probabilistic interpretation and the connection with the maximum likelihood estimation are introduced in Section 4. Extensive numerical experiments are demonstrated in Section 5 to show the advantages of the iWGAN framework. Proofs of theorems and additional numerical results are provided in the Appendix.

3.2 The iWGAN Model

The autoencoder generative model consists of two parts: an encoder Q and a generator G. The encoder Q maps a data sample $x \in \mathcal{X}$ to a latent variable $z \in \mathcal{Z}$, and the generator G takes a latent variable $z \in \mathcal{Z}$ to produce a sample $G(z)$. In general, the autoencoder generative model should satisfy the following three conditions *simultaneously*: (a) The generator can generate images which have a similar distribution with observed images, i.e., the distribution of $G(Z)$ is similar to that of P_X; (b) The encoder can produce meaningful encodings in the latent space, i.e., $Q(X)$ has a similar distribution with Z; (c) The reconstruction errors of this model based on these meaningful encodings are small, i.e., the difference between X and $G(Q(X))$ is small.

We emphasize that the benefit of using an autoencoder is to encourage the model to better represent *all* the data it is trained with, so that it discourages mode-collapse. We first show that, for any distribution residing on a smooth manifold, there always exists an encoder Q^* which guarantees meaningful encodings and exists a generator G^* which generates samples with the same distribution as data points by using these meaningful codes.

Theorem 3.2.1 *Consider a continuous random variable $X \in \mathcal{X}$, where $\mathcal{X}$ is a d-dimensional smooth Riemannian manifold. Then, there exist two mappings $Q^* : \mathcal{X} \to \mathbb{R}^p$ and $G^* : \mathbb{R}^p \to \mathcal{X}$, with $p = \max\{d(d+5)/2, d(d+3)/2+5\}$, such that $Q^*(X)$ follows a multivariate normal distribution with zero mean and identity covariance matrix and $G^* \circ Q^*$ is an identity mapping, i.e., $X = G^*(Q^*(X))$.*

Learning Q^* and G^* from the data points is a challenging task. Consider a general f-GAN model [25] here. Let $h : \mathbb{R} \to (-\infty, \infty]$ be a convex function with $h(1) = 0$. The f-GAN defines the f-divergence between the data distribution P_X and the generative model distribution $P_{G(Z)}$ for the generator G as:

$$\mathrm{GAN}_h(P_X, P_{G(Z)}) = \sup_{f \in \mathcal{F}} \Big[\mathbb{E}_X \left\{f(X)\right\} - \mathbb{E}_Z \left\{h^*(f(G(Z))\right\}\Big],$$

where $h^*(x) = \sup_y\{x \cdot y - h(y)\}$ is the convex conjugate of h and $\mathcal{F}$ is an arbitrary class of functions $f : \mathcal{X} \to \mathbb{R}$. If $h(x) = x\log(x) - (x+1)\log(x+1) - 2\log 2$, f-GAN recovers the original vanilla GAN [8]. If $h(x) = 0$ when $x = 1$ and $h(x) = \infty$ otherwise, we have $h^*(x) = x$. With the property that $\mathcal{F}$ is 1-Lipschitz function class, the f-GAN turns to be the WGAN.

For ease of presentation, we illustrate our methodology by mainly focusing on Wasserstein distance and the inferential WGAN (iWGAN) model. The extension to general inferential f-GANs (ifGANs) is straightforward and will be presented in Section 3.2.3.

3.2.1 iWGAN

Recall that the 1-Wasserstein distance between P_X and $P_{G(Z)}$ is defined as

$$W_1(P_X, P_{G(Z)}) = \inf_{\pi \in \Pi(P_X, P_Z)} \mathbb{E}_{(X,Z)\sim\pi}\big\|X - G(Z)\big\|, \tag{3.1}$$

where $\|\cdot\|$ represents the L_2-norm and $\Pi(P_X, P_Z)$ is a set of all joint distributions of (X, Z) with marginal measures P_X and P_Z, respectively. The main difficulty in (3.1) is to find the optimal coupling π and this is a constrained optimization because the joint distribution π needs to match these two marginal distributions P_X and P_Z.

Based on the Kantorovich-Rubinstein duality [12], the WGAN studies the 1-Wasserstein distance (3.1) through its dual format

$$W_1(P_X, P_{G(Z)}) = \sup_{f\in\mathcal{F}} \Big[\mathbb{E}_{X\sim P_X}\big\{f(X)\big\} - \mathbb{E}_{Z\sim P_Z}\big\{f(G(Z))\big\}\Big], \tag{3.2}$$

where $\mathcal{F}$ is the set of all bounded 1-Lipschitz functions. This is also a constrained optimization due to the Lipschitz constraint on f such that $f(x) - f(y) \le \|x - y\|$ for all $x, y \in \mathcal{X}$. Weight clipping [1] and gradient penalty [13] have been used to satisfy the constraint of Lipschitz continuity. The experiment of [1] showed that the WGAN can avoid the problem of gradient vanishment. However, the WGAN does not produce meaningful encodings and many experiments still display the problem of mode collapse [1, 13].

On the other hand, the Wasserstein Autoencoder (WAE) [10], after introducing an encoder $Q : \mathcal{X} \to \mathcal{Z}$ to approximate the conditional distribution of Z given X, minimizes the reconstruction error $\inf_{Q\in\mathcal{Q}} \mathbb{E}_X \big\|X - G(Q(X))\big\|$, where $\mathcal{Q}$ is a set of encoder mappings whose elements satisfies $P_{Q(X)} = P_Z$. The penalty, such as $\mathcal{D}(P_{Q(X)}, P_Z)$, is added to the objective to satisfy this constraint, where $\mathcal{D}$ is an arbitrary divergence between $P_{Q(X)}$ and P_Z. The WAE can produce meaningful encodings and have controlled reconstruction error. However, the WAE defines a generative model in an implicit way and does not model the generator through $G(Z)$ with $Z \sim P_Z$ directly.

To take the advantages of both WGAN and WAE, we propose a new autoencoder GAN model, called iWGAN, which defines the divergence between P_X and $P_{G(Z)}$ by

$$\overline{W}_1(P_X, P_{G(Z)}) = \inf_{Q\in\mathcal{Q}} \sup_{f\in\mathcal{F}} \Big[\mathbb{E}_X\|X - G(Q(X))\| + \mathbb{E}_X\big\{f(G(Q(X)))\big\} - \mathbb{E}_Z\big\{f(G(Z))\big\}\Big]. \tag{3.3}$$

The term $\|X - G(Q(X))\|$ can be treated as the autoencoder reconstruction error as well as a loss to match the distributions between X and $G(Q(X))$. We note that the L_1-norm $\|\cdot\|_1$ has been used for the reconstruction term by α-GAN [11] and CycleGAN [26]. Another term $\mathbb{E}_{X\sim P_X}\{f(G(Q(X)))\} - \mathbb{E}_{Z\sim P_Z}\{f(G(Z))\}$ can be treated as a loss for the generator as well as a loss to match the distribution between $G(Q(X))$ and $G(Z)$. We emphasize that this term is different with the objective function of the WGAN in (3.2). The properties of (3.3) will be discussed in Theorem 3.2.2. The primal and dual explanation of (3.3) will be presented in Section 3.2.2.

Furthermore, it is challenging for practitioners to determine when to stop training GANs. Most of the GAN algorithms do not provide any explicit standard for the convergence of the model. However, the measure of convergence for iWGAN becomes very natural and we use the duality gap as the measure. The duality gap for $(\widetilde{G}, \widetilde{Q}, \widetilde{f})$ is defined as

$$\mathrm{DualGap}(\widetilde{G}, \widetilde{Q}, \widetilde{f}) = \sup_{f\in\mathcal{F}} L(\widetilde{G}, \widetilde{Q}, f) - \inf_{G\in\mathcal{G}, Q\in\mathcal{Q}} L(G, Q, \widetilde{f}), \tag{3.4}$$

where $L(G, Q, f)$ is

$$L(G, Q, f) = \mathbb{E}_X\|X - G(Q(X))\| + \mathbb{E}_X\{f(G(Q(X)))\} - \mathbb{E}_Z\{f(G(Z))\}.$$

Theorem 3.2.2 *The iWGAN objective (3.3) is equivalent to*

$$\overline{W}_1(P_X, P_{G(Z)}) = \inf_{Q \in \mathcal{Q}} \Big\{ W_1(P_X, P_{G(Q(X))}) + W_1(P_{G(Q(X))}, P_{G(Z)}) \Big\}. \tag{3.5}$$

Therefore, $W_1(P_X, P_{G(Z)}) \le \overline{W}_1(P_X, P_{G(Z)})$. If there exists a $Q^* \in \mathcal{Q}$ such that $Q^*(X)$ has the same distribution with Z, then $W_1(P_X, P_{G(Z)}) = \overline{W}_1(P_X, P_{G(Z)})$. Let $(\widetilde{Q}, \widetilde{G}, \widetilde{f})$ be a fixed solution and assume that the encoder, generator, and discriminator all have enough capacities. Then the duality gap is larger than $W_1(P_X, P_{\widetilde{G}(\widetilde{Q}(X))}) + W_1(P_{\widetilde{G}(\widetilde{Q}(X))}, P_{\widetilde{G}(Z)})$. Moreover, if $\widetilde{G}$ outputs the same distribution as X and $\widetilde{Q}$ outputs the same distribution as Z, both the duality gap and $\overline{W}_1(P_X, P_{\widetilde{G}(Z)})$ are zeros and $X = \widetilde{G}(\widetilde{Q}(X))$ for $X \sim P_X$.

According to Theorem 3.2.2, the iWGAN objective is in general the upper bound of $W_1(P_X, P_{G(Z)})$. However, this upper bound is tight. When the space $\mathcal{Q}$ includes a special encoder Q^* such that $Q^*(X)$ has the same distribution as Z, the iWGAN objective is exactly the same as $W_1(P_X, P_{G(Z)})$. Theorem 3.2.2 also provides an appealing property from a practical point of view. The values of both the duality gap and $\overline{W}_1(P_X, P_{\widetilde{G}(Z)})$ give us a natural criteria to justify the algorithm convergence.

3.2.2 A Primal-Dual Explanation

We explain the iWGAN objective function (3.3) from the view of primal and dual problems. Note that both the primal problem (3.1) and the dual problem (3.2) are constrained optimization problems. First, the primal variable f for the dual problem (3.2) is also a dual variable for the primal problem (3.1). From the Lagrange multiplier perspective, we can write the primal problem (3.1) such that

$$\begin{aligned}
&\inf_\pi \mathbb{E}_\pi \Big\| X - G(Z) \Big\| + \int_x f(x)\Big(p_X(x) - \int_z \pi(x,z)dz\Big)dx - \int_z f(G(z))\Big(p_Z(z) - \int_x \pi(x,z)dx\Big)dz \\
&= \inf_{Q \in \mathcal{Q}} \mathbb{E}_X \Big\{ \|X - G(Q(X))\| + f(G(Q(X))) \Big\} - \mathbb{E}_Z \big\{ f(G(Z)) \big\}.
\end{aligned}$$

Second, the primal variable π for the primal problem (3.1) is also a dual variable for the dual problem (3.2). We can write the dual problem (3.2) such that

$$\sup_{f\in\mathcal{F}} \mathbb{E}_X\{f(X)\} - \mathbb{E}_Z\{f(G(Z))\} - \int_{\mathcal{X}\times\mathcal{Z}} \pi(x,z)\Big(f(x) - f(G(z)) - \|x - G(z)\|\Big)dxdz$$
$$= \sup_{f\in\mathcal{F}} \mathbb{E}_X\Big\{\|X - G(Q(X))\| + f(G(Q(X)))\Big\} - \mathbb{E}_Z\{f(G(Z))\}.$$

When we solve primal and dual problems iteratively, this turns out to be exactly the same as our iWGAN algorithm.

In addition, the optimal value of the primal problem (3.1) satisfies

$$\inf_{Q\in\mathcal{Q}} \sup_{f\in\mathcal{F}} \mathbb{E}_X\Big\{\|X - G(Q(X))\| + f(G(Q(X)))\Big\} - \mathbb{E}_Z\{f(G(Z))\},$$

and the optimal value of the dual problem (3.2) satisfies

$$\sup_{f\in\mathcal{F}} \inf_{Q\in\mathcal{Q}} \mathbb{E}_X\Big\{\|X - G(Q(X))\| + f(G(Q(X)))\Big\} - \mathbb{E}_Z\{f(G(Z))\}.$$

The difference between the optimal primal and dual values is exactly the duality gap in (3.4).

3.2.3 Extension to f-GANs

This framework can be easily extended to other types of GANs. Assume that $\mathcal{F}$ is the 1-Lipschitz function class. We extend the iWGAN framework to the inferential f-GAN (ifGAN) framework. Define the ifGAN objective function as follows:

$$\overline{W}_{1,h}(P_X, P_{G(Z)}) = \inf_{Q\in\mathcal{Q}} \sup_{f\in\mathcal{F}} \Big[\mathbb{E}_X\|X - G(Q(X))\| + \mathbb{E}_X\{f(G(Q(X)))\} - \mathbb{E}_Z\{h^*(f(G(Z)))\}\Big]. \tag{3.6}$$

Following this definition, we have

$$\overline{W}_{1,h}(P_X, P_{G(Z)}) = \inf_{Q\in\mathcal{Q}} \Big\{W_1(P_X, P_{G(Q(X))}) + \mathrm{GAN}_h(P_{G(Q(X))}, P_{G(Z)})\Big\}.$$

We show $\mathrm{GAN}_h(P_X, P_{G(Z)}) \le \overline{W}_{1,h}(P_X, P_{G(Z)})$. This is because

$$\mathrm{GAN}_h(P_X, P_{G(Z)}) = \sup_{f\in\mathcal{F}} \mathbb{E}_X\left\{f(X)\right\} - \mathbb{E}_Z\left\{h^*(f(G(Z))\right\}$$
$$\le \inf_{Q\in\mathcal{Q}} \left[\sup_{f\in\mathcal{F}} \mathbb{E}_X\left\{f(X)\right\} - \mathbb{E}_X\left\{f(G(Q(X)))\right\} + \sup_{f\in\mathcal{F}} \mathbb{E}_X\left\{f(G(Q(X)))\right\} - \mathbb{E}_Z\left\{h^*(f(G(Z))\right\}\right]$$
$$= \overline{W}_{1,h}(P_X, P_{G(Z)}).$$

This indicates that the ifGAN objective (3.6) is an upper bound of the f-GAN objective.

3.3 Generalization Error Bound and the Algorithm

Suppose that we observe n samples $\{x_1, \ldots, x_n\}$. In practice, we minimize the empirical version, denoted by $\widehat{\overline{W}}_1(P_X, P_{G(Z)})$, of $\overline{W}_1(P_X, P_{G(Z)})$ to learn both the encoder and the generator, where,

$$\widehat{\overline{W}}_1(P_X, P_{G(Z)}) = \inf_{Q \in \mathcal{Q}} \sup_{f \in \mathcal{F}} \Big[\hat{\mathbb{E}}_{obs} \|x - G(Q(x))\| + \hat{\mathbb{E}}_{obs}\big\{f(G(Q(x)))\big\} - \hat{\mathbb{E}}_z\big\{f(G(z))\big\} \Big]. \tag{3.7}$$

Here $\hat{\mathbb{E}}_{obs}\{\cdot\}$ denotes the empirical average on the observed data $\{x_i\}$ and $\hat{\mathbb{E}}_z$ denotes the empirical average on a random sample of standard normal random variables. Before we present the details of the algorithm, we first establish the generalization error bound for iWGANs in this section.

In the context of supervised learning, generalization error is defined as the gap between the empirical risk and the expected risk. The empirical risk is corresponding to the training error, and the expected risk is corresponding to the testing error. Mathematically, the difference between the expected risk and the empirical risk, i.e. the generalization error, is a measure of how accurately an algorithm is able to predict outcome values for previously unseen data. However, in the context of GANs, neither the training error nor the test error is well defined. But we can define the generalization error in a similar way. Explicitly, we define the "training error" as $\widehat{\overline{W}}_1(P_X, P_{G(Z)})$ in (3.7), which is minimized based on observed samples. Define the "test error" as $W_1(P_X, P_{G(Z)})$ in (3.1), which is the true 1-Wasserstein distance between P_X and $P_{G(Z)}$. The generalization error for iWGANs is defined as the gap between these two "errors". In other words, for a iWGAN model with the parameter (G, Q, f), the generalization error is defined as $\widehat{\overline{W}}_1(P_X, P_{G(Z)}) - W_1(P_X, P_{G(Z)})$. For discussions of generalization performance of classical GANs, see [27] and [28].

Theorem 3.3.1 *Given a generator $G \in \mathcal{G}$, and n samples $(x_1, \ldots, x_n)$ from $\mathcal{X} = \{x : \|x\| \leq B\}$, with probability at least $1 - \delta$ for any $\delta \in (0, 1)$, we have*

$$W_1(P_X, P_{G(Z)}) \leq \widehat{\widehat{W}}_1(P_X, P_{G(Z)}) + 2\widehat{\mathfrak{R}}_n(\mathcal{F}) + 3B\sqrt{\frac{2}{n}\log\left(\frac{2}{\delta}\right)}, \tag{3.8}$$

where $\widehat{\mathfrak{R}}_n(\mathcal{F}) = \mathbb{E}_\epsilon \left\{\sup_{f\in\mathcal{F}} n^{-1} \sum_{i=1}^{n} \epsilon_i f(x_i)\right\}$ is the empirical Rademacher complexity of the 1-Lipschitz function set $\mathcal{F}$, in which ϵ_i is the Rademacher variable.

For a fixed generator G, Theorem 3.3.1 holds uniformly for any discriminator $f \in \mathcal{F}$. It indicates that the 1-Wasserstein distance between P_X and $P_{G(Z)}$ can be dominantly upper bounded by the empirical $\widehat{\widehat{W}}_1(P_X, P_{G(Z)})$ and Rademacher complexity of $\mathcal{F}$. Since $\widehat{\widehat{W}}_1(P_X, P_{G(Z)}) \leq \widehat{W}_1(P_X, P_{G(Q(X))}) + \widehat{W}_1(P_{G(Q(X))}, P_{G(Z)})$ for any $Q \in \mathcal{Q}$, the capacity of $\mathcal{Q}$ determines the value of $\widehat{\widehat{W}}_1(P_X, P_{G(Z)})$.

In learning theory, Rademacher complexity, named after Hans Rademacher, measures richness of a class of real-valued functions with respect to a probability distribution. There are several existing results on the empirical Rademacher complexity of neural networks. For example, when $\mathcal{F}$ is a set of 1-Lipschitz neural networks, we can apply the conclusion from [29] to $\widehat{\mathfrak{R}}_n(\mathcal{F})$, which produces an upper bound scaling as $\mathcal{O}(B\sqrt{L^3/n})$. Here L denotes the depth of network $f \in \mathcal{F}$. Similar upper bound with an order of $\mathcal{O}(B\sqrt{Ld^2/n})$ can be obtained by utilizing the results from [30], where d is the width of the network.

Next, we introduce the details of the algorithm. Our target is to solve the following optimization problem:

$$\min_{G\in\mathcal{G},Q\in\mathcal{Q}} \max_{f\in\mathcal{F}} \hat{\mathbb{E}}_{obs}\|x - G(Q(x))\| + \hat{\mathbb{E}}_{obs}\{f(G(Q(x)))\} - \hat{\mathbb{E}}_z\{f(G(z))\}$$
$$+\lambda_1 J_1(f) + \lambda_2 J_2(Q), \tag{3.9}$$

where $J_1(f)$ and $J_2(Q)$ are regularization terms for f and Q respectively. We approximate G, Q, f by three neural networks with pre-specified architectures. The details of the neural network architectures are provided in the Appendix E. Since f is assumed to be 1-Lipschitz, we adopt the gradient penalty defined as $J_1(f) =$

Algorithm 1 The training algorithm of iWGAN

```
While DualGap $> \epsilon_1$ or $L(G^i, Q^i, f^i) > \epsilon_2$
for $t = 1, ..., n_{critic}$ do
    Sample real data $x^i \sim P_X$, latent variable $z^i \sim P_Z$ and a random number $\epsilon \sim U[0,1]$
    $\hat{x}^i \leftarrow \epsilon x^i + (1-\epsilon) G^i(z^i)$
    Calculate $L^i = L(G^i, Q^i, f^i | x^i, z^i)$ and gradient of $-L^i$
    Update $f$ by Adam: $f^{i+1} \leftarrow Adam(-\nabla_f L^i)$
    where for $f^i$,
    $-\nabla_f L^i = \nabla_f \frac{1}{n} \sum_{k=1}^n \Big( f^i(G^i(z_k^i)) - f^i(G^i(Q^i(x_k^i))) + \lambda_1 (\|\nabla_{\hat{x}^i} f^i(\hat{x}^i)\|_2 - 1)^2 \Big)$
end for
for $t = 1, ..., n_{critic}$ do
    Sample real data $x'^i \sim P_X$, latent variable $z'^i \sim P_Z$
    Calculate $L'^i = L(G^i, Q^i, f^{i+1} | x'^i, z'^i)$ and gradient of $L'^i$
    Update $G$, $Q$ by Adam: $G^{i+1}, Q^{i+1} \leftarrow Adam(\nabla_{G,Q} L'^i)$
    where for $G^i$, $Q^i$,
    $\nabla_{G,Q} L'^i = \nabla_{G,Q} \frac{1}{n} \sum_{k=1}^n (\|x_k^i - G^i(Q^i(x_k^i))\| + f^{i+1}(G^i(Q^i(x_k^i))) - f^{i+1}(G^i(z_k^i)))$
        $+ \frac{\lambda_2}{n(n-1)} \sum_{l \neq j} k(z_l^i, z_j^i) + \frac{\lambda_2}{n(n-1)} \sum_{l \neq j} k(Q(x_l^i), Q(x_j^i)) - \frac{2\lambda_2}{n^2} \sum_{l,j} k(z_l^i, Q(x_j^i))$
end for
```

$\mathbb{E}_X\big\{(\|\nabla_X f(X)\|_2 - 1)^2\big\}$ in [13] to enforce the 1-Lipschitz constraint on $f \in \mathcal{F}$. Furthermore, since we expect $Q(X)$ follows approximately standard normal, we use the maximum mean discrepancy (MMD) penalty [31], denoted by $J_2(Q) = \text{MMD}_k(P_{Q(X)}, P_Z)$, to enforce $Q(X)$ to converge to P_Z, where k is a kernel function. We have adopted the stochastic gradient descent to estimate the unknown parameters in neural networks. The optimization (3.9) consists of two tuning parameters λ_1 and λ_2 and we select the optimal tuning parameters by grid search using cross validation. The details of the algorithm are presented in Algorithm 1.

3.4 Probabilistic Interpretation and the MLE

The iWGAN has proposed an efficient framework to stably and automatically estimate both the encoder and the generator. In this section, we provide a probabilistic interpretation of the iWGAN under the framework of maximum likelihood estimation.

Maximum likelihood estimator (MLE) is a fundamental statistical framework for learning models from data. However, for complex models, MLE can be computationally prohibitive due to the intractable normalization constant. MCMC has been used to approximate the intractable likelihood function but do not work efficiently in practice. The iWGAN can be treated as an adaptive method for the MLE training, which not only provides computational advantages but also allows us to generate more realistic-looking images. Furthermore, this probabilistic interpretation enables other novel applications such as image quality checking and outlier detection.

Let X denote the image. Define the density of X by an energy-based model based on an autoencoder [32, 33]:

$$p(x|\theta) = \exp\big(- \|x - G_\theta(Q_\theta(x))\| - V(\theta)\big),$$

where

$$V(\theta) = \log \int \exp(-\|x - G_\theta(Q_\theta(x))\|)dx,$$

and θ is the unknown parameter and $V(\theta)$ is the log normalization constant. The major difficulty for the likelihood inference is due to the intractable function $V(\theta)$. Suppose that we have the observed data $\{x_i : i = 1, \dots, n\}$. The log-likelihood function of θ is $\ell(\theta) = n^{-1}\sum_{i=1}^n \log\ p(x_i|\theta)$, whose gradient is

$$\nabla_\theta \ell(\theta) = -\hat{\mathbb{E}}_{obs}\big\{\partial_\theta \|x - G_\theta(Q_\theta(x))\|\big\} + \mathbb{E}_\theta\big\{\partial_\theta \|x - G_\theta(Q_\theta(x))\|\big\}, \tag{3.10}$$

where $\hat{\mathbb{E}}_{obs}[\cdot]$ denotes the empirical average on the observed data $\{x_i\}$ and $\mathbb{E}_\theta[\cdot]$ denotes the expectation under model $p(x|\theta)$. The key computational obstacle lies in the approximations of the model expectation $\mathbb{E}_\theta[\cdot]$. To address this problem, we propose a novel dual approximation for this expectation. By Theorem 5.10 of [12], there exists an optimal f^* such that

$$\mathbb{P}_{(x,y)\sim\pi}\big(f^*(y) - f^*(x) = \|y - x\|\big) = 1 \tag{3.11}$$

for the optimal coupling π. Therefore, there exists a f^* such that $f^*(x) - f^*(G_\theta(Q_\theta(x))) = \|x - G_\theta(Q_\theta(x))\|$ with probability one with respect to the distribution of x. Since $\mathbb{E}_\theta$ in (3.10) is taken under the current estimated θ and we also require G_θ to be a good generator and the distributions of $G_\theta(z)$ and $G_\theta(Q_\theta(x))$ to be close, we approximate $\|x - G_\theta(Q_\theta(x))\|$ by $f^*(G_\theta(z)) - f^*(G_\theta(Q_\theta(x)))$. We replace $\|x - G_\theta(Q_\theta(x))\|$ in the second term of (3.10) by $f^*(G_\theta(z)) - f^*(G_\theta(Q_\theta(x)))$, yielding a gradient update for θ of form $\theta \leftarrow \theta + \epsilon\hat{\nabla}_\theta\ell(\theta)$, where

$$\hat{\nabla}_\theta\ell(\theta) = -\hat{\mathbb{E}}_{obs}\big\{\partial_\theta\big\|x - G_\theta(Q_\theta(x))\big\|\big\} + \mathbb{E}_\theta\big\{\partial_\theta f^*(G_\theta(z)) - \partial_\theta f^*(G_\theta(Q_\theta(x))\big\}. \tag{3.12}$$

Here f^* needs to be learned and is solved by the corresponding dual problem at each iteration. We approximate f^* by a network f_η with an unknown parameter η, yielding a gradient update for η of form

$$\eta \leftarrow \eta + \epsilon\ \mathbb{E}_\theta\big\{\partial_\eta f_\eta(G_\theta(z)) - \partial_\eta f_\eta(G_\theta(Q_\theta(x))\big\}. \tag{3.13}$$

The advantage of using expectations in (3.12) and (3.13) is that we can evaluate them by using only marginal distributions of z and x. The above iterative updating process is exactly the same as in Algorithm 1. Therefore, the training of iWGAN is to seek the MLE. This probabilistic interpretation provides a novel alternative method to tackle problems with the intractable normalization constant in latent variable models. The MLE gradient update of $p(x|\theta)$ decreases the energy of the training data and increases the dual objective. Compare with original GANs or WGANs, our method gives much faster convergence and simultaneously provides a higher quality generated images.

The probabilistic modeling opens a door for many interesting applications. Next, we present a completely new approach for determining a highest density region (HDR) estimate for the distribution of X. What makes HDR distinct from other statistical methods is that it finds the smallest region, denoted by $U(\alpha)$, in the high dimensional space with a given probability coverage $1-\alpha$, i.e., $\mathbb{P}(X \in U(\alpha)) = 1-\alpha$. We can use $U(\alpha)$ to assess each individual sample quality. Note that commonly used inception scores (IS) and Fréchet inception distances (FID) are to measure the whole sample quality, not at the individual sample level. More introductions of IS and FID are given in next section. Let $\hat{\theta}$ be the MLE. The density ratio at x_1 and x_2 is

$$\frac{p(x_1|\hat{\theta})}{p(x_2|\hat{\theta})} = \exp\big\{-(\|x_1 - G_{\hat{\theta}}(Q_{\hat{\theta}}(x_1))\| - \|x_2 - G_{\hat{\theta}}(Q_{\hat{\theta}}(x_2))\|)\big\}.$$

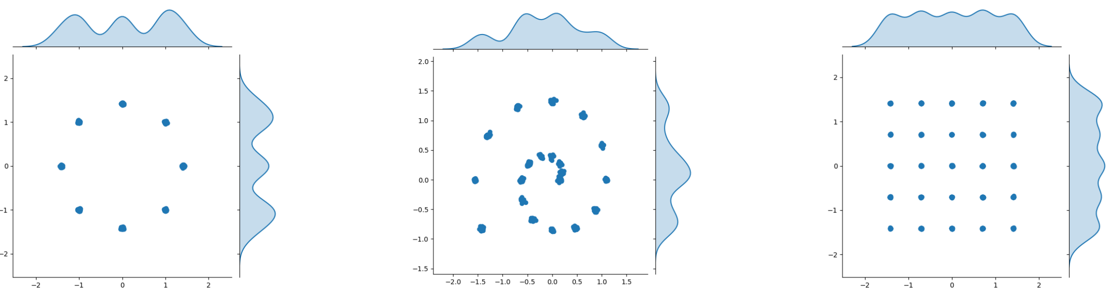
The smaller the reconstruction error is, the larger the density value is. We can define the HDR for x through the HDR for the reconstruction error $e_x := \|x - G_{\hat{\theta}}(Q_{\hat{\theta}}(x))\|$, which is simple because it is a one-dimensional problem. Let $\tilde{U}(\alpha)$ be the HDR for e_x. Then, $U(\alpha) = \{x : e_x \in \tilde{U}(\alpha)\}$. Here $Q_{\hat{\theta}}(U(\alpha))$ defines the corresponding region in the latent space, which can be used to generate better quality samples.

3.5 Experimental Results

The goal of our numerical experiments is to demonstrate that iWGAN can achieve the following three objectives simultaneously: high-quality generative samples, meaningful latent codes, and small reconstruction errors. We also compare iWGAN with other well-known GAN models such as Wasserstein GAN with gradient penalty (WGAN-GP) [13], Wasserstein Autoencoder (WAE) [10], and Adversarial Learning Inference (ALI) [16] to illustrate a competitive and stable performance with state-of-the-art for benchmark datasets.

3.5.1 Mixture of Gaussians

We first train our iWGAN model on three datasets from the mixture of Gaussians with an increasing difficulty shown in the Figure 3.1: a). RING: a mixture of 8 Gaussians with means $\{(2\cdot\cos\frac{2\pi\cdot i}{8}, 2\cdot\sin\frac{2\pi\cdot i}{8})|i=0,\dots 7\}$ and standard deviation



(a) RING (b) Swiss Roll (c) GRID

Fig. 3.1. Mixture of Gaussians

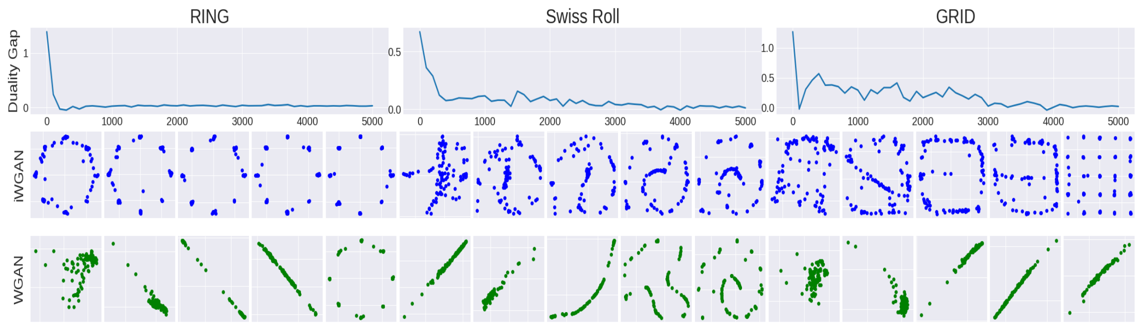
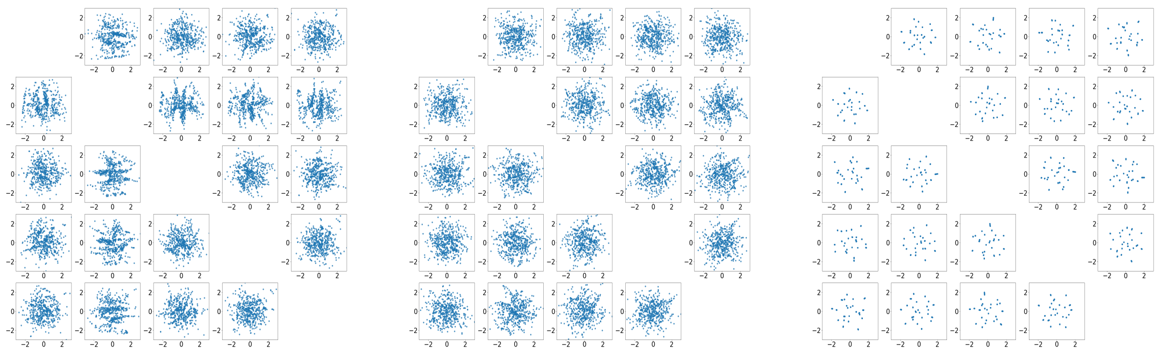


Fig. 3.2. Duality gap, generated samples from iWGAN, and generated samples from WGAN-GP on mixture of Gaussians

0.02, b). SPIRAL: a mixture of 20 Gaussians with means $\{(0.1+0.1\cdot\frac{2\pi}{20}\cdot\cos\frac{2\pi i}{20}, 0.1+0.1\cdot\frac{2\pi}{20}\cdot\sin\frac{2\pi i}{20})|i=0,\ldots,19\}$ and standard deviation 0.02 and c). GRID: a mixture of 25 Gaussians with means $\{(2\cdot i, 2\cdot j)|i=-2,-1,\ldots,2, j=-2,-1,\ldots,2\}$ and standard deviation 0.02. As the true data distributions are known, this setting allows for tracking of convergence and mode dropping.

Duality gap and convergence We illustrate that as the duality gap converges to 0, our model converges to the generated samples from the true distribution. We keep track of the generated samples using $G(z)$ and record the duality gap at each iteration to check the corresponding generated samples. We compare our method with WGAN-GP in Figure 3.2. Both methods adopt the same structure, learning rate, number of critical steps, and other hyper-parameters.

Figure 3.2 shows that iWGAN converges quickly in terms of both the duality gap and the true distributions learning. Duality gap has also been a good indicator of whether the model has generated the desired distribution. When comparing with the WGAN model, the iWGAN surpasses the performance of the WGAN-GP at very early stage and avoids the appearance of mode collapse.



(a) RING (b) Swiss Roll (c) GRID

Fig. 3.3. Latent Space of Mixture of Gaussians

Latent space We choose the latent distribution to be a five dimensional standard normal distribution $Z \sim N(0, I_5)$. During the training each batch size is chosen to be 256. After training, the distribution of $Q(X)$ is expected to be close to the distribution of Z. To demonstrate the latent distribution visually, we plot the $Q(X)_i$ against $Q(X)_j$ for all $i \neq j$ in Figure 3.3. We can tell that the joint distribution of any two dimensions of $Q(X)$ is close to a bivariate normal distribution.

Mode collapse We investigate the mode collapse problem for the iWGAN. If we draw two random samples in the latent space $z_1, z_2 \sim N(0, I_5)$, the interpolation, $G(\lambda z_1 + (1 - \lambda)z_2)$, $0 \leq \lambda \leq 1$, should fall around the mode to represent a reasonable sample. In Figure 3.4, we select $\lambda \in \{0, 0.05, 0.10, \ldots, 0.95, 1.0\}$, and do interpolations on two random samples. We repeat this procedure several times on 3 datasets as demonstrated in Figure 3.4. No matter where the interpolations start and end, the interpolations would fall around the modes other than the locations where true distribution has a low density. There may still be some samples that appears in the

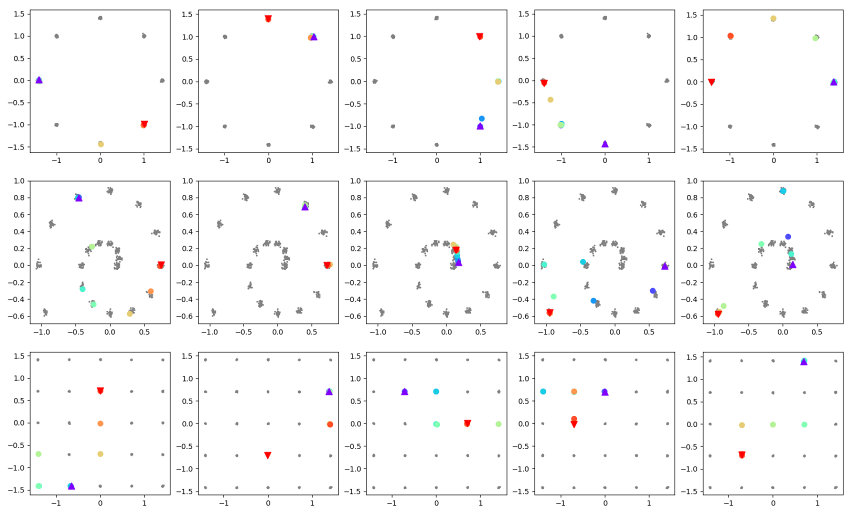


Fig. 3.4. Interpolation: ▼ and ▲ indicates the first and last samples in the interpolations, other colored samples are the interpolations.

middle of two modes. This may be because the generator G is not able to approximate a step function well.

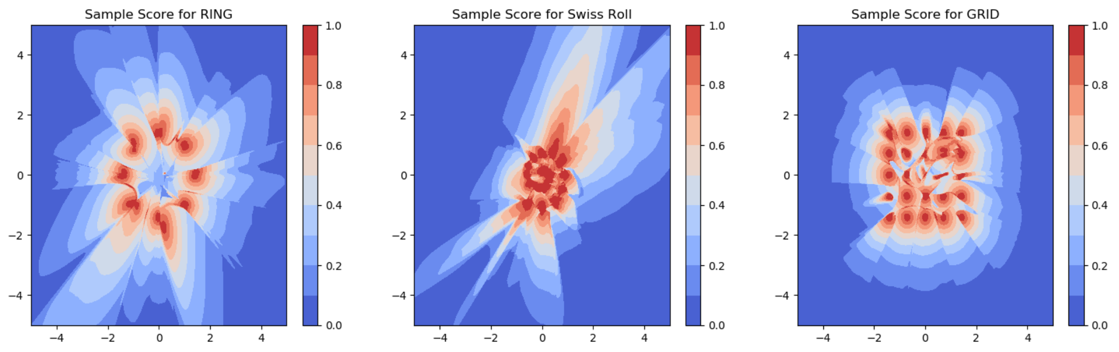


Fig. 3.5. Quality Check

Individual sample quality check From the probability interpretation of iW-GANs, we naturally adopt the reconstruction error $\|X - G(Q(X))\|$, or the *quality score*

$$\text{Quality Score} = \exp\left(-\|X - G(Q(X))\|\right)$$

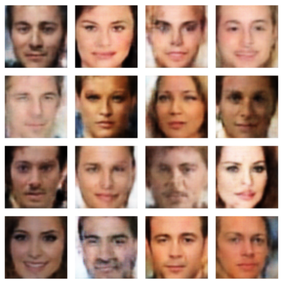
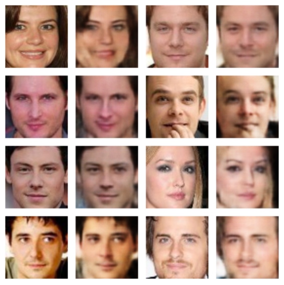
as the metric of the quality of any individual sample. The larger the quality score is, the better quality the sample has.

Figure 3.5 shows their quality scores for different samples. The quality scores of samples near the modes of the true distribution are close to 1, and become smaller as the sample draw is away from the modes. This indicates that the iWGAN does converge and learns the distribution well, and the *quality score* is a reliable metric for the individual sample quality.

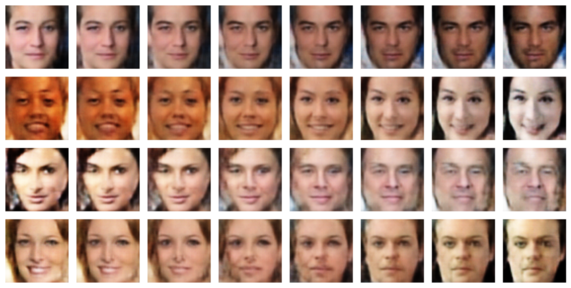
3.5.2 MNIST & CelebA

We experimentally demonstrate our model's ability on two well-known benchmark datasets, MNIST and CelebA. We present the performance of iWGAN on CelebA in this section and the performance on MNIST in the Appendix. CelebA (CelebFaces Attributes Dataset) is a large-scale face attributes dataset with $202,599$ $3\times64\times64$ colored celebrity face images, which cover large pose variations and diverse people. This dataset is ideal for training models to generate synthetic images. The MNIST database (Modified National Institute of Standards and Technology database) is another large database of handwritten digits $0 \sim 9$ that is commonly used for training various image processing systems. The MNIST database contains $70,000$ 28×28 grey images. CelebA is a more complex dataset than MNIST. The CelebA dataset is available at `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html` and the MNIST dataset is available at `http://yann.lecun.com/exdb/mnist/`.

The first result by iWGAN on CelebA is shown in Figure 3.6. We chosen 64to be dimensions of the latent space Z of CelebA. For each panel, Figure 3.6 respectively shows the generated samples from $G(Z)$, the reconstructed samples from $G(Q(X))$, and the latent space interpolation between two randomly chosen images. In particular, we perform latent space interpolations between CelebA validation set examples. We sample pairs of validation set examples x_1 and x_2 and project them into z_1 and z_2 by the encoder Q. We then linearly interpolate between z_1 and z_2 and pass the

(a) Generated samples (b) Reconstructions



(c) Interpolations

Fig. 3.6. iWGAN on CelebA

intermediary points through the decoder to plot the input-space interpolations. In addition, Figure 3.7 shows the first 8 dimensions of the latent space calculated by $Q(x)$ on CelebA. Figures 3.6 and 3.7 visually demonstrate that iWGANs can simultaneously generate high quality samples, produce small reconstruction errors, and have meaningful latent codes. Figure 3.8 also displays images with high and low quality scores selected from CelebA.

We compare iWGAN, both visually and numerically, with WGAN-GP, WAE, and ALI. Figure 3.9a, Figure 3.9b, Figure 3.9c, and Figure 3.9d display the random generated samples from iWGAN, WGAN-GP, WAE, and ALI, respectively. The generated faces by iWGAN demonstrate higher qualities than other three methods. The top panel of Figure 3.10 shows the comparison between real images and reconstructed images among three methods, iWGAN, WAE and ALI. Note that WGAN-GP cannot

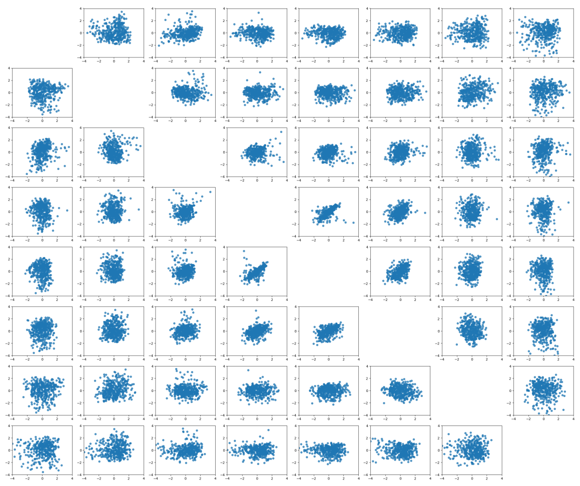


Fig. 3.7. Latent Space of CelebA dataset

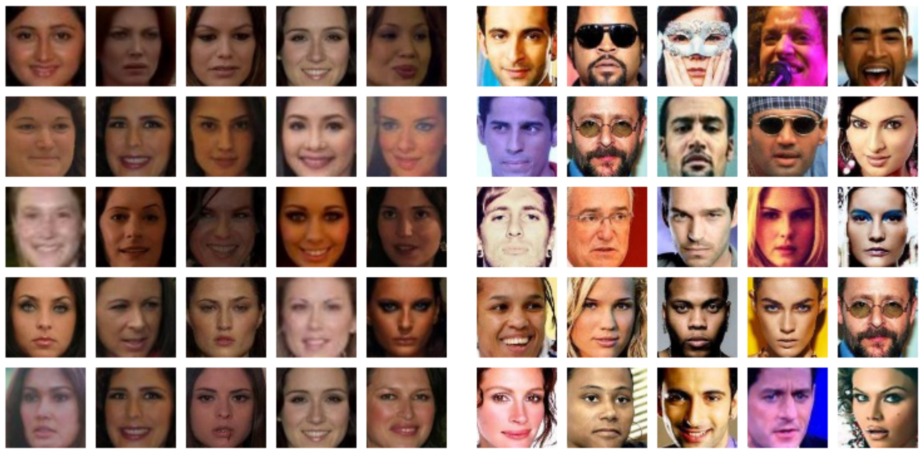
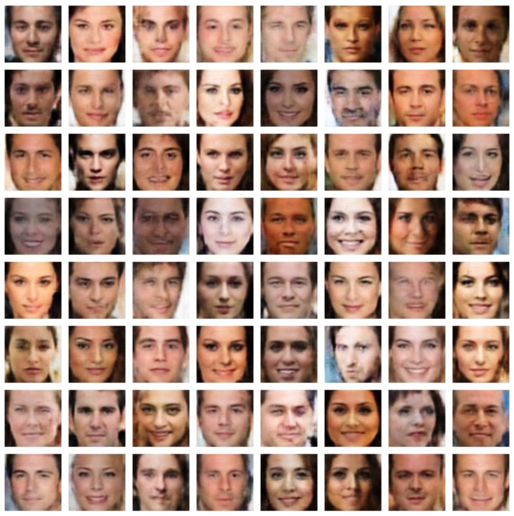
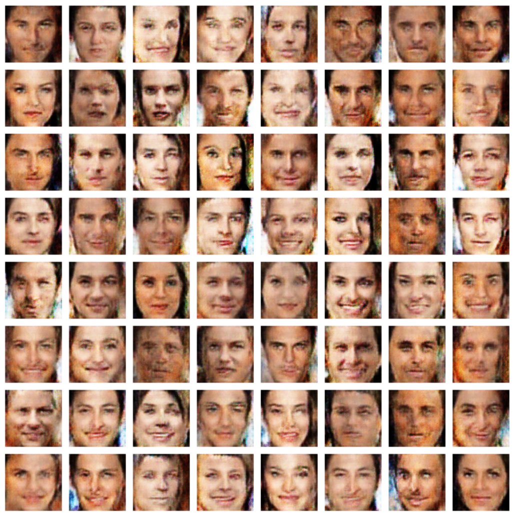


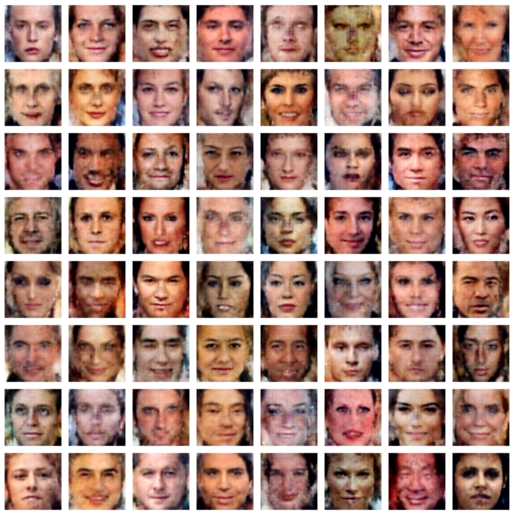
Fig. 3.8. Images with high (left) and low (right) quality scores by iWGAN

provide reconstructed images since it does not produce the latent codes. The bottom panel of Figure 3.10 shows the interpolated images by iWGAN, WAE, and ALI.

We numerically compare these four methods using four performance measures such as inception scores (IS), Fréchet inception distances (FID), reconstruction errors (RE), and maximum mean discrepancy (MMD) between encodings and standard normal random variables. Proposed by [34], the IS involves using a pre-trained Inception v3


(a) iWGAN


(b) WGAN-GP


(c) WAE

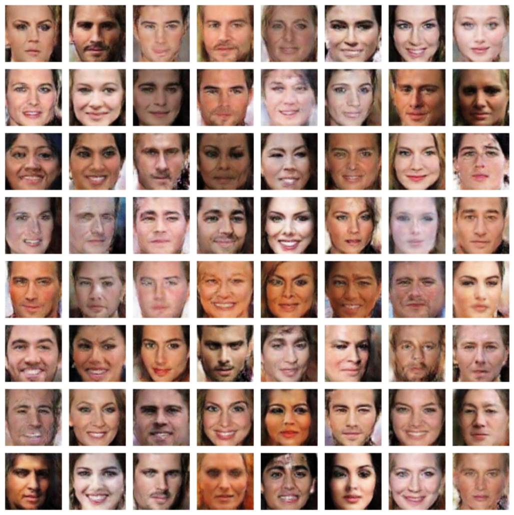
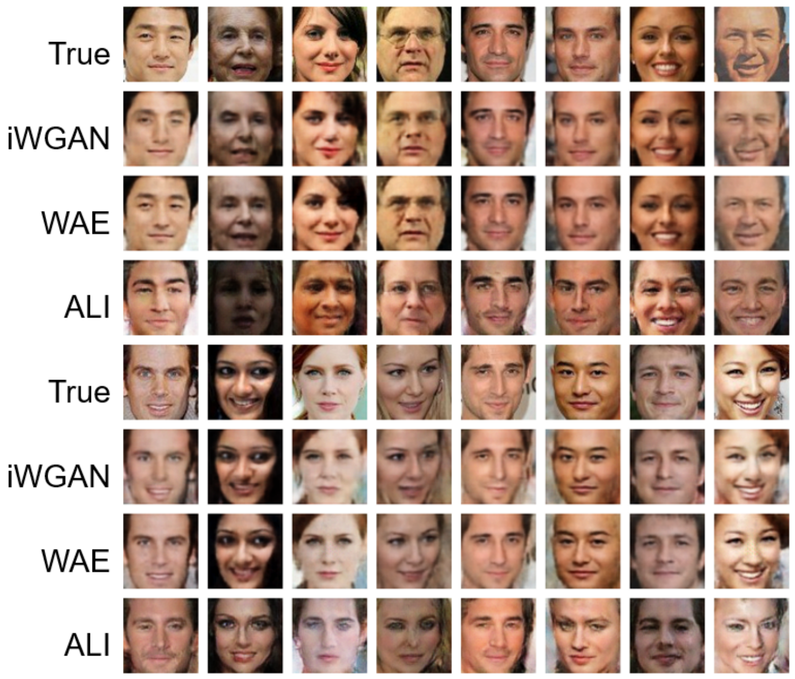

(d) ALI

Fig. 3.9. Generated samples by different models.

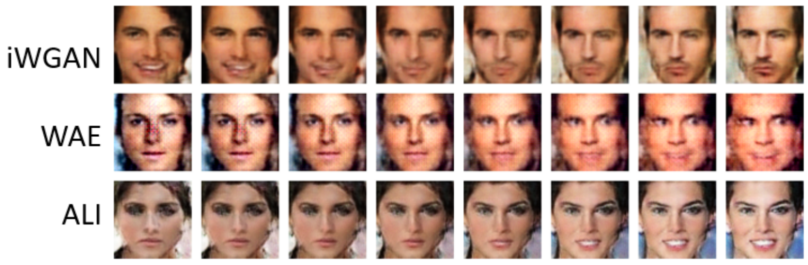
model to predict the class probabilities for each generated image. These predictions are then summarized into the IS by the KL divergence as following,

$$\mathrm{IS} = \exp\left(\mathbb{E}_{x \sim P_{G(Z)}} D_{KL}\left(p(y|x) \| p(y)\right)\right), \tag{3.14}$$

where $p(y|x)$ is the predicted probabilities conditioning on the generated images, and $p(y)$ is the corresponding marginal distribution. Higher scores are better, correspond-



(a) Reconstructions



(b) Interpolations

Fig. 3.10. Reconstruction comparison.

ing to a larger KL-divergence between the two distributions. The FID is proposed by [35] to improve the IS by actually comparing the statistics of generated samples to real samples. It is defined as the Fréchet distance between two multivariate Gaussians,

$$\mathrm{FID} = \|\mu_r - \mu_G\|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_G - 2(\Sigma_r \Sigma_G)^{1/2}\right), \tag{3.15}$$

where $X_r \sim N(\mu_r, \Sigma_r)$ and $X_G \sim N(\mu_G, \Sigma_G)$ are the 2048-dimensional activations of the Inception-v3 pool-3 layer for real and generated samples respectively. For the FID, the lower the better. However, as discussed in [36], IS is not a reliable metric for the wellness of generated samples. This is also consistent with our experiments. Although WAE delivers the best inception scores among four methods, WAE also has the worst FID scores. The generated samples (Figure 3.9c) show that WAE is not the best generative model compared with other three methods. Furthermore, the reconstruction error (RE) is defined as

$$\mathrm{RE} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{X}_i - X_i\|_2, \tag{3.16}$$

where $\hat{X}_i$ is the reconstructed sample for X_i. RE is used to measure if the method has generated meaningful latent encodings. Smaller reconstruction errors indicate a more meaningful latent space which can be decoded into the original samples. The maximum mean discrepancy (MMD) is defined as

$$\mathrm{MMD} = \frac{1}{N(N-1)}\sum_{l\neq j} k(z_l, z_j) + \frac{1}{N(N-1)}\sum_{l\neq j} k(\tilde{z}_l, \tilde{z}_j) - \frac{2}{N^2}\sum_{l,j} k(z_l, \tilde{z}_j) \tag{3.17}$$

where k is a positive-definite reproducing kernel, z_i's are drawn from prior distribution P_Z, and $\tilde{z}_i = Q(x_i)$ are the latent encodings of real samples. MMD is used to measure the difference between distribution of latent encodings and standard normal random variables. Smaller MMD indicates that the distribution of encodings is close to the standard normal distribution.

From Table 3.1, in terms of generative models, iWGAN and ALI are better models, where WGAN-GP comes after, but WAE is suffering from generating clear pictures. In terms of RE and MMD, iWGAN and WAE are better choices, where ALI cannot always reconstruct the sample to itself (Figure 3.10a). In general, Table 3.1 shows that iWGAN has successfully produced both meaningful encodings and reliable generator simultaneously.

Table 3.1. Comparison of iWGAN, ALI, WAE, WGAN-GP

Methods	IS	FID	RE	MMD
True	1.96(0.019)	18.63	–	–
iWGAN	1.51(0.017)	**51.20**	**13.55(2.41)**	$\mathbf{6 \times 10^{-3}}$
ALI	1.50(0.014)	**51.12**	34.49(8.23)	0.39
WAE	**1.71(0.029)**	77.53	**9.88(1.42)**	$\mathbf{4 \times 10^{-3}}$
WGAN-GP	**1.54(0.016)**	61.39	–	–

3.6 Conclusion

We have developed a novel iWGAN model, which fuses auto-encoders and GANs in a principle way. We have established a generalization error bound for iWGAN. We have provided a solid probabilistic interpretation on iWGAN using the maximum likelihood principle. Our training algorithm with an iterative primal and dual optimization has demonstrated an efficient and stable learning. We have proposed a stopping criteria for our algorithm and a metric for individual sample quality checking. The empirical results on both synthetic and benchmark datasets are state-of-the-art.

We now mention several future directions for research on iWGAN. First, one might be interested in applying iWGAN into image-to-image translation, as the extension should be straightforward. A second direction is to develop a formal hypothesis testing procedure to test whether the samples generated from iWAGN is the same as the data distribution. We are also working on incorporating iWGAN into the recent GAN modules such as BigGAN [37], which can produce high-resolution and high-fidelity images. As its name suggests, the BigGAN focuses on scaling up the GAN models including more model parameters, larger batch sizes, and architectural changes. Instead, iWGAN is able to stabilize GAN training, and it is a promising idea to fuse these two frameworks together.

3.A Proof of Theorem 3.2.1

According to the Nash embedding theorem [38, 39], every d-dimensional smooth Riemannian manifold $\mathcal{X}$ possesses a smooth isometric embedding into $\mathbb{R}^p$ with $p = \max\{d(d+5)/2, d(d+3)/2+5\}$. Therefore, there exists an injective mapping $u : \mathcal{X} \to \mathbb{R}^p$ which preserves the metric in the sense that the manifold metric on $\mathcal{X}$ is equal to the pullback of the usual Euclidean metric on $\mathbb{R}^p$ by u. The mapping u is injective so that we can define the inverse mapping $u^{-1} : u(\mathcal{X}) \to \mathcal{X}$.

Let $\tilde{X} = u(X) \in \mathbb{R}^p$, and write $\tilde{X} = (\tilde{X}_1, \dots, \tilde{X}_p)$. Let $F_i(x) = \mathbb{P}(\tilde{X}_i \le x)$, $i = 1, \dots, p$, be the marginal cdfs. By applying the probability integral transformation to each component, the random vector

$$\big(U_1, U_2, \dots, U_p\big) := \big(F_1(\tilde{X}_1), F_2(\tilde{X}_2), \dots, F_p(\tilde{X}_p)\big)$$

has uniformly distributed marginals. Let $C : [0,1]^p \to [0,1]$ be the copula of $\tilde{X}$, which is defined as the joint cdf of $(U_1, \dots, U_p)$:

$$C(u_1, u_2, \dots, u_p) = \mathbb{P}\big(U_1 \le u_1, U_2 \le u_2, \dots, U_p \le u_p\big).$$

The copula C contains all information on the dependence structure among the components of $\tilde{X}$, while the marginal cumulative distribution functions F_i contain all information on the marginal distributions. Therefore, the joint cdf of $\tilde{X}$ is

$$H(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_p) = C\big(F_1(\tilde{x}_1), F_2(\tilde{x}_2), \dots, F_p(\tilde{x}_p)\big).$$

Define, for $i = 2, \dots, p$,

$$C_i(u_1, u_2, \dots, u_i) = C\big(u_1, u_2, \dots, u_i, 1, \dots, 1\big).$$

The conditional distribution of U_k, given $U_1, \dots, U_{k-1}$, is given by [40]

$$\begin{aligned} C_k(u_k|u_1, \dots, u_{k-1}) &= \mathbb{P}\big(U_k \le u_k | U_1 = u_1, \dots, U_{k-1} = u_{k-1}\big) \\ &= \frac{\big[\partial^{k-1} C_k(u_1, \dots, u_k)/\partial u_1 \cdots \partial u_{k-1}\big]}{\big[\partial^{k-1} C_{k-1}(u_1, \dots, u_k)/\partial u_1 \cdots \partial u_{k-1}\big]}, \end{aligned}$$

for $k = 2, \dots, p$.

We will construct Q^* as follows. First, we obtain $\tilde{X} \in \mathbb{R}^p$ by $\tilde{X} = u(X)$. Second, we transform $\tilde{X}$ into a random vector with uniformly distributed marginals $(U_1, \dots, U_p)$ by the marginal cdf F_i. Then, define $\tilde{U}_1 = U_1$ and

$$\tilde{U}_k = C_k\big(U_k | U_1, \dots, U_{k-1}\big), \quad k = 2, \dots, p.$$

Hence, $\tilde{U}_1, \dots, \tilde{U}_p$ are independent uniform random variables. Finally, let $Z_i = \Phi^{-1}(U_i)$ for $i = 1, \dots, p$. This completes the transformation Q^* from X to $Z = (Z_1, \dots, Z_p)$.

The above process can be inverted to obtain G^*. First, we transform Z into independent uniform random variables by $\tilde{U}_i = \Phi(Z_i)$ for $i = 1, \dots, p$. Next, let $U_1 = \tilde{U}_1$. Define

$$U_k = C_k^{-1}(\tilde{U}_k | \tilde{U}_1, \dots, \tilde{U}_{k-1}), \quad i = 2, \dots, p,$$

where $C_k^{-1}(\cdot | u_1, \dots, u_k)$ is the inverse of C_k and can be obtained by numerical root finding. Finally, let $\tilde{X}_i = F_i^{-1}(U_i)$ for $i = 1, \dots, p$ and $X = u^{-1}(\tilde{X})$, where $u^{-1} : u(\mathcal{X}) \to \mathcal{X}$ is the inverse mapping of u. This completes the transformation G^* from Z to X.

3.B Proof of Theorem 3.2.2

By the iWGAN objective (3), (5) holds. Since W_1 is a distance between two probability measures, $W_1(P_X, P_{G(Z)}) \le \overline{W}_1(P_X, P_{G(Z)})$. If there exists a $Q^* \in \mathcal{Q}$ such that $Q^*(X)$ has the same distribution as P_Z, we have

$$\overline{W}_1(P_X, P_{G(Z)}) \le W_1(P_X, P_{G(Q^*(X))}) + W_1(P_{G(Q^*(X))}, P_{G(Z)}) = W_1(P_X, P_{G(Z)}).$$

Hence, $W_1(P_X, P_{G(Z)}) = \overline{W}_1(P_X, P_{G(Z)})$. We also observe that

$$\sup_f L(\widetilde{G}, \widetilde{Q}, f) = W_1(P_X, P_{\widetilde{G}(\widetilde{Q}(X))}) + W_1(P_{\widetilde{G}(\widetilde{Q}(X))}, P_{\widetilde{G}(Z)}).$$

By Theorem 3.2.1, we have $\inf_{G,Q} L(G, Q, \widetilde{f}) \leq L(G^*, Q^*, \widetilde{f}) = 0$ when the encoder and the decoder have enough capacities. Therefore, the duality gap is larger than $W_1(P_X, P_{\widetilde{G}(\widetilde{Q}(X))}) + W_1(P_{\widetilde{G}(\widetilde{Q}(X))}, P_{\widetilde{G}(Z)})$. It is easy to see that, if $\widetilde{G}$ outputs the same distribution as X and $\widetilde{Q}$ outputs the same distribution as Z, both the duality gap and $\overline{W}_1(P_X, P_{G(Z)})$ are zeros and $X = \widetilde{G}(\widetilde{Q}(X))$ for $X \sim P_X$.

3.C Proof of Theorem 3.3.1

We first consider the difference between population $W_1(P_X, P_{G(Z)})$ and empirical $\widehat{W}_1(P_X, P_{G(Z)})$ given n samples $S = \{x_1, \dots, x_n\}$. Let f_1 and f_2 be their witness function respectively. Using the dual form of 1-Wassertein distance, we have

$$\begin{aligned}
&W_1(P_X, P_{G(Z)}) - \widehat{W}_1(P_X, P_{G(Z)}) \\
=&\mathbb{E}_{X\sim P_X}\{f_1(X)\} - \mathbb{E}_{Z\sim P_Z}\{f_1(G(Z))\} - \frac{1}{n}\sum_{i=1}^{n} f_2(x_i) + \mathbb{E}_{Z\sim P_Z}\{f_2(G(Z))\} \\
\leq&\mathbb{E}_{X\sim P_X}\{f_1(X)\} - \mathbb{E}_{Z\sim P_Z}\{f_1(G(Z))\} - \frac{1}{n}\sum_{i=1}^{n} f_1(x_i) + \mathbb{E}_{Z\sim P_Z}\{f_1(G(Z))\} \\
\leq& \sup_f \mathbb{E}_{X\sim P_X}\{f(X)\} - \frac{1}{n}\sum_{i=1}^{n} f(x_i) \triangleq \Phi(S).
\end{aligned}$$

Given another sample set $S' = \{x_1, \dots, x_i', \dots, x_n\}$, it is clear that

$$\Phi(S) - \Phi(S') \leq \sup_f \frac{|f(x_i) - f(x_i')|}{n} \leq \frac{\|x_i - x_i'\|}{n} \leq \frac{2B}{n},$$

where the second inequality is obtained since f is 1-Lipschitz continuous function. Applying McDiamond's Inequality, with probability at least $1-\delta/2$ for any $\delta \in (0,1)$, we have

$$\Phi(S) \leq \mathbb{E}\{\Phi(S)\} + B\sqrt{\frac{2}{n}\log\left(\frac{2}{\delta}\right)}. \tag{3.18}$$

By the standard technique of symmetrization in [41], we have

$$\mathbb{E}\{\Phi(S)\} = \mathbb{E}\left\{\sup_f \mathbb{E}_{X\sim P_X}\{f(X)\} - \frac{1}{n}\sum_{i=1}^{n} f(x_i)\right\} \leq 2\mathfrak{R}_n(\mathcal{F}). \tag{3.19}$$

It has been proved in [41] that with probability at least $1-\delta/2$ for any $\delta \in (0,1)$,

$$\mathfrak{R}_n(\mathcal{F}) \leq \widehat{\mathfrak{R}}_n(\mathcal{F}) + B\sqrt{\frac{2}{n}\log\left(\frac{2}{\delta}\right)}. \tag{3.20}$$

Combining Equation (3.18), Equation (3.19) and Equation (3.20), we have

$$W_1(P_X, P_{G(Z)}) \leq \widehat{W_1}(P_X, P_{G(Z)}) + 2\widehat{\mathfrak{R}}_n(\mathcal{F}) + 3B\sqrt{\frac{2}{n}\log\left(\frac{2}{\delta}\right)}.$$

By Theorem 3.2.2, we have $\widehat{W_1}(P_X, P_{G(Z)}) \leq \widehat{\widehat{W}}_1(P_X, P_{G(Z)})$. Thus,

$$W_1(P_X, P_{G(Z)}) \leq \widehat{\widehat{W}}_1(P_X, P_{G(Z)}) + 2\widehat{\mathfrak{R}}_n(\mathcal{F}) + 3B\sqrt{\frac{2}{n}\log\left(\frac{2}{\delta}\right)}.$$

3.D Experimental Results on MNIST

3.D.1 Latent Space

Figure 3.11 shows the latent space of MNIST, i.e. $Q(X)_i$ against $Q(X)_j$ for all $i \neq j$.

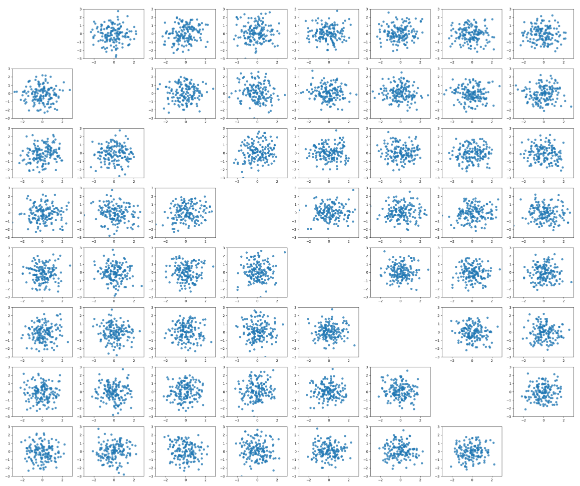
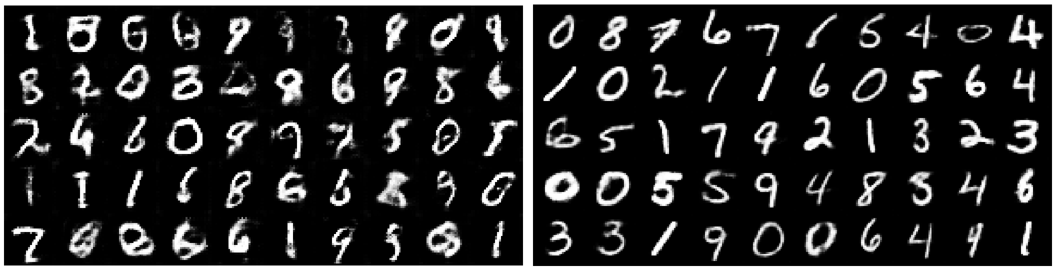


Fig. 3.11. Latent Space of MNIST dataset

3.D.2 Generated Samples

Figure 3.12 shows the comparison of random generated samples between WGAN-GP and iWGAN. Figure 3.13 shows examples of interpolations of two random generated samples.



(a) WGAN-GP (b) iWGAN

Fig. 3.12. Generated samples on MNIST

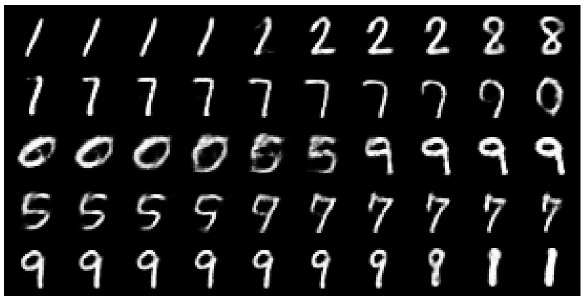
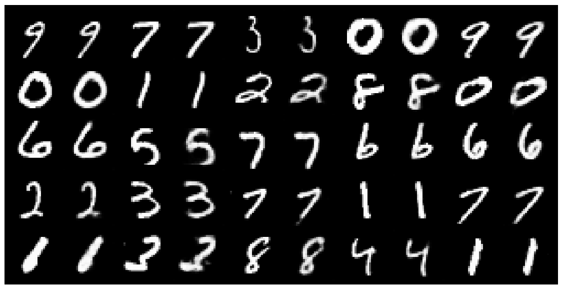
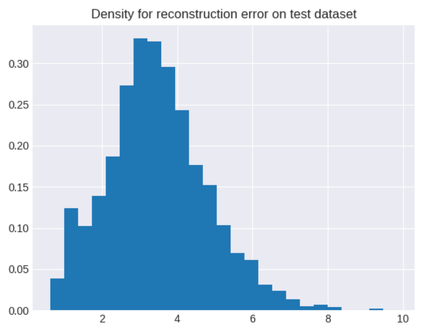


Fig. 3.13. Interpolations by iWGAN on MNIST

3.D.3 Reconstruction

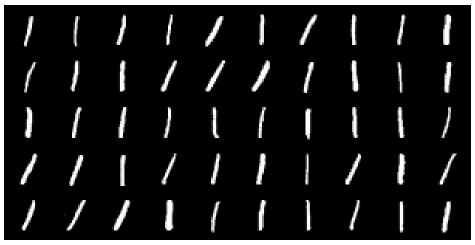
Figure 3.14b shows, based on the samples from validation dataset, the distribution of reconstruction error. Figure 3.14a shows examples of reconstructed samples. Figure 3.15 shows the best and worst samples based on quality scores from the validation dataset.


(a) Reconstructions



(b) Histogram of RE

Fig. 3.14. Reconstructions on MNIST


(a) Samples with high quality scores

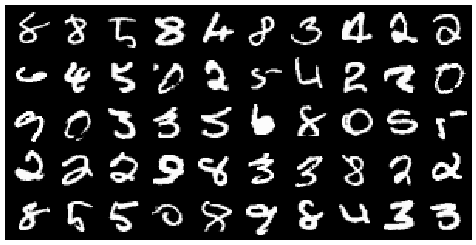

(b) Samples with low quality scores

Fig. 3.15. Sample quality check by iWGAN on the validation dataset of MNIST

3.E Architectures

The codes used for this chapter is available at: `https://drive.google.com/drive/folders/1-_vIrbOYwf2BH1lOrVEcEPJUxkyV5CiB?usp=sharing`. In this section, we present the architectures used for each experiment.

3.E.1 Mixture of Guassians

For Mixture Guassians, the latent space $Z \in \mathbb{R}^5$, for each batch, the sample size is 256.

Encoder architecture:

$$\begin{aligned} x \in \mathbb{R}^2 &\to FC_{1024} \to RELU \\ &\to FC_{512} \to RELU \\ &\to FC_{256} \to RELU \\ &\to FC_{128} \to RELU \to FC_5 \end{aligned}$$

Generator architecture:

$$\begin{aligned} z \in \mathbb{R}^5 &\to FC_{512} \to RELU \\ &\to FC_{512} \to RELU \\ &\to FC_{512} \to RELU \to FC_2 \end{aligned}$$

Discriminator architecture:

$$\begin{aligned} x \in \mathbb{R}^2 &\to FC_{512} \to RELU \\ &\to FC_{512} \to RELU \\ &\to FC_{512} \to RELU \to FC_1 \end{aligned}$$

3.E.2 MNIST

For MNIST, the latent space $Z \in \mathbb{R}^8$ and batch size is 250.

Encoder architecture:

$$\begin{aligned} x \in \mathbb{R}^{28\times 28} &\to Conv_{128} \to RELU \\ &\to Conv_{256} \to RELU \\ &\to Conv_{512} \to RELU \to FC_8 \end{aligned}$$

Generator architecture:

$$\begin{aligned} z \in \mathbb{R}^8 &\rightarrow FC_{4\times4\times512} \rightarrow RELU \\ &\rightarrow ConvTrans_{256} \rightarrow RELU \\ &\rightarrow ConvTrans_{128} \rightarrow RELU \rightarrow ConvTrans_1 \end{aligned}$$

Discriminator architecture:

$$\begin{aligned} x \in \mathbb{R}^{28\times28} &\rightarrow Conv_{128} \rightarrow RELU \\ &\rightarrow Conv_{256} \rightarrow RELU \\ &\rightarrow Conv_{512} \rightarrow RELU \rightarrow FC_1 \end{aligned}$$

3.E.3 CelebA

For CelebA, the latent space $Z \in \mathbb{R}^{64}$ and batch size is 64.

Encoder architecture:

$$\begin{aligned} x \in \mathbb{R}^{64\times64\times3} &\rightarrow Conv_{128} \rightarrow LeakyRELU \\ &\rightarrow Conv_{256} \rightarrow InstanceNorm \rightarrow LeakyRELU \\ &\rightarrow Conv_{512} \rightarrow InstanceNorm \rightarrow LeakyRELU \rightarrow Conv_1 \end{aligned}$$

Generator architecture:

$$\begin{aligned} z \in \mathbb{R}^{64} &\rightarrow FC_{4\times4\times1024} \\ &\rightarrow ConvTrans_{512} \rightarrow BN \rightarrow RELU \\ &\rightarrow ConvTrans_{256} \rightarrow BN \rightarrow RELU \\ &\rightarrow ConvTrans_{128} \rightarrow BN \rightarrow RELU \rightarrow ConvTrans_3 \end{aligned}$$

Discriminator architecture:

$$\begin{aligned} x \in \mathbb{R}^{64\times 64\times 3} &\rightarrow Conv_{128} \rightarrow LeakyRELU \\ &\rightarrow Conv_{256} \rightarrow InstanceNorm \rightarrow LeakyRELU \\ &\rightarrow Conv_{512} \rightarrow InstanceNorm \rightarrow LeakyRELU \rightarrow Conv_{1} \end{aligned}$$

4. NONLINEAR VARIABLE SELECTION VIA DEEP NEURAL NETWORKS

4.1 Introduction

Recent advances in technology, such as biotechnologies and artificial intelligence, have pushed real-world data into an extremely high-dimensional space. The demand for high-dimensional data analysis has grown significantly, especially in the applications of social networking, bioinformatics, and neuroscience. At the same time, high-dimensional data analysis has been a challenge to researchers in the field of statistics and machine learning. How to approximate the true underlying system and identify relevant features based on high-dimensional data (typically with the sample size smaller than the dimension, a.k.a. small-n-large-p) is a challenging task. Starting from [42], [43], [44] who introduced, respectively, the famous criteria C_p, AIC and BIC, the problem of variable selection was extensively studied in the statistical and machine learning literature from both theoretical and algorithm perspective.

The literature on high-dimensional linear regression is too vast to summarize. Popular methods including Lasso [45] and elastic net [46], based on an ℓ_1-penalty of the coefficients, have played an important role in variable selection. Non-convex penalties such as SCAD [47] and MCP [48] have also demonstrated promising results for variable selection. Another direction is the non-convex greedy pursuit including Orthogonal Matching Pursuit(OMP) [49, 50], Compressive Sampling Matching Pursuit [51]. These greedy algorithms implicitly approximate the solution to the ℓ_0-constrained least squares problem on the linear regression model. High-dimensional generalized linear models are studied by [52] and [53]. It is well-known that, with a design matrix satisfying some variant of irrepresentable condition, consistent variable selection is possible under the condition $s \log(p/s) = o(n)$ as $n \to \infty$ [54], where s is the intrinsic dimension of relevant variables.

Besides the curse of high-dimensionality, the challenge of this problem also lies in the possible nonlinear relationship between the predictors and the response. The estimation and hypotheses testing problems for high-dimensional sparse additive model were studied in [55], [56] and [57]. In contrast, for high-dimensional nonparametric regression, [58] established tight conditions making it possible to consistently estimate the set of relevant variables. In particular, when the intrinsic dimension s is fixed, the situation in nonparametric regression is the same as in linear regression, that is, $\log p$ is small compared to the sample size n. [58] also presented when the number of relevant variables s tends to infinity, consistent variable selection in nonparametric set-up is possible only if $s + \log\log p$ is small compared to $\log n$. On the other hand, there are few papers discussing practical procedures for recovering the sparsity for nonlinear models. A few exceptions are [59–61], and some kernel methods including [62], [63], [64]. However, [61] only discusses a single layer neural network model without considering the approximation error, and kernel methods are essentially equivalent to a single hidden layer neural network [65].

In this chapter, we develop a novel deep learning-based variable selection method for high-dimensional nonlinear systems. During the past decade, deep learning has demonstrated great successes in solving many complex artificial intelligence tasks such as pattern recognition and speech understanding [66]. The problems that the current deep learning techniques work well usually have a very large sample size and a relatively small number of features (without or with very few false features). It is hard to apply deep learning directly to the so-called small-n-large-p problem. This chapter approximates the nonlinear system by a deep neural network, which composes of both a selection layer and approximation layers. The problem can be cast as a sparsity-constrained optimization with a sparse parameter in the selection layer and other parameters in the approximation layers. We propose a novel greedy algorithm, called *Deep Feature Selection (DFS)*, to estimate both parameters under the framework of supervised learning. The contributions of the chapter are: (a). We have proved the accuracy of DFS for a class of loss functions that have a *Generalized Stable Restricted Hessian (GSRH)*. The GSRH is an extension of the *Stable Restricted Hessian (SRH)* introduced in [67]. This condition requires the loss function to have well-conditioned Hessian when restricted to sparse canonical subspaces. The original SRH only has a sparse parameter, while GSRH deals with both a sparse parameter in the selection layer and other parameters in the approximation layers. (b) We have developed an efficient DFS algorithm and it provides a practical procedure for recovering the sparsity pattern for nonlinear systems. Obtaining the global optima is NP-hard with an ℓ_0-constraint, but with the theoretical justifications, DFS yields accurate solutions and is a tractable algorithm. (c) We have evaluated the performance of DFS using simulated data and real data. Even for linear regression, our algorithm, without knowing the underlying linear structure, demonstrates superior performance on variable selection over other well-known algorithms such as LASSO and SCAD. For high-dimensional nonlinear classification, DFS outperforms other methods on both the selection error and the prediction error.

The rest of the chapter is organized as follows. In section 4.2, we describe the neural network architecture and provide the approximation error. In section 4.3, we present our algorithm. In section 4.4, we state main theorems, and the GSRH condition for the loss function is provided as well. In section 4.5, we compare DFS with other related algorithms for both a linear regression example and a nonlinear classification example. section 4.6 provides analysis on real datasets. section 4.7 gives some concluding remarks. The proofs are provided in Appendix.

4.2 The Model and The Approximation Error

Let $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^p$ be the high-dimensional explanatory variable and $y \in \mathcal{Y} \subset \mathbb{R}$ be the response variable. Consider a supervised learning problem:

$$y = t(f(\boldsymbol{x}) + \epsilon), \tag{4.1}$$

where $t : \mathbb{R} \to \mathcal{Y}$ is a fixed function related to the prediction problem, $f : \mathbb{R}^p \to \mathbb{R}$ is an unknown function, and ϵ is an independent noise. For regression, we may define $t(z) = z$, while for classification, we may define $t(z) = sign(z)$. For a large-p-small-n problem, assume $f \in \mathcal{F}_s$ is a s-sparse high-dimensional nonlinear function, where $\mathcal{F}_s$ is defined as:

$$\mathcal{F}_s = \Big\{ f : \mathbb{R}^p \to \mathbb{R} : \ \exists\ \bar{f}(\tilde{\boldsymbol{x}}) = f(\boldsymbol{x}), \tilde{\boldsymbol{x}} \in \boldsymbol{R}^s, \forall\ \boldsymbol{x} \in \boldsymbol{R}^p, \bar{f}\ is\ Lipschitz \Big\}, \tag{4.2}$$

where $\tilde{\boldsymbol{x}}$ represents the sub-vector of $\boldsymbol{x}$ only including all the coordinates with indices lying inside $\mathcal{S}$, which is a small subset of the index set $\{1, \ldots, p\}$ and $\text{Card}(\mathcal{S}) = s$.

We use deep neural networks to approximate such s-sparse functions. In particular, we adopt the network architecture displayed in Figure 4.1. This network has two main functional parts: the selection part and the approximation part. The main difference of Figure 4.1 from the usual neural network is that we add a sparse one-to-one linear layer between the input layer and the selection layer. In particular, the input variable $\boldsymbol{x} = (x_1, \ldots, x_p)^\top \in \mathbb{R}^p$ is transformed into $\boldsymbol{w} * \boldsymbol{x} = (w_1 x_1, \ldots, w_p x_p)^\top \in \mathbb{R}^p$, where $\boldsymbol{w}$ is a sparse parameter. The sparsity constraint on $\boldsymbol{w}$ only allows that the features corresponding to nonzero w_i's are selected. The function represented by this neural network can be written as

$$h(\boldsymbol{x}) = T_{H+1} \circ \sigma \circ T_H \circ \sigma \circ \cdots \sigma \circ T_1 \circ (\boldsymbol{w} * \boldsymbol{x}), \tag{4.3}$$

where $T_i(u) = \Theta^{(i)} u + b^{(i)}$, $i = 1, \ldots, H+1$, are affine transformations with unknown parameters $(\Theta^{(i)}, b^{(i)})$ with $\Theta^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$ and $b^{(i)} \in \mathbb{R}^{d_i}$, and $\sigma(\cdot)$ is the activation function. Popular choice of activation functions includes rectified linear unit (ReLU), sigmoid, and tanh. In this chapter, we will mainly focus on neural networks with the ReLU activation function.

The function $h(\boldsymbol{x})$ in (4.3) is a high-dimensional s-sparse piecewise affine function [68]. The collection of such functions is defined as

$$\mathcal{H}_{s,d} = \Big\{ h : h(\boldsymbol{x}) = T_{d+1} \circ \sigma \circ T_d \circ \sigma \circ \cdots \sigma \circ T_1 \circ (\boldsymbol{w} * \boldsymbol{x}),\ \|\boldsymbol{w}\|_0 = s \Big\}.$$

With an abuse of notation, we use $\boldsymbol{w}$ to denote the sparse parameter in the selection layer and $\boldsymbol{\Theta}$ to denote all other parameters in the approximation layers. The advantage of this architecture lies in the capability of simultaneously selecting relevant features and modeling complex nonlinear systems.

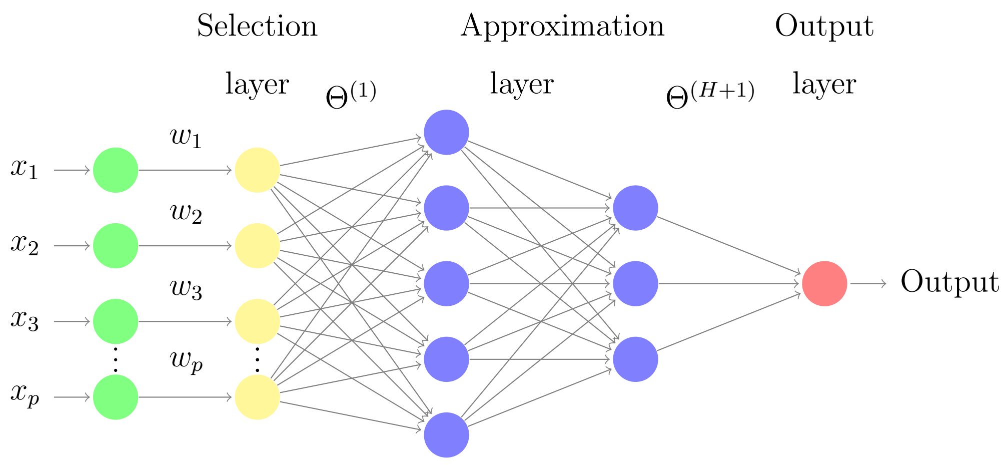


Fig. 4.1. Deep Feature Selection Architecture

In the following, we establish the approximation error when using a deep neural network (4.3) to approximate a high-dimensional s-sparse function in $\mathcal{F}_s$.

Theorem 4.2.1 *Assume that, for the function $f : \boldsymbol{R}^p \to \boldsymbol{R}$, there exists a Lipschitz function $\bar{f} : \boldsymbol{R}^s \to \boldsymbol{R}$ defined on the R-ball in l_2-norm, satisfying $|\bar{f}(\tilde{\boldsymbol{x}})| \leq \eta$ and $|\bar{f}(\tilde{\boldsymbol{x}}) - \bar{f}(\tilde{\boldsymbol{y}})| \leq \eta R^{-1} \|\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}}\|_2$, such that $f(\boldsymbol{x}) = \bar{f}(\tilde{\boldsymbol{x}})$ as defined in Equation 4.2. Then there exists a neural network $h(\boldsymbol{x}) = \sum_{i=1}^{2+2k} \Theta_i^{(2)} \sigma(\Theta_{i,\cdot}^{(1)}(\boldsymbol{w} * \boldsymbol{x}) + b_i^{(1)})$ with $\|\boldsymbol{w}\|_0 = s$ and $\left\|\Theta^{(2)}\right\|_2 \leq C_1$, where C_1 is a constant depending on s and η only, such that*

(a) Both $h(\boldsymbol{x})$ and $f(\boldsymbol{x})$ are s-sparse functions and have the same support;

(b)

$$\sup_{\boldsymbol{x}} |f(\boldsymbol{x}) - h(\boldsymbol{x})| \leq C_2 \eta k^{-1/s} \log k,$$

where C_2 is a constant depending on s only.

4.3 The Algorithm

Assume that $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ are n i.i.d. samples on $\mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^p \times \mathbb{R}$. Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a loss function, which could be cross entropy for classification, mean squared error for regression, or other loss functions. The empirical risk function is

$$l(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i)). \tag{4.4}$$

Since we are using a deep neural network (4.3) to approximate $f \in \mathcal{F}$, we minimize the empirical risk function over the neural network space $\mathcal{H}_{s,d}$. Recall that sparsity constraint is applied on $\boldsymbol{w}$ to achieve the task of variable selection. Here we adopted ℓ_0-penalty to realize the sparsity on w. [62] has similar scheme, but uses ℓ_1-penalty to achieve the sparsity of coefficients. However, ℓ_1 penalty tends to overshrink large coefficients and results in biased estimators, while ℓ_0 penalty can eliminate the estimation bias and enjoy the oracle properties [69].

Consider the following optimization problem:

$$\text{minimize} \quad g(\boldsymbol{w}, \boldsymbol{\Theta}) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i; \boldsymbol{w}, \boldsymbol{\Theta})) + \lambda_1 \|\boldsymbol{w}\|_2^2 + \lambda_2 \|\boldsymbol{\Theta}\|_2^2, \quad s.t. \quad \|\boldsymbol{w}\|_0 \leq s, \tag{4.5}$$

where $\|\cdot\|_2$ is the Frobenius norm, s is a positive integer controlling the sparsity level, and λ_1, λ_2 are regularization parameters. The ℓ_2 regularization is introduced to prevent overfitting on neural network and also meet the Generalized Stable Restricted Hessian (GSRH) conditions in the Section 4.4. We propose a novel algorithm called *Deep Feature Selection (DFS)*, which is inspired by and generalizes the GraSP algorithm [67], to approximate the solution to (4.5) for a broader class of objective functions.

Even for a simple quadratic objective, (4.5) is a NP-hard problem due to the combinatorial complexity. However, similar to the results of compressive sensing, if the objective function satisfies certain properties, this allows us to obtain accurate estimates through tractable algorithms. As described in section 4.4, to assure that DFS yields accurate solutions, we require the objective function to have a *Generalized*

Stable Restricted Hessian (GSRH). This property is analogous to the RIP condition in the standard compressive sensing framework and generalizes the SRH [67].

Algorithm 2 The DFS algorithm

```
**Input** $g(\cdot)$ and $s$
**Output** $\hat{\boldsymbol{w}}$, $\hat{\boldsymbol{\Theta}}$ initialization: $\hat{\boldsymbol{w}}$, $\hat{\boldsymbol{\Theta}}$
**WHILE** halting condition not holds
compute local gradient: $(\boldsymbol{z_w}, \boldsymbol{z_\Theta}) = \nabla g(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}})$
identify directions: $\mathcal{Z} = supp(\boldsymbol{z}_{\boldsymbol{w}_{2s}})$
merge supports: $\mathcal{T} = \mathcal{Z} \cup supp(\hat{\boldsymbol{w}})$
minimize over support: $(\boldsymbol{b_w}, \boldsymbol{b_\Theta}) = \arg\min_{\boldsymbol{w},\boldsymbol{\Theta}} g(\boldsymbol{w}, \boldsymbol{\Theta}),\ s.t.\ \boldsymbol{w}|_{\mathcal{T}^c} = \boldsymbol{0}$
prune estimate: $\hat{\boldsymbol{w}} = \boldsymbol{b}_{\boldsymbol{w}_s}$, $\hat{\boldsymbol{\Theta}} = \boldsymbol{b_\Theta}$
```

DFS is an iterative algorithm, given in Algorithm 2, that updates an estimated sparse $\hat{\boldsymbol{\Theta}}$ and non-sparse $\hat{\boldsymbol{\Theta}}$ at every iteration. First all weight matrices are initialized by a uniform distribution, $w_i \sim U(-\frac{1}{\sqrt{p}}, \frac{1}{\sqrt{p}})$ and $\theta_i \sim U(-\frac{1}{\sqrt{d_{i-1}}}, \frac{1}{\sqrt{d_{i-1}}})$, or simply follow the default normalization in your neural network packages. Then in each iteration, we first compute the gradient of the objective function, $(\boldsymbol{z_w}, \boldsymbol{z_\Theta}) = \nabla g(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}})$, at the current estimate. Then $2s$ coordinates of $\boldsymbol{z_w}$ with the largest magnitude are identified as $\mathcal{Z}$. These indices are combined with the current support of $\hat{\boldsymbol{w}}$ to form $\mathcal{T} = \mathcal{Z} \cup \hat{\mathcal{S}}$. Note that $\mathcal{T}$ has at most $3s$ indices. Next, we minimize the objective function subject to the constraint $\boldsymbol{w}|_{\mathcal{T}^c} = \boldsymbol{0}$ to obtain an intermediate estimate $(\boldsymbol{b_w}, \boldsymbol{b_\Theta})$. Finally, $\hat{\boldsymbol{w}}$ is updated as the best s-term approximation of the intermediate estimate $\boldsymbol{b_w}$, i.e. keep the s largest absolute values of $\boldsymbol{b_w}$, and $\hat{\boldsymbol{\Theta}} = \boldsymbol{b_\Theta}$.

The halting conditions of the algorithm are usually set to be a maximum iteration step and an unchanged support $\mathcal{T}$ from previous step. The maximum step can be set under the guidance of Corollary 4.4.1. If the minimization over support $\mathcal{T}$ reaches the global optima of objective function subject to the given support, the unchanged estimation of $(\boldsymbol{b_w}, \boldsymbol{b_\Theta})$, which returns unchanged estimation of the support. The minimization step in DFS can be relaxed by applying a restricted gradient descent, where the gradient of g is restricted on the union of $\mathcal{T}$ and the indices of the support of $\hat{\boldsymbol{\Theta}}$. Furthermore, Algorithm 2 is proposed for a fixed s. In practice, we apply the BIC-like criteria to tune an optimal s, and more details are given in section 4.5.

4.4 Theoretical Results

We first characterize the objective function for which accuracy of DFS can be guaranteed. For a twice continuous differentiable objective function, we impose a condition called *Generalized Stable Restricted Hessian.* The properties of this condition basically require that the curvature of the objective function over the sparse subspaces can be bounded locally, which weaken the convexity requirements of the objective function.

Definition 4.4.1 (Generalized Stable Restricted Hessian) *Suppose that g is a twice continuously differentiable function whose Hessian is denoted by $\boldsymbol{H}_g(\cdot)$. Furthermore, let*

$$A_k(\boldsymbol{w}, \boldsymbol{\Theta}) = \sup\left\{\boldsymbol{\Delta}^{\mathrm{T}}\boldsymbol{H}_g(\boldsymbol{w}, \boldsymbol{\Theta})\boldsymbol{\Delta}\,\middle|\, |\mathrm{supp}(\boldsymbol{w}) \cup \mathrm{supp}(\boldsymbol{\Delta}_{\boldsymbol{w}})| \leq k, \|\boldsymbol{\Delta}\|_2 = 1\right\} \quad (4.6)$$

and

$$B_k(\boldsymbol{w}, \boldsymbol{\Theta}) = \inf\left\{\boldsymbol{\Delta}^{\mathrm{T}}\boldsymbol{H}_g(\boldsymbol{w}, \boldsymbol{\Theta})\boldsymbol{\Delta}\,\middle|\, |\mathrm{supp}(\boldsymbol{w}) \cup \mathrm{supp}(\boldsymbol{\Delta}_{\boldsymbol{w}})| \leq k, \|\boldsymbol{\Delta}\|_2 = 1\right\}, \quad (4.7)$$

for all k-sparse vectors $\boldsymbol{w}$, where $\boldsymbol{\Delta}^{\mathrm{T}} = (\boldsymbol{\Delta}_{\boldsymbol{w}}^{\mathrm{T}}, \boldsymbol{\Delta}_{\boldsymbol{\Theta}}^{\mathrm{T}})$. Then g is said to have a Generalized Stable Restricted Hessian (GSRH) with constant μ_k, or in short μ_k-GSRH, if $1 \leq \frac{A_k(\boldsymbol{w},\boldsymbol{\Theta})}{B_k(\boldsymbol{w},\boldsymbol{\Theta})} \leq \mu_k$.

Next, we provide a trivial example to illustrate the motivation of the GSRH condition.

Example 1 *Let $g(\boldsymbol{w}, \boldsymbol{\Theta}) = \frac{1}{2}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{w} + \lambda_1\|\boldsymbol{w}\|_2^2 + \lambda_2\|\boldsymbol{\Theta}\|_2^2)$, where $\boldsymbol{Q} = 2 \times \mathbf{1}\mathbf{1}^{\mathrm{T}} - \boldsymbol{I}$. We have:*

$$\boldsymbol{H}_g(\boldsymbol{w}, \boldsymbol{\Theta}) = \begin{bmatrix} \boldsymbol{Q} + \lambda_1 I_p & \mathbf{0} \\ \mathbf{0} & \lambda_2 I_P \end{bmatrix} \quad (4.8)$$

Note that all diagonal entries of $\boldsymbol{Q}$ are all equal to one, while its off-diagonal entries are all equal to two. Therefore, for any 1-sparse vector $\boldsymbol{\Delta}_1$ we have $\boldsymbol{\Delta}_1^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{\Delta}_1 = \|\boldsymbol{\Delta}\|_2^2$. However, for $\boldsymbol{\Delta}_1 = (1, -1, 0, \dots, 0)^{\mathrm{T}}$, we have $\boldsymbol{\Delta}_1^{\mathrm{T}}(\boldsymbol{Q} + \lambda_1 I_p)\boldsymbol{\Delta}_1 = -2 + 2\lambda_1 < 0$, when $\lambda_1 < 1$, which means the Hessian of g is not positive semi-define, thus g is not convex. Then for any $\boldsymbol{\Delta} = (\boldsymbol{\Delta}_1, \boldsymbol{\Delta}_2)$, $\|\boldsymbol{\Delta}\| = 1$ and $\boldsymbol{\Delta}_1$ is a 1-sparse vector. $\boldsymbol{\Delta}^{\mathrm{T}}\boldsymbol{H}_g\boldsymbol{\Delta} = (1 + \lambda_1)\|\boldsymbol{\Delta}_1\|_2^2 + \lambda_2\|\boldsymbol{\Delta}_2\|_2^2$. If $(1 + \lambda_1) > \lambda_2$, $A_1(\boldsymbol{w}, \boldsymbol{\Theta}) = 1 + \lambda_1$ and $B_1(\boldsymbol{w}, \boldsymbol{\Theta}) = \lambda_2$. Therefore, $\mu_1 = \dfrac{1 + \lambda_1}{\lambda_2}$, and we can manipulate the ratio of λ_1 and λ_2 to make the condition of theorem verified.

Now define

$$(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star) = \arg\min g(\boldsymbol{w}, \boldsymbol{\Theta}), \quad s.t. \;\; \|\boldsymbol{w}\|_0 \le s \tag{4.9}$$

Theorem 4.4.1 *Suppose that g is a twice continuously differentiable function that has μ_{4s}-GSRH with $\mu_{4s} \le \frac{1+\sqrt{3}}{2}$. Furthermore, suppose that for some $\epsilon > 0$ we have $\epsilon \le B_{4s}(\boldsymbol{w}, \boldsymbol{\Theta})$ for all $4s$-sparse $\boldsymbol{w}$. Then,*

(a). The estimate at the i-th iteration, $(\hat{\boldsymbol{w}}^{(i)}, \hat{\boldsymbol{\Theta}}^{(i)})$, satisfies

$$\begin{aligned}\left\|(\hat{\boldsymbol{w}}^{(i)}, \hat{\boldsymbol{\Theta}}^{(i)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 \le\ & 2^{-i}\left\|(\hat{\boldsymbol{w}}^{(0)}, \hat{\boldsymbol{\Theta}}^{(0)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 \\ & + \frac{6 + 2\sqrt{3}}{\epsilon}\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2.\end{aligned} \tag{4.10}$$

(b). The support set of $\hat{\boldsymbol{w}}^{(i)}$, denoted as $\hat{\mathcal{S}}^{(i)}$, satisfies

$$\begin{aligned}\left\|\boldsymbol{w}^\star|_{\mathcal{S}\setminus\hat{\mathcal{S}}^{(i)}}\right\|_2 \le\ & 2^{-i}\left\|(\hat{\boldsymbol{w}}^{(0)}, \hat{\boldsymbol{\Theta}}^{(0)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 \\ & + \frac{6 + 2\sqrt{3}}{\epsilon}\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2,\end{aligned} \tag{4.11}$$

where $\mathcal{I}$ is the position of the $3s$ largest entries of $\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)$ in magnitude in terms of $\boldsymbol{w}$, $\mathcal{R}_{\boldsymbol{\Theta}}$ is the coordinates set of $\boldsymbol{\Theta}$, and $\mathcal{S}$ is the support set of $\boldsymbol{w}^\star$.

Remark 1 *Inequality (4.10) of Theorem 4.4.1 shows that $\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)$ determines how accurate the estimate can be. In fact, (4.10) holds for any s-sparse $\boldsymbol{w}^\star$ and $\boldsymbol{\Theta}^\star$, even if $(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)$ does not obey (4.9). However, arbitrary choices of $\boldsymbol{w}^\star$ and $\boldsymbol{\Theta}^\star$*

may result in a large norm of $\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}$, so that inequality (4.10) and (4.11) cannot be bounded well. From (4.5), $\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)$ can be decomposed into two parts

$$\nabla l(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star) + 2(\lambda_1 \|\boldsymbol{w}^\star\|_1 + \lambda_2 \|\boldsymbol{\Theta}^\star\|_1) \tag{4.12}$$

If $(\boldsymbol{w}^\star,\ \boldsymbol{\Theta}^\star)$ is sufficiently close to an unconstrained minimum of l, then the first part of (4.12) is negligible since $\nabla l(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)$ has small magnitude at minimum. For the second part, the magnitude is determined by the magnitude of λ's. Similarly, inequality (4.11) of Theorem 4.4.1 shows that the selection consistency is also determined by the norm of $\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}$.

Remark 2 *In practice, we tune λ_1 and λ_2 to have the objective function g satisfying a GSRH. Note that*

$$\boldsymbol{\Delta}^{\mathrm{T}} \boldsymbol{H}_g(\boldsymbol{w}, \boldsymbol{\Theta}) \boldsymbol{\Delta} = \boldsymbol{\Delta}^{\mathrm{T}} \Big(\boldsymbol{H}_l(\boldsymbol{w}, \boldsymbol{\Theta}) + \begin{bmatrix} \lambda_1 I_p & \mathbf{0} \\ \mathbf{0} & \lambda_2 I_P \end{bmatrix} \Big) \boldsymbol{\Delta} \tag{4.13}$$

where P is the dimension of $\boldsymbol{\Theta}$. When λ_1 and λ_2 have larger magnitude than the eigenvalues of $\boldsymbol{H}_l$, (4.13) is approximately $\dfrac{\max(\lambda_1, \lambda_2)}{\min(\lambda_1, \lambda_2)}$, so that $\mu_k \approx \dfrac{\max(\lambda_1, \lambda_2)}{\min(\lambda_1, \lambda_2)}$. By manipulating the ratio of λ_1 and λ_2, μ_{4s} in Theorem 4.4.1 can be easily bounded by $\frac{1+\sqrt{3}}{2}$. On the other hand, from Remark 1, large λ_1 and λ_2 will relax the right side of (4.10) and (4.11), which costs the loss of accuracy in both estimation and selection. In practice, the ratio of λ_1 and λ_2 is fixed to be 1.25. A sequence of (λ_1, λ_2) is tested, and the pair which returns the lowest loss function value will be adopted.

Let $\bar{M} = \max\{\max(|\hat{\boldsymbol{w}}_j^{(0)}|, |\boldsymbol{w}_j^\star|), j \in \hat{\mathcal{S}}^{(0)} \cup \mathcal{S}\}$, and $m = \min\{|\boldsymbol{w}_j^\star|, j \in \mathcal{S}\}$.

Corollary 4.4.1 *Suppose that g is a twice continuously differentiable function that has μ_{4s}-GSRH with $\mu_{4s} \leq \frac{1+\sqrt{3}}{2}$, and for some $\epsilon > 0$ we have $\epsilon \leq B_{4s}(\boldsymbol{w}, \boldsymbol{\Theta})$ for all $4s$-sparse $\boldsymbol{w}$. Furthermore, suppose $m \geq \frac{6+2\sqrt{3}}{\xi\epsilon} \left\| \nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}} \right\|_2$ for some $0 < \xi < 1$, and $\max(\|\boldsymbol{\Theta}^\star\|_\infty, \|\hat{\boldsymbol{\Theta}}^{(0)}\|_\infty) \leq c/2$, then we have*

$$\hat{\mathcal{S}}^{(i)} = \mathcal{S} \quad \textit{if} \quad i \geq \log_2 \frac{2\sqrt{s}\bar{M} + c\sqrt{P}}{(1-\xi)m}. \tag{4.14}$$

where P is the dimension of $\boldsymbol{\Theta}^\star$.

Remark 3 *When $\boldsymbol{w}$ is exactly s-sparse, Corollary 4.4.1 shows that our algorithm sequentially achieves the exact recovery of true support within a finite number of iterations. For most applications, our algorithm makes the right selection within 20 iterations. In practice, weight normalization can be applied to all approximation layers to speed up the convergence. In such case, $\|\boldsymbol{\Theta}^{\star}\|_{\infty} \leq 1$ is satisfied.*

In reality, our algorithm converges most of the time. However if divergences occurs due to improper hyper-parameters, like learning rate or the structure of the model, this can be detected by the loss over iteration not decreasing.

4.5 Simulation Studies

We illustrate the performance of the proposed model and algorithm using two simulated examples. We compare our method with other competing algorithms for both cases.

4.5.1 A Linear Regression Example

Although DFS is designed for high-dimensional nonlinear variable selection, we demonstrate that DFS is also competitive with other linear variable selection methods when the underlying model is a linear model, with $n < p$,

$$y = X\beta + \epsilon, \quad X \in \mathbb{R}^{n \times p}, \quad \beta \in \mathbb{R}^{p}. \tag{4.15}$$

We generated 100 datasets from this linear system. The number of nonzero coefficients is $s = 100$, the sample size is $n = 500$, and the feature dimension is $p = 1000$. To generate the design matrix X, we first generate an $n \times p$ random Gaussian matrix $\bar{X}$ whose entries are i.i.d. $\mathcal{N}(0, 1)$ and then normalize its columns to the $\sqrt{n}$-length. Denote X_i and $\bar{X}_i$ to be the ith column of X and $\bar{X}$, respectively. Then X is generated with $X_1 = \bar{X}_1$, $X_j = \bar{X}_j + 0.2 \times (\bar{X}_{j+1} + \bar{X}_{j-1})$, $j = 2, ..., p-1$ and $X_p = \bar{X}_p$. That is, the features $x_1, \ldots, x_p$ are mutually correlated. The underlying regression coefficient β is generated with nonzero coefficients uniformly distributed in $[m, M]$, where $m = \sqrt{2\log(p)/n}$ and $M = 100m$, and first 100 coefficients are set to be non-zero. The response vector y is generated by (4.15) with $\epsilon_1, \ldots, \epsilon_n$ independently from $\mathcal{N}(0,1)$. The variable selection performance is evaluated by the *False Selection Rate* and the *Negative Selection Rate*:

$$fsr = (\sum_{k=1}^{K} |\hat{\mathcal{S}}_k \backslash \mathcal{S}_k|)/(\sum_{k=1}^{K} |\hat{\mathcal{S}}_k|), \quad nsr = (\sum_{k=1}^{K} |\mathcal{S}_k \backslash \hat{\mathcal{S}}_k|)/(\sum_{k=1}^{K} |\mathcal{S}_k|) \tag{4.16}$$

, where $\mathcal{S}$ is the true support of $\boldsymbol{w}$, $\hat{\mathcal{S}}$ is the support of estimated $\hat{\boldsymbol{w}}$ and $|\cdot|$ denotes the cardinality of a set, K is the total number of datasets. The smaller the values of *fsr* and *nsr* are, the better the performance of the variable selection is.

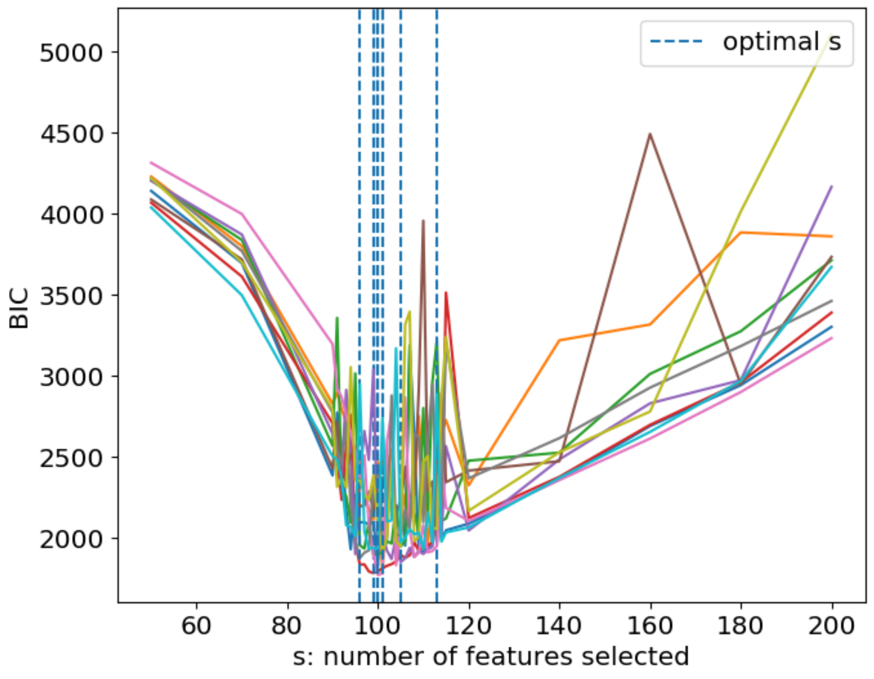


Fig. 4.2. Linear Regression Example: BIC-like criteria for selection of s in linear system (colored curves refer BIC curves for different datasets; dotted vertical lines refer the position of smallest BIC for different datasets)

We first apply DFS with different choices of s's, and use a BIC-like criteria for high dimensional model [70, 71], which is defined as $BIC = n \cdot \log \hat{\sigma}^2 + c \cdot s \cdot \log n$, where $\hat{\sigma}^2$ is the MSE of the model, and c is set to be 3, to select an optimal s. For the first 10 datasets out 100, the selection of s is shown in Figure 4.2. For given s, it costs about 1.65 minutes for each run on a single CPU@2.6GHz with single thread called. For comparison, we have also applied the popular linear variable selection algorithms, including Lasso [45] and LARS [72], elastic net [46], and SCAD [47]. Lasso and elastic net are tested using existing python packages *'sklearn'* [73]. SCAD is performed by an existing R package *'ncvreg'* [74]. For all algorithms, the best model is selected by BIC-like criteria. That is a sequence of hyper-parameters will be tested on each methods and the one returns the lowest BIC value will be adopted as the best hyper-parameter.

Table 4.1. Comparison Table of DFS (Deep Feature Selection), LASSO, Elastic Net and SCAD (Smoothly Clipped Absolute Deviation) in Linear Variable Selection. $|\hat{\mathcal{S}}|$ denotes the average number of variable selected for 100 datasets; the number in parentheses denote the standard deviations of the corresponding values.

Methods	$\lvert\hat{\mathcal{S}}\rvert$	fsr	nsr	training MSE	testing MSE
DFS(two-step)	**99.53 (4.24)**	0.0235	0.0286	**1.6620(1.75)**	5.6496 (7.24)
LASSO	121.74(5.95)	0.1819	**0.0041**	2.4124(0.39)	**4.2604(0.77)**
Elastic Net	127.22 (7.44)	0.2173	0.0042	2.6095(0.55)	4.7369(0.99)
SCAD	**99.08 (1.21)**	**0.0029**	0.0121	4.0373(0.99)	6.2901(1.74)

Table 4.1 displays the average number of selected features and average *fsr* and *nsr* across 100 datasets. DFS method has selected very close to 100 variables with pretty small fsr and nsr. In practice, we recommend a two-step procedure for prediction because that the tuning parameters λ_1 and λ_2 are good for selection but may not be good for prediction and DFS algorithm has early stop criteria which cause the optimization procedure is not exhausted. So the training and testing error reported in Table 4.1 is based on two-step procedure. Lasso and elastic net have chosen much more variables than needed with a high *fsr*. SCAD has performed the best among

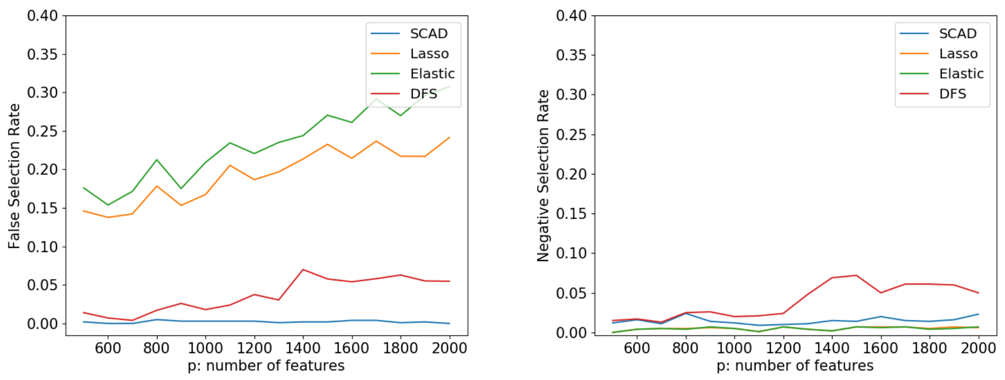


Fig. 4.3. Numerical results of false selection rate and negative selection rate between different linear variable selection methods. (left) false selection rate of SCAD, Lasso, Elastic Net and DFS when p increases. (right) negative selection rate of SCAD, Lasso, Elastic Net and DFS when p increase

three linear methods in terms of feature selection. However, SCAD dose not perform well in term of prediction under BIC model selection schemes.

Next, we also generate another 10 datasets from (4.15) with different choices of p varying from 500 to 2000. Let s varies from 90 to 110. The *fsr* and *nsr* of different algorithms are shown in Figure 4.3. For the false selection rate, DFS and SCAD remains at lower level compare with Lasso and Elastic Net. And as p increase, fsr of Lasso and Elastic Net increases significantly. For the negative selection rate, although DFS performs highest nsr, but remains at a very low level with other three methods. The increase of *nsr* and *fsr* of DFS might be due to the increase of the neural network complexity when p increases, which leads to a great presentation power and a larger possibility of overfitting. Reducing the complexity of approximation layers with a finer tuned λ_1 and λ_2 might help decrease the *nsr*.

4.5.2 A Nonlinear Classification Example

In this section, consider a high-dimensional nonlinear classification example. 30 datasets were generated from the nonlinear system [60]

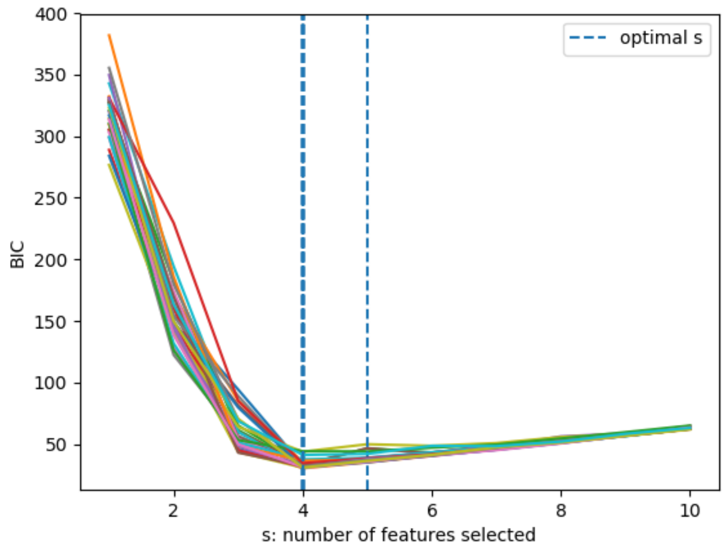


Fig. 4.4. Nonlinear Classification Example: BIC-like criteria for selection of s in nonlinear system(colored curves refer BIC curves for different datasets; dotted vertical lines refer the position of smallest BIC for different datasets)

$$y = \begin{cases} 1, & e^{x_1} + x_2^2 + 5\sin(x_3 x_4) - 3 > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{4.17}$$

where the variables $x_1, \dots, x_4$ together with additional 496 variables were generated in:

$$x_i = (e + z_i)/2, \quad i = 1, \dots, P, \tag{4.18}$$

where e and z_i are independently generated from $\mathcal{N}(0, 1)$. That is, the variables $x_1, \dots, x_p$ are mutually correlated with a correlation coefficient of 0.5. Each dataset consists of 600 observations where 300 observations have a response value of 1 and others have a response value of 0. For each dataset, we used 300 observations as the training data, and the remaining for the testing. The number of 0-1 are balanced in both training and testing set.

We again used a BIC-like criteria to select s using the training set. The BIC is defined as: $BIC = -2 \cdot \log \hat{L} + s \cdot \log n$, where $\hat{L}$ is the maximized value of the likelihood function of the model. For given s, it takes DFS 5.57 minutes on average with single CPU@2.6GHz and single thread. Figure 4.4 shows that the tuned s is 4 in 29 out of 30 datsets, which indicates that DFS has successfully handled the nonlinear relationship. We compare DFS with SCAD [47], the generalized additive model (GAM) [75], random forest (RF) [76], Bayesian adaptive regression trees (BART) [77, 78], and Bayesian neural network (BNN) [60]. GAM provides a penalized likelihood approach for fitting sparse generalized additive models in high dimension. Both RF and BART are regression tree-based methods, which can yield a flexible model capturing nonlinearities and interaction effects in the unknown regression function. The BRNN and BNN are both bayesian methods. Table 4.2 shows the average number of features selected, *fsr*, *nsr* and the training and testing error. The training and testing error is defined as the percentage of mis-classified samples in training set and testing set. The training error and testing error is reported based on two-step procedure. The one-step model in this case also provides very decent training error (5.19%) and testing error (8.38%). The comparison indicates that DFS outperforms all other methods in terms of variable selection. Among the five methods, DFS achieves the lowest *fsr* and *nsr*. DFS has achieved the exact recovery of support on 25 out of 30 datasets. At the same time, DFS also achieves the top training and testing error with two-step procedure. We also run 100 replicas on DFS without carefully tuned hyper-parameters. DFS selected 4.09 (0.29) variables with false selection rate and negative selection rate equal 0.0782 and 0.0575 respectively.

4.6 Real Data Analysis

4.6.1 MNIST

In this section, we apply DFS on the handwritten digits database, MNIST. The database has a training set of $60,000$ examples, and a test set of $10,000$ examples. The

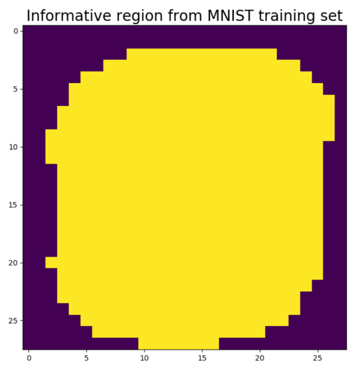
Table 4.2. Comparison Table of DFS (Deep Feature Selection), BNN (Bayesian Neural Networks), GAM (generalized additive model), RF (random Forest), BART (Bayesian adaptive regression trees) and SCAD (Smoothly Clipped Absolute Deviation) in non-linear variable selection and prediction. $|\hat{\mathcal{S}}|$ denotes the average number of variable selected for 10 datasets; training and testing error denotes the percentage of mis-classified samples in the corresponding datasets; the number in parentheses denote the standard deviations of the corresponding values.

Methods	$\|\hat{\mathcal{S}}\|$	fsr	nsr	Training Error	Testing Error
DFS (two-step)	**4.03(0.18)**	**0.049**	**0.042**	**0.08 % (0.002)**	**5.72 % (0.03)**
BNN	**4.3 (0.26)**	0.093	**0.025**	3.57 % (0.47)	**9.70 % (0.99)**
GAM	13.5 (3.68)	0.730	0.100	12.13 % (1.34)	15.57 % (1.97)
RF	5.2 (0.39)	0.310	0.100	**0.0 % (0.0)**	14.30 % (1.04)
BART	3.3 (0.33)	**0.030**	0.200	4.47 % (0.55)	16.90 (1.58) %
SCAD	5.0 (1.45)	0.460	0.325	18.70 % (2.91)	21.43 % (2.48)

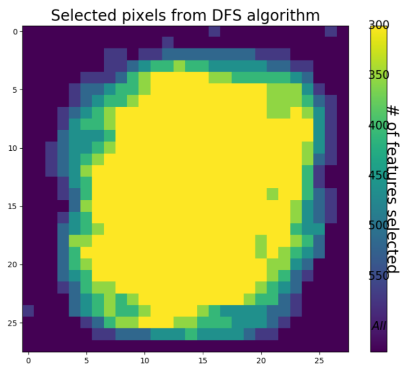
digits have been size-normalized and centered in a fixed-size image. Each handwritten digit is of dimension 28×28, and was labeled from 0 to 9. This serves as a multi-classification problem.

Since all images are centered, the edge of each image is expected to be non-informative. We first use the one hot encoding version of the MNIST database (the values of pixels are either 0 or 1), and sum up all $60,000$ training samples for each individual pixel location. If the sum at a pixel location has value under 100, we naively believe that this pixel is not informative for classification, and the rest may be informative. Such region consists of 550 pixels and is shown in Figure 4.5.

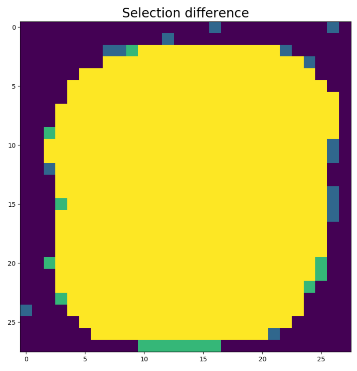
We run our DFS algorithm on the MNIST training data to make the variable selection. Selection regions with different choices of pixel numbers, s from 300 to 550, are shown in Figure 4.5b. It is interesting to compare the informative region (Figure 4.5a) with the selected region with $s = 550$ from our algorithm in Figure 4.5c. These two regions almost exactly match with each other.



(a) Informative region from cumulated one-hot encoding training samples



(b) Region selected by DFS method with different number of pixels s to be selected



(c) Region difference between informative region and 550 selected pixels from DFS

Fig. 4.5. MNIST experiments

We train a simple one hidden layer neural network (800 hidden unit) with all pixels as well as with the pixels selected by our algorithm only. The purpose of this experiment is to demonstrate superior pixel selection performance rather than the prediction performance of a well-tuned model. We control all other parameters, including training steps, learning rate, the same for all models. The performance on testing set is performed in Table 4.3.

Table 4.3. MNIST Experiments: Testing accuracy of simple neural networks applying DFS on support with different number of pixels s

# of s	All pixels (784)	550	500	450	**400**	350	300
Accuracy	98.09%	98.56%	98.53%	98.45%	**98.61%**	98.45%	98.22%

Table 4.3 suggests, on testing set, the same model works better with selected pixels instead of all pixels. The region with 400 pixels yields the highest accuracy on test data. The highest accuracy also beats the benchmark, with 1.6% error, on MNIST website for 2-layer neural network with 800 hidden unit. Besides the benefit of higher accuracy, our algorithm also saves up to 1/3 of the computing time for training model with less features.

4.6.2 Selection of Drug Sensitive Genes

In this section, we apply DFS on the data from a cancer cell line encyclopedia (CCLE) study to identify genes that are associated with anticancer drug sensitivities. The CCLE dataset consists of 8-point dose-response curves for 24 chemical compounds (drugs) across over 400 cell lines, which is aviable at `www.broadinstitute.org/ccle`. For each cell line, it consists of the expression data of 18,926 genes. We use the area under the dose-response curve, which is termed as *activity area* in [79], as our response to measure the sensitivity of a drug for each cell line. Our goal is to use DFS to identify the genes that are associated with the drug sensitivity for each drug. We select 3 out of 24 drugs whose drug sensitivity gene has been verified by other studies for the testing. Before running DFS, we apply marginal feature screening [80] to reduce the number of genes to around 90. Slightly less than 500 observations (cell lines) are collected for each drug. DFS returns 4 to 6 relevant genes for each drug. The selected genes are given in Table 4.4.

Table 4.4 indicates that DFS selections are consistent with these verified genes. Specifically, for 17-AAG, [81] and [79] reported NQO1 as the top predictive biomarker

Table 4.4. Drug-sensitive genes identified by DFS. Verified gene denotes the verified gene by medical and biological studies; Selected gene denotes the genes selected by DFS.

Drug	Verified Gene	Selected Gene
17-AAG	NQO1	**NQO1**, MMP24, ATP6V0E1, SEC23IP, INO80, C9f37
Paclitaxel	BCL2L1	**BCL2L1**, SSRP1, ZNRD1, SLC35F5
Topotecan	SLFN11	**SLFN11**, ELAVL1, HMGB2, CD63

for it. For the drug Paclitaxel, it has been confirmed that BCL2L1 is predictive of treatment response in [82, 83]. For the drug Topotecan, both [79] and [84] reported that SLFN11 is the predictive of treatment response. Bayesian Neural Network method has provided very similar results but need much more computing resources.

4.7 Concluding Remarks

This chapter studies high-dimensional nonlinear variable selection via deep neural networks. This problem is converted into a sparsity-constrained optimization. We introduce a greedy algorithm called the Deep Feature Selection (DFS). Under the condition of a Generalized Stable Restricted Hessian, we provide theoretical convergence guarantees and establish the selection consistency. Further, DFS outperforms many other algorithms on a variety of numerical examples.

There are several extensions we can pursue for future work. It will be interesting to study the capacity of proposed networks based on Rademacher complexities. This can guide us to control the generalization error to avoid the overfitting. There are many hyperparameters, such as the depth and width of the network, which need to be fine-tuned. It is interesting to apply the Bayesian optimization [85] technique to automatically select these hyperparameters. Let $J(f) = \mathbb{E}_{(\boldsymbol{x},y)}[l(f)]$. The approximation error is defined as $\inf_{f\in\mathcal{H}_{s,d}} J(f) - \inf_{f\in\mathcal{F}_s} J(f)$, which can be bounded using Theorem 2.1. On the other hand, the generalization error is defined as $\sup_{f\in\mathcal{H}_{s,d}} |J(f) - l(f)|$, which is typically bounded by the Rademacher complexity of function class $\mathcal{H}_{s,d}$ [86]. We leave this for future research. The approximation error and the generalization error add up to excess risk.

Characterizing the uncertainty for the high-dimensional variable selection is another challenging task. Combining variable selection with the popular variational auto-encoders [7] or generative adversarial networks [8] may provide promising answers to this question.

4.A Proof of Theorem 4.2.1

In this section, we would like to approximate function f by our proposed deep neural network h with one selection layer and one approximation layer. Explicitly speaking, $h : \boldsymbol{R}^p \to \boldsymbol{R}$ has the following structure:

$$h(\boldsymbol{x}) = \sum_{i=1}^{l} \Theta_i^{(2)} \sigma\left(\Theta_{i,\cdot}^{(1)}(\boldsymbol{w} * \boldsymbol{x}) + b_i^{(1)}\right) \quad \forall \boldsymbol{x} \in \boldsymbol{R}^p,$$

where $*$ is the element-wise multiplication, $\boldsymbol{w} \in \boldsymbol{R}^p$ is the sparse parameter for the selection layer, $\Theta_{i,\cdot}^{(1)} \in \boldsymbol{R}^p$ is the i-th row of $\Theta^{(1)}$, $b_i^{(1)} \in \boldsymbol{R}$ and $\Theta_i^{(2)} \in \boldsymbol{R}$ are the i-th element of $b^{(1)}$ and $\Theta^{(2)}$ respectively, for $i = 1, \ldots, l$. We set $\|\boldsymbol{w}\|_0 = s$ and $\|\boldsymbol{w}\|_2 \leq c_1$ to perform variable selection. Without loss of generality and by homogeneity of σ, we may assume $\|(\Theta_{i,\cdot}^{(1)}, b_i^{(1)})\|_2 = c_2$, for $i = 1, \ldots, k$. The sparsity of $\boldsymbol{w}$ results in $(\boldsymbol{w} * \boldsymbol{x})^\top = ((\tilde{\boldsymbol{w}} * \tilde{\boldsymbol{x}})^\top, \boldsymbol{0}^\top)$ after rearranging the position of $\boldsymbol{x}$. Here $\tilde{\boldsymbol{w}}$ stands for the sub-vector of $\boldsymbol{w}$ with nonzero values. This also implies $\Theta_{i,\cdot}^{(1)} = (\tilde{\Theta}_{i,\cdot}^{(1)}, \boldsymbol{0}^\top)$ because of the feed-forward property of neural network, where $\tilde{\Theta}_{i,\cdot}^{(1)} \in \boldsymbol{R}^s$ is the weight corresponding to the neurons $\tilde{\boldsymbol{w}} * \tilde{\boldsymbol{x}}$. Thus, we can find a function $\bar{h} : \boldsymbol{R}^s \to \boldsymbol{R}$ such that $h(\boldsymbol{x}) = \bar{h}(\tilde{\boldsymbol{x}})$ by rewriting σ as a function of $\tilde{\boldsymbol{x}}$:

$$\sigma\Big(\Theta^{(1)}_{i,\cdot}(\boldsymbol{w}*\boldsymbol{x})+b^{(1)}_i\Big)=\sigma\left((\tilde{\Theta}^{(1)}_{i,\cdot},\boldsymbol{0}^\top)\begin{pmatrix}\tilde{\boldsymbol{w}}*\tilde{\boldsymbol{x}}\\ \boldsymbol{0}\end{pmatrix}+b^{(1)}_i\right)=\sigma\Big(\tilde{\Theta}^{(1)}_{i,\cdot}(\tilde{\boldsymbol{w}}*\tilde{\boldsymbol{x}})+b^{(1)}_i\Big),$$

that is to say, for any $\tilde{\boldsymbol{x}}\in\boldsymbol{R}^s$, $\bar{h}(\tilde{\boldsymbol{x}})=\sum_{i=1}^{l}\Theta^{(2)}_i\sigma(\tilde{\Theta}^{(1)}_{i,\cdot}(\tilde{\boldsymbol{w}}*\tilde{\boldsymbol{x}})+b^{(1)}_i)$ satisfying $\|\tilde{\boldsymbol{w}}\|_2\le c_1$ and $\|(\tilde{\Theta}^{(1)}_{i,\cdot},b^{(1)}_i)\|_2=c_2$, for $i=1,\ldots,l$.

To approximate $\bar{f}(\tilde{\boldsymbol{x}})$ by $\bar{h}(\tilde{\boldsymbol{x}})$, we first introduce a space of functions, in which functions can be represented by an uncountable number of basis. For the theoretical analysis, we assume that $\tilde{\boldsymbol{x}}\in\boldsymbol{R}^s$ are almost surely bounded by R in l_2-norm. We then augment the variable $\tilde{\boldsymbol{x}}$ to $\tilde{\boldsymbol{z}}=(\tilde{\boldsymbol{x}}^\top,R)^\top\in\boldsymbol{R}^{s+1}$ by appending the constant R to $\tilde{\boldsymbol{x}}$. This leads to $\|\tilde{\boldsymbol{z}}\|_2\le\sqrt{2}R$. By defining the vector $\boldsymbol{v}_i=(\tilde{\Theta}^{(1)}_{i,\cdot}*\tilde{\boldsymbol{w}}^\top,\frac{b^{(1)}_i}{R})^\top\in\boldsymbol{R}^{s+1}$, we may write the neurons in the approximation layer as

$$\phi_{\boldsymbol{v}_i}(\tilde{\boldsymbol{x}})=\sigma\Big(\tilde{\Theta}^{(1)}_{i,\cdot}(\tilde{\boldsymbol{w}}*\tilde{\boldsymbol{x}})+b^{(1)}_i\Big)=\sigma\Big((\tilde{\Theta}^{(1)}_{i,\cdot}*\tilde{\boldsymbol{w}}^\top)\tilde{\boldsymbol{x}}+b^{(1)}_i\Big)=\sigma(\boldsymbol{v}_i^\top\tilde{\boldsymbol{z}})=(\boldsymbol{v}_i^\top\tilde{\boldsymbol{z}})_+,$$

which now turn to a function of $\tilde{\boldsymbol{z}}$. Without loss of generality, we may further suppose that the l_2-norm of each vector $\boldsymbol{v}$ is equal to $1/R$. This means the space of $\boldsymbol{v}$, denoted as $\mathcal{V}$, is the $1/R$-sphere for the l_2-norm. We also assume that for any given $\tilde{\boldsymbol{x}}\in\boldsymbol{R}^s$, the functions $\boldsymbol{v}\to\phi_{\boldsymbol{v}}(\tilde{\boldsymbol{x}})$ are continuous. Based on the set of units, we define our space of functions $\mathcal{Q}_1$ from $\boldsymbol{R}^s$ to $\boldsymbol{R}$, in which function q can be decomposed as $q(\tilde{\boldsymbol{x}})=\int_{\mathcal{V}}\phi_{\boldsymbol{v}}(\tilde{\boldsymbol{x}})d\mu(\boldsymbol{v})$, where μ is a signed Radon measure on $\mathcal{V}$. The norm $\alpha_1(q)$ equipped for $\mathcal{Q}_1$ is defined by the infimum of the total variation $|\mu|(\mathcal{V})$ over all decomposition of $q(\tilde{\boldsymbol{x}})$. If the signed measure μ has a density with respect to a fixed probability measure τ with full support on $\mathcal{V}$, that is, $d\mu(\boldsymbol{v})=p(\boldsymbol{v})d\tau(\boldsymbol{v})$, then the norm $\alpha_1(q)$ is equal to the smallest value of $|\mu|(\mathcal{V})=\int_{\mathcal{V}}|p(\boldsymbol{v})|d\tau(\boldsymbol{v})$ over all integrable functions p such that $q(\tilde{\boldsymbol{x}})=\int_{\mathcal{V}}\phi_{\boldsymbol{v}}(\tilde{\boldsymbol{x}})p(\boldsymbol{v})d\tau(\boldsymbol{v})$.

Similarly, we may also define a squared norm $\alpha_2^2(q)$ as the infimum of $\int_{\mathcal{V}}|p(\boldsymbol{v})|^2d\tau(\boldsymbol{v})$ over the same decomposition. The function space in which $q(\tilde{\boldsymbol{x}})=\int_{\mathcal{V}}\phi_{\boldsymbol{v}}(\tilde{\boldsymbol{x}})p(\boldsymbol{v})d\tau(\boldsymbol{v})$, equipped with norm $\alpha_2(q)$ is denoted as $\mathcal{Q}_2$. Due to Jensen's inequality, it is straightforward that $\alpha_2(q)$ is greater than $\alpha_1(q)$. Therefore, $\mathcal{Q}_2$ is included in $\mathcal{Q}_1$.

Lemma 1 *For δ larger than a constant that depends only on s, for any function $\bar{f} : \boldsymbol{R}^s \to \boldsymbol{R}$ such that for all $\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{y}} \in \boldsymbol{R}^s$ satisfying $\|\tilde{\boldsymbol{x}}\|_2 \leq R$ and $\|\tilde{\boldsymbol{y}}\|_2 \leq R$, $|\bar{f}(\tilde{\boldsymbol{x}})| \leq \eta$ and $|\bar{f}(\tilde{\boldsymbol{x}}) - \bar{f}(\tilde{\boldsymbol{y}})| \leq \eta R^{-1} \|\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}}\|_2$, there exists a neural network $\bar{q}(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{2+2k} \Theta_i^{(2)} \sigma((\tilde{\Theta}_{i,\cdot}^{(1)} * \tilde{\boldsymbol{w}}^\top)\tilde{\boldsymbol{x}} + b_i^{(1)})$ with $\left\|\Theta^{(2)}\right\|_2 \leq 2 + \delta$ and $\|(\tilde{\Theta}_{i,\cdot}^{(1)} * \tilde{\boldsymbol{w}}^\top, b_i^{(1)}/R)\|_2 = 1/R$, for $i = 1, \ldots, 2+2k$, that can approximate $\bar{f}(\tilde{\boldsymbol{x}})$ with uniform error:*

$$\left|\bar{f}(\tilde{\boldsymbol{x}}) - \bar{q}(\tilde{\boldsymbol{x}})\right| \leq C(s)\eta(\frac{\delta}{\eta})^{-2/(s+1)} \log(\frac{\delta}{\eta}) + C(s)\delta k^{-(s+3)/2s},$$

where $C(s)$ is a constant related to s.

Proof The proof can be accomplished by two parts. We first approximate this ηR^{-1}-Lipschitz-continuous function $\bar{f}$ by a function q in the function space $\mathcal{Q}_2$, then use a network with $2 + 2k$ neurons to approximate q.

The first part has been proved by [65], that is, for this ηR^{-1}-Lipschitz-continuous function $\bar{f}(\tilde{\boldsymbol{x}})$, there exists a function $q(\tilde{\boldsymbol{x}}) \in \mathcal{Q}_2$, such that $\alpha_2(q) \leq \delta$ and for any $\|\tilde{\boldsymbol{x}}\|_2 \leq R$,

$$|q(\tilde{\boldsymbol{x}}) - \bar{f}(\tilde{\boldsymbol{x}})| \leq C(s)\eta(\frac{\delta}{\eta})^{-2/(s+1)} \log(\frac{\delta}{\eta}). \tag{4.19}$$

We will use the results in [87] to prove the second part, which shows that for any probability measure μ (positive and with finite mass) on the sphere $\boldsymbol{S}^s$ and any $\epsilon \in (0, \frac{1}{2})$, there exists a set of m points $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$, such that for all $\boldsymbol{u} \in \boldsymbol{S}^s$,

$$\Big|\int_{\boldsymbol{S}^s} |\boldsymbol{v}^\top \boldsymbol{u}| d\mu(\boldsymbol{v}) - \frac{1}{m}\sum_{i=1}^m |\boldsymbol{v}_i^\top \boldsymbol{u}|\Big| \leq \epsilon \tag{4.20}$$

with $m \leq C(s)\epsilon^{-2+6/(s+3)} = C(s)\epsilon^{-2s/(s+3)}$, for some constant $C(s)$ that depends only on s.

Since $\mathcal{Q}_2 \subset \mathcal{Q}_1$, we also have $q(\tilde{\boldsymbol{x}}) \in \mathcal{Q}_1$, and we rewrite it as a function of $\tilde{\boldsymbol{z}}$:

$$q(\tilde{\boldsymbol{x}}) = \int_{\boldsymbol{S}^s} \sigma\Big((\tilde{\Theta}_{;\cdot}^{(1)} * \tilde{\boldsymbol{w}}^\top)\tilde{\boldsymbol{x}} + b\Big) d\mu(\boldsymbol{v}) = \int_{\boldsymbol{S}^s} \sigma(\boldsymbol{v}^\top \tilde{\boldsymbol{z}}) d\mu(\boldsymbol{v}) = \int_{\boldsymbol{S}^s} (\boldsymbol{v}^\top \tilde{\boldsymbol{z}})_+ d\mu(\boldsymbol{v}) \triangleq r(\tilde{\boldsymbol{z}}).$$

Furthermore, because $(\boldsymbol{v}^\top \tilde{\boldsymbol{z}})_+ = \boldsymbol{v}^\top \tilde{\boldsymbol{z}}/2 + |\boldsymbol{v}^\top \tilde{\boldsymbol{z}}|/2$, and $\mu = \mu_+ - \mu_-$, where μ_+ and μ_- are positive measures, $r(\tilde{\boldsymbol{z}})$ can be decomposed as:

$$r(\tilde{\boldsymbol{z}}) = \frac{1}{2}\int_{\mathcal{S}^s} (\boldsymbol{v}^\top \tilde{\boldsymbol{z}}) d\mu(\boldsymbol{v}) + \frac{\mu_+(\boldsymbol{S}^s)}{2}\int_{\mathcal{S}^s} |\boldsymbol{v}^\top \tilde{\boldsymbol{z}}| \frac{d\mu_+(\boldsymbol{v})}{\mu_+(\boldsymbol{S}^s)} - \frac{\mu_-(\boldsymbol{S}^s)}{2}\int_{\mathcal{S}^s} |\boldsymbol{v}^\top \tilde{\boldsymbol{z}}| \frac{d\mu_-(\boldsymbol{v})}{\mu_-(\boldsymbol{S}^s)}.$$

The first term of $r(\tilde{\boldsymbol{z}})$ is a liner function $\boldsymbol{t}^\top\tilde{\boldsymbol{z}}$ of $\tilde{\boldsymbol{z}}$, with $\|\boldsymbol{t}\|_2 \le 1$. We may write it as the sum of two units:

$$\boldsymbol{t}^\top\tilde{\boldsymbol{z}} = \|\boldsymbol{t}\|_2 \left(\left(\frac{\boldsymbol{t}}{\|\boldsymbol{t}\|_2}\right)^\top \tilde{\boldsymbol{z}}\right)_+ + (-\|\boldsymbol{t}\|_2)\left(\left(\frac{-\boldsymbol{t}}{\|\boldsymbol{t}\|_2}\right)^\top \tilde{\boldsymbol{z}}\right)_+.$$

For the second term of $r(\tilde{\boldsymbol{z}})$, we may approximate it with k_1 term. Following (4.20), we have

$$\left|\frac{\mu_+(\boldsymbol{S}^s)}{2}\int_{\boldsymbol{S}^s}\left|\boldsymbol{v}^\top \frac{\tilde{\boldsymbol{z}}}{\|\tilde{\boldsymbol{z}}\|_2}\right|\frac{d\mu_+(\boldsymbol{v})}{\mu_+(\boldsymbol{S}^s)} - \frac{\mu_+(\boldsymbol{S}^s)}{2}\frac{1}{k_1}\sum_{i=1}^{k_1}\left|\boldsymbol{v}_i^\top \frac{\tilde{\boldsymbol{z}}}{\|\tilde{\boldsymbol{z}}\|_2}\right|\right| \le \epsilon\mu_+(\boldsymbol{S}^s).$$

This implies

$$\left|\frac{\mu_+(\boldsymbol{S}^s)}{2}\int_{\boldsymbol{S}^s}|\boldsymbol{v}^\top\tilde{\boldsymbol{z}}|\frac{d\mu_+(\boldsymbol{v})}{\mu_+(\boldsymbol{S}^s)} - \frac{\mu_+(\boldsymbol{S}^s)}{2}\frac{1}{k_1}\sum_{i=1}^{k_1}|\boldsymbol{v}_i^\top\tilde{\boldsymbol{z}}|\right| \le \epsilon\mu_+(\boldsymbol{S}^s)\,\|\tilde{\boldsymbol{z}}\|_2 \le \sqrt{2}R\epsilon\mu_+(\boldsymbol{S}^s).$$

We then write $|\boldsymbol{v}^\top\tilde{\boldsymbol{z}}|$ as $|\boldsymbol{v}^\top\tilde{\boldsymbol{z}}| = (\boldsymbol{v}^\top\tilde{\boldsymbol{z}})_+ + (-\boldsymbol{v}^\top\tilde{\boldsymbol{z}})_+$, which leads to an approximation of $\sqrt{2}R\epsilon\mu_+(\boldsymbol{S}^s)$ with $2k_1$ units, where $k_1 \le C(s)\epsilon^{-2s/(s+3)}$. Similarly, we can approximate the last term with $2k_2$ units for an approximation error of $\sqrt{2}R\epsilon\mu_-(\boldsymbol{S}^s)$, where $k_2 \le C(s)\epsilon^{-2s/(s+3)}$. Let $\bar{q}(\tilde{\boldsymbol{x}})$ denote the neuron network composed of these $2+2k$ units, that is, $\bar{q}(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{2+2k}\Theta_i^{(2)}\sigma((\tilde{\Theta}_{i,\cdot}^{(1)} * \tilde{\boldsymbol{w}}^\top)\tilde{\boldsymbol{x}} + b_i^{(1)})$. Thus,

$$|q(\tilde{\boldsymbol{x}}) - \bar{q}(\tilde{\boldsymbol{x}})| \le \sqrt{2}R\epsilon\mu_+(\boldsymbol{S}^s) + \sqrt{2}R\epsilon\mu_-(\boldsymbol{S}^s) = \sqrt{2}R\epsilon\alpha_1(q), \tag{4.21}$$

where $k = k_1 + k_2$ satisfying $k \le C(s)\epsilon^{-2s/(s+3)}$. Moreover, the output weights $\Theta^{(2)}$ satisfying:

$$\|\Theta^{(2)}\|_2 \le \|\Theta^{(2)}\|_1 = 2\,\|\boldsymbol{t}\|_2 + \mu_+(\boldsymbol{S}^s) + \mu_-(\boldsymbol{S}^s) = 2\,\|\boldsymbol{t}\|_2 + \alpha_1(q) \le 2 + \delta,$$

since $\alpha_1(q) \le \alpha_2(q) \le \delta$.

Combining the (4.19) and (4.21) together, we have

$$|\bar{f}(\tilde{\boldsymbol{x}}) - \bar{q}(\tilde{\boldsymbol{x}})| \le C(s)\eta(\frac{\delta}{\eta})^{-2/(s+1)}\log(\frac{\delta}{\eta}) + C(s)\delta k^{-(s+3)/2s},$$

where $\left\|\Theta^{(2)}\right\|_2 \le 2+\delta$ and $\|(\tilde{\Theta}_{i,\cdot}^{(1)} * \tilde{\boldsymbol{w}}^\top, b_i^{(1)}/R)\|_2 = 1/R$, for $i = 1, \ldots, 2+2k$. ■

A.1 Proof of Theorem 4.2.1

Proof For the uniform error in Lemma 1, we can optimize over δ to obtain a uniform approximation bound proportional to $\eta k^{-1/s} \log k$ by using $\delta = \eta k^{(s+1)/(2s)}$. Combining this and Lemma 1, we may approximate $\bar{f}(\tilde{\boldsymbol{x}})$ by a $\bar{h}(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{2+2k} \Theta_i^{(2)} \sigma(\tilde{\Theta}_{i,\cdot}^{(1)}(\tilde{\boldsymbol{w}} * \tilde{\boldsymbol{x}}) + b_i^{(1)})$ with $\left\|\Theta^{(2)}\right\|_2 \leq C(s, \eta)$. By writing $\Theta_{i,\cdot}^{(1)} = (\tilde{\Theta}_{i,\cdot}^{(1)}, \boldsymbol{0}^\top)$ and $\boldsymbol{w}^\top = (\tilde{\boldsymbol{w}}^\top, \boldsymbol{0}^\top)$, we have $h(\boldsymbol{x}) = \bar{h}(\tilde{\boldsymbol{x}})$, satisfying $\|\boldsymbol{w}\|_0 = s$ and $\left\|\Theta^{(2)}\right\|_2 \leq C(s, \eta)$. Thus, we can conclude that

$$|f(\boldsymbol{x}) - h(\boldsymbol{x})| \leq C(s)\eta k^{-1/s} \log k.$$

■

4.B Proof of section 4.4

Let $\boldsymbol{v}|_I$ denote restriction of vector $\boldsymbol{v}$ to the rows indicated by indices in I, or a vector that equals $\boldsymbol{v}$ except for coordinates in I^c where it is zero depending on the context. Let $\boldsymbol{P}_I$ denote the restriction of identity matrix to the columns indicated by I. To be convenient, we also denote $\alpha_k(\boldsymbol{p}, \boldsymbol{q}) = \int_0^1 A_k(t\boldsymbol{q} + (1-t)\boldsymbol{q})dt$, $\beta_k(\boldsymbol{p}, \boldsymbol{q}) = \int_0^1 B_k(t\boldsymbol{q} + (1-t)\boldsymbol{q})dt$, and $\gamma_k(\boldsymbol{p}, \boldsymbol{q}) = \alpha_k(\boldsymbol{p}, \boldsymbol{q}) - \beta_k(\boldsymbol{p}, \boldsymbol{q})$, where $A_k(\cdot)$ and $B_k(\cdot)$ are defined by (4.6) and (4.7) respectively.

Proposition 1 *Let $\boldsymbol{Q}(t)$ be a matrix-valued function such that for all $t \in [0, 1]$, $\boldsymbol{Q}(t)$ is symmetric and its eigenvalues lie in interval $[B(t), A(t)]$ with $B(t) > 0$. Then for any vector $\boldsymbol{v}$ we have*

$$\Big(\int_0^1 B(t)dt\Big) \|\boldsymbol{v}\|_2 \leq \left\| \Big(\int_0^1 \boldsymbol{Q}(t)dt\Big)\boldsymbol{v} \right\|_2 \leq \Big(\int_0^1 A(t)dt\Big) \|\boldsymbol{v}\|_2$$

Proposition 2 *Let $\boldsymbol{Q}(t)$ be a matrix-valued function such that for all $t \in [0, 1]$, $\boldsymbol{Q}(t)$ is symmetric and its eigenvalues lie in interval $[B(t), A(t)]$ with $B(t) > 0$. If Γ is a subset of row/column indices of $\boldsymbol{Q}(t)$ then for any vector $\boldsymbol{v}$ we have*

$$\left\| \Big(\int_0^1 \boldsymbol{P}_\Gamma^\top \boldsymbol{Q}(t) \boldsymbol{P}_{\Gamma^c} dt\Big)\boldsymbol{v} \right\|_2 \leq \Big(\int_0^1 \frac{A(t) - B(t)}{2} dt\Big) \|\boldsymbol{v}\|_2$$

The detailed proof of Proposition 1 and Proposition 2 can refer to [67], so we omit here.

Lemma 2 *Let $\mathcal{R}_{\boldsymbol{w}}$ denote the set supp$(\hat{\boldsymbol{w}} - \boldsymbol{w}^\star)$. The current estimate $(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}})$ then satisfies*

$$\begin{aligned}
&\|(\hat{\boldsymbol{w}} - \boldsymbol{w}^\star)|_{\mathcal{Z}^c}\|_2 \\
\leq &\frac{\gamma_{4s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)) + \gamma_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}{2\beta_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))} \left\|(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 \\
+ &\frac{\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}}\right\|_2 + \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{Z} \setminus \mathcal{R}_{\boldsymbol{w}}}\right\|_2}{\beta_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}.
\end{aligned}$$

Proof For simplicity, we re-write $(\boldsymbol{z}_{\boldsymbol{w}}, \boldsymbol{z}_{\boldsymbol{\Theta}})$ as $\boldsymbol{z}$ in the following proof. Since $\mathcal{Z} =$ *supp*$(\boldsymbol{z}_{\boldsymbol{w}_{2s}})$ and $|\mathcal{R}_{\boldsymbol{w}}| \leq 2s$, we have $\left\|\boldsymbol{z}|_{\mathcal{R}_{\boldsymbol{w}}}\right\|_2 \leq \|\boldsymbol{z}|_{\mathcal{Z}}\|_2$. Thus,

$$\left\|\boldsymbol{z}|_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}}\right\|_2 \leq \left\|\boldsymbol{z}|_{\mathcal{Z} \setminus \mathcal{R}_{\boldsymbol{w}}}\right\|_2. \tag{4.22}$$

According to the algorithm, $\boldsymbol{z} = \nabla g(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}})$, therefore,

$$\begin{aligned}
&\left\|\boldsymbol{z}|_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}}\right\|_2 \\
\geq &\left\|\nabla g(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}})|_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}} - \nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}}\right\|_2 - \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}}\right\|_2 \\
= &\left\|\Big(\int_0^1 \boldsymbol{P}^\top_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}} \boldsymbol{H}_g(t(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) + (1-t)(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))dt\Big)\big((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\big)\right\|_2 \\
- &\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}}\right\|_2 \\
\geq &\left\|\Big(\int_0^1 \boldsymbol{P}^\top_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}} \boldsymbol{H}_g(t(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) + (1-t)(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))\boldsymbol{P}_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}} dt\Big)\big((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\big)|_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}}\right\|_2 \\
- &\left\|\Big(\int_0^1 \boldsymbol{P}^\top_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}} \boldsymbol{H}_g(t(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) + (1-t)(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))\boldsymbol{P}_{(\mathcal{R}_{\boldsymbol{w}} \cap \mathcal{Z}) \cup \mathcal{R}_{\boldsymbol{\Theta}}} dt\Big)\big((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\big)|_{(\mathcal{R}_{\boldsymbol{w}} \cap \mathcal{Z}) \cup \mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2 \\
- &\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}}\right\|_2
\end{aligned}$$

By splitting $\mathcal{R}_{\boldsymbol{w}} \cup \mathcal{R}_{\boldsymbol{\Theta}}$ into two sets $\mathcal{R}_{\boldsymbol{w}} \setminus \mathcal{Z}$ and $(\mathcal{R}_{\boldsymbol{w}} \cap \mathcal{Z}) \cup \mathcal{R}_{\boldsymbol{\Theta}}$, then applying the triangle inequality, we have the last inequality. Using Proposition 1 and 2, we have

$$\begin{aligned}
\left\|\boldsymbol{z}|_{\mathcal{R}_{\boldsymbol{w}}\setminus\mathcal{Z}}\right\|_2 \geq\ & \beta_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)) \left\|\big((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\big)|_{\mathcal{R}_{\boldsymbol{w}}\setminus\mathcal{Z}}\right\|_2 \\
& - \frac{\gamma_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}{2} \left\|\big((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\big)|_{(\mathcal{R}_{\boldsymbol{w}}\cap\mathcal{Z})\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2 \\
& - \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}}\setminus\mathcal{Z}}\right\|_2 \\
\geq\ & \beta_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)) \left\|\big((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\big)|_{\mathcal{R}_{\boldsymbol{w}}\setminus\mathcal{Z}}\right\|_2 \\
& - \frac{\gamma_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}{2} \left\|(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 \\
& - \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}}\setminus\mathcal{Z}}\right\|_2
\end{aligned} \tag{4.23}$$

Similarly, by Proposition 2, we obtain

$$\begin{aligned}
& \left\|\boldsymbol{z}|_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}}\right\|_2 \\
\leq & \left\|\nabla g(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}})|_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}} - \nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}}\right\|_2 + \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}}\right\|_2 \\
= & \left\|\Big(\int_0^1 \boldsymbol{P}_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}}^\top \boldsymbol{H}_g(t(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) + (1-t)(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)) \boldsymbol{P}_{\mathcal{R}_{\boldsymbol{w}}\cup\mathcal{R}_{\boldsymbol{\Theta}}} dt\Big)\big((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\big)|_{\mathcal{R}_{\boldsymbol{w}}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2 \\
+ & \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}}\right\|_2 \\
\leq & \frac{\gamma_{4s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}{2} \left\|(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 + \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}}\right\|_2 .
\end{aligned} \tag{4.24}$$

Since $\left\|\big((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\big)|_{\mathcal{R}_{\boldsymbol{w}}\setminus\mathcal{Z}}\right\|_2 = \|(\hat{\boldsymbol{w}} - \boldsymbol{w}^\star)|_{\mathcal{Z}^c}\|_2$, combining (4.22), (4.23) and (4.24), we get

$$\begin{aligned}
& \frac{\gamma_{4s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}{2} \left\|(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 + \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}}\right\|_2 \\
\geq & \left\|\boldsymbol{z}|_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}}\right\|_2 \\
\geq & \left\|\boldsymbol{z}|_{\mathcal{R}_{\boldsymbol{w}}\setminus\mathcal{Z}}\right\|_2 \\
\geq & \beta_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)) \left\|(\hat{\boldsymbol{w}} - \boldsymbol{w}^\star)|_{\mathcal{Z}^c}\right\|_2 \\
- & \frac{\gamma_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}{2} \left\|(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 - \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}}\setminus\mathcal{Z}}\right\|_2
\end{aligned}$$

Therefore,

$$\begin{aligned}&\|(\hat{\boldsymbol{w}}-\boldsymbol{w}^\star)|_{\mathcal{Z}^c}\|_2\\ \le&\frac{\gamma_{4s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))+\gamma_{2s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}{2\beta_{2s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\left\|(\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}})-(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)\right\|_2\\ +&\frac{\left\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}}\setminus\mathcal{Z}}\right\|_2+\left\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{Z}\setminus\mathcal{R}_{\boldsymbol{w}}}\right\|_2}{\beta_{2s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}.\end{aligned}$$

■

Lemma 3 *The vector* $(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})$ *given by*

$$(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})=\arg\min_{\boldsymbol{w},\boldsymbol{\Theta}} g(\boldsymbol{w},\boldsymbol{\Theta}),\quad s.t.\quad \boldsymbol{w}|_{\mathcal{T}^c}=0 \tag{4.25}$$

satisfies

$$\begin{aligned}&\|(\boldsymbol{w}^\star|_{\mathcal{T}},\boldsymbol{\Theta}^\star)-(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\|_2\\ \le&\frac{\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\|_2}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}+\frac{\gamma_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}{2\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\|\boldsymbol{w}^\star|_{\mathcal{T}^c}\|_2\end{aligned}$$

Proof By definition,

$$\begin{aligned}&\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)-\nabla g(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\\ =&\int_0^1\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\big)dt\big((\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)-(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\big)\end{aligned}$$

Because $(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})$ is the solution of (4.25), we have $\nabla g(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}=0$. Therefore,

$$\begin{aligned}&\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\\ =&\int_0^1\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\big)dt\big((\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)-(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\big)\\ =&\int_0^1\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\big)\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}dt\big((\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)-(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\big)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\\ +&\int_0^1\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\big)\boldsymbol{P}_{\mathcal{T}^c}dt\big((\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)-(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\big)|_{\mathcal{T}^c}\end{aligned} \tag{4.26}$$

Since g has μ_{4s}-GSRH and $|\mathcal{T}\cup supp(t\boldsymbol{w}^\star+(1-t)\boldsymbol{b}_{\boldsymbol{w}})|\le 4s$ for all $t\in[0,1]$, functions $A_{4s}(\cdot)$ and $B_{4s}(\cdot)$, defined by (4.6) and (4.7), exist such that

$$B_{4s}(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}))\le\lambda_{\min}(\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\big)\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}})$$

and

$$A_{4s}(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b_w},\boldsymbol{b_\Theta}))\geq\lambda_{\max}(\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b_w},\boldsymbol{b_\Theta})\big)\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}).$$

Hence, Proposition 1 can be applied and we have

$$\beta_{4s}((\boldsymbol{b_w},\boldsymbol{b_\Theta}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))\leq\lambda_{\min}(\int_0^1\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b_w},\boldsymbol{b_\Theta})\big)\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}dt)$$

and

$$\alpha_{4s}((\boldsymbol{b_w},\boldsymbol{b_\Theta}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))\geq\lambda_{\max}(\int_0^1\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b_w},\boldsymbol{b_\Theta})\big)\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}dt)$$

The results implies that the matrix $\int_0^1\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b_w},\boldsymbol{b_\Theta})\big)\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}dt$, denoted by $\boldsymbol{M}$, is positive-definite. Henceforth it is invertible and

$$\frac{1}{\alpha_{4s}((\boldsymbol{b_w},\boldsymbol{b_\Theta}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\leq\lambda_{\min}(M^{-1})\leq\lambda_{\max}(M^{-1})\leq\frac{1}{\beta_{4s}((\boldsymbol{b_w},\boldsymbol{b_\Theta}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}. \tag{4.27}$$

Multiplying both sides of (4.26) by $\boldsymbol{M}^{-1}$, we obtain

$$\begin{aligned}&\boldsymbol{M}^{-1}\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\\=&\big((\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)-(\boldsymbol{b_w},\boldsymbol{b_\Theta})\big)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\\&+\boldsymbol{M}^{-1}\big(\int_0^1\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b_w},\boldsymbol{b_\Theta})\big)\boldsymbol{P}_{\mathcal{T}^c}dt\big)\boldsymbol{w}^\star|_{\mathcal{T}^c},\end{aligned}$$

where we use the fact that $\big((\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)-(\boldsymbol{b_w},\boldsymbol{b_\Theta})\big)|_{\mathcal{T}^c}=(\boldsymbol{w}^\star-\boldsymbol{b_w})|_{\mathcal{T}^c}=\boldsymbol{w}^\star|_{\mathcal{T}^c}$, because $\boldsymbol{b_w}|_{\mathcal{T}^c}=0$. By (4.27) and Proposition 2, we have

$$\begin{aligned}&\|(\boldsymbol{w}^\star|_{\mathcal{T}},\boldsymbol{\Theta}^\star)-(\boldsymbol{b_w},\boldsymbol{b_\Theta})\|_2\\=&\big\|\big((\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)-(\boldsymbol{b_w},\boldsymbol{b_\Theta})\big)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\big\|_2\\\leq&\big\|\boldsymbol{M}^{-1}\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\big\|_2\\&+\left\|\boldsymbol{M}^{-1}\big(\int_0^1\boldsymbol{P}_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}^\top\boldsymbol{H}_g\big(t(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)+(1-t)(\boldsymbol{b_w},\boldsymbol{b_\Theta})\big)\boldsymbol{P}_{\mathcal{T}^c}dt\big)\boldsymbol{w}^\star|_{\mathcal{T}^c}\right\|_2\\\leq&\frac{\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\|_2}{\beta_{4s}((\boldsymbol{b_w},\boldsymbol{b_\Theta}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}+\frac{\gamma_{4s}((\boldsymbol{b_w},\boldsymbol{b_\Theta}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}{2\beta_{4s}((\boldsymbol{b_w},\boldsymbol{b_\Theta}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\|\boldsymbol{w}^\star|_{\mathcal{T}^c}\|_2\end{aligned} \tag{4.28}$$

■

Lemma 4 *The estimation error in the current iteration,* $\left\|(\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}})-(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)\right\|_2$*, and that in the next iteration,* $\|(\boldsymbol{b}_{\boldsymbol{w}_s},\boldsymbol{b}_{\boldsymbol{\Theta}})-(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)\|_2$*, are related by the inequality:*

$$\begin{aligned}
&\|(\boldsymbol{b}_{\boldsymbol{w}_s},\boldsymbol{b}_{\boldsymbol{\Theta}})-(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)\|_2\\
\leq&\frac{\gamma_{4s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))+\gamma_{2s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}{2\beta_{2s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\big(1+\frac{\gamma_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\big)\\
\times&\left\|(\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}})-(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)\right\|_2+\frac{2\left\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\\
+&\big(1+\frac{\gamma_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\big)\frac{\left\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}}\backslash\mathcal{Z}}\right\|_2+\left\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{Z}\backslash\mathcal{R}_{\boldsymbol{w}}}\right\|_2}{\beta_{2s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}
\end{aligned}\tag{4.29}$$

Proof Since $\mathcal{Z}\subset\mathcal{T}$, we have $\mathcal{T}^c\subset\mathcal{Z}^c$. Thus,

$$\|\boldsymbol{w}^\star|_{\mathcal{T}^c}\|_2=\|(\hat{\boldsymbol{w}}-\boldsymbol{w}^\star)|_{\mathcal{T}^c}\|_2\leq\|(\hat{\boldsymbol{w}}-\boldsymbol{w}^\star)|_{\mathcal{Z}^c}\|_2$$

Using Lemma1, we get

$$\begin{aligned}
\|\boldsymbol{w}^\star|_{\mathcal{T}^c}\|_2\leq&\frac{\gamma_{4s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))+\gamma_{2s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}{2\beta_{2s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\left\|(\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}})-(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)\right\|_2\\
&+\frac{\left\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}}\backslash\mathcal{Z}}\right\|_2+\left\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{Z}\backslash\mathcal{R}_{\boldsymbol{w}}}\right\|_2}{\beta_{2s}((\hat{\boldsymbol{w}},\hat{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}.
\end{aligned}\tag{4.30}$$

Furthermore,

$$\begin{aligned}
&\|(\boldsymbol{b}_{\boldsymbol{w}_s},\boldsymbol{b}_{\boldsymbol{\Theta}})-(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)\|_2\\
\leq&\|(\boldsymbol{b}_{\boldsymbol{w}_s},\boldsymbol{b}_{\boldsymbol{\Theta}})-(\boldsymbol{w}^\star|_{\mathcal{T}},\boldsymbol{\Theta}^\star)\|+\|\boldsymbol{w}^\star|_{\mathcal{T}^c}\|_2\\
\leq&\|(\boldsymbol{w}^\star|_{\mathcal{T}},\boldsymbol{\Theta}^\star)-(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\|_2+\|(\boldsymbol{b}_{\boldsymbol{w}_s},\boldsymbol{b}_{\boldsymbol{\Theta}})-(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\|_2+\|\boldsymbol{w}^\star|_{\mathcal{T}^c}\|_2\\
\leq&2\|(\boldsymbol{w}^\star|_{\mathcal{T}},\boldsymbol{\Theta}^\star)-(\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}})\|_2+\|\boldsymbol{w}^\star|_{\mathcal{T}^c}\|_2,
\end{aligned}$$

where the last equation holds because $\|\boldsymbol{w}^\star|_{\mathcal{T}}\|_0\leq s$ and $\boldsymbol{b}_{\boldsymbol{w}_s}$ is the best s-term approximation to $\boldsymbol{b}_{\boldsymbol{w}}$. Therefore, following Lemma 2,

$$\begin{aligned}
&\|(\boldsymbol{b}_{\boldsymbol{w}_s},\boldsymbol{b}_{\boldsymbol{\Theta}})-(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)\|_2\\
\leq&\frac{2\left\|\nabla g(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}+\big(1+\frac{\gamma_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}},\boldsymbol{b}_{\boldsymbol{\Theta}}),(\boldsymbol{w}^\star,\boldsymbol{\Theta}^\star))}\big)\|\boldsymbol{w}^\star|_{\mathcal{T}^c}\|_2
\end{aligned}\tag{4.31}$$

Combining (4.30) and (4.31), we obtain

$$
\begin{aligned}
&\|(\boldsymbol{b}_{\boldsymbol{w}_s}, \boldsymbol{b}_{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\|_2 \\
\leq &\frac{\gamma_{4s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)) + \gamma_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}{2\beta_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}\big(1 + \frac{\gamma_{4s}((\boldsymbol{b}_{\boldsymbol{w}}, \boldsymbol{b}_{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}}, \boldsymbol{b}_{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}\big) \\
\times &\left\|(\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 + \frac{2\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}}, \boldsymbol{b}_{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))} \\
+ &\big(1 + \frac{\gamma_{4s}((\boldsymbol{b}_{\boldsymbol{w}}, \boldsymbol{b}_{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}}, \boldsymbol{b}_{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}\big)\frac{\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}}\backslash\mathcal{Z}}\right\|_2 + \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{Z}\backslash\mathcal{R}_{\boldsymbol{w}}}\right\|_2}{\beta_{2s}((\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}
\end{aligned}
$$

■

B.1 Proof of Theorem 4.4.1

Proof Following Definition 4.4.1, it is easy to verify that for $k \leq k^{'}$ and any vector $\boldsymbol{u}$, we have $A_k(\boldsymbol{u}) \leq A^{'}_k(\boldsymbol{u})$ and $B_k(\boldsymbol{u}) \geq B^{'}_k(\boldsymbol{u})$. Henceforth, for $k \leq k^{'}$ and any pair of vectors $\boldsymbol{p}$ and $\boldsymbol{q}$, we have $\alpha_k(\boldsymbol{p}, \boldsymbol{q}) \leq \alpha^{'}_k(\boldsymbol{p}, \boldsymbol{q})$, $\beta_k(\boldsymbol{p}, \boldsymbol{q}) \geq \beta^{'}_k(\boldsymbol{p}, \boldsymbol{q})$ and $\mu_k \leq \mu_{k'}$. Consequently, for any function that satisfies μ_k-GSRH,

$$
\frac{\alpha_k(\boldsymbol{p}, \boldsymbol{q})}{\beta_k(\boldsymbol{p}, \boldsymbol{q})} = \frac{\int_0^1 A_k(t\boldsymbol{q} + (1-t)\boldsymbol{p})dt}{\int_0^1 B_k(t\boldsymbol{q} + (1-t)\boldsymbol{p})dt} \leq \frac{\int_0^1 \mu_k B_k(t\boldsymbol{q} + (1-t)\boldsymbol{p})dt}{\int_0^1 B_k(t\boldsymbol{q} + (1-t)\boldsymbol{p})dt} = \mu_k
$$

holds and thereby $\frac{\gamma_k(\boldsymbol{p},\boldsymbol{q})}{\beta_k(\boldsymbol{p},\boldsymbol{q})} \leq \mu_k - 1$. Thus, applying Lemma 3 to the estimation in the i-th iteration, we obtain

$$
\begin{aligned}
&\left\|(\hat{\boldsymbol{w}}^{(i)}, \hat{\boldsymbol{\Theta}}^{(i)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 \\
&\leq (\mu_{4s} - 1)\mu_{4s}\left\|(\hat{\boldsymbol{w}}^{(i-1)}, \hat{\boldsymbol{\Theta}}^{(i-1)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 + \frac{2\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{T}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}}, \boldsymbol{b}_{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))} \\
&+ \mu_{4s}\frac{\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{R}_{\boldsymbol{w}}\backslash\mathcal{Z}}\right\|_2 + \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{Z}\backslash\mathcal{R}_{\boldsymbol{w}}}\right\|_2}{\beta_{2s}((\hat{\boldsymbol{w}}^{(i-1)}, \hat{\boldsymbol{\Theta}}^{(i-1)}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))} \\
&\leq (\mu_{4s} - 1)\mu_{4s}\left\|(\hat{\boldsymbol{w}}^{(i-1)}, \hat{\boldsymbol{\Theta}}^{(i-1)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 + \frac{2\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2}{\beta_{4s}((\boldsymbol{b}_{\boldsymbol{w}}, \boldsymbol{b}_{\boldsymbol{\Theta}}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))} \\
&+ 2\mu_{4s}\frac{\left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2}{\beta_{2s}((\hat{\boldsymbol{w}}^{(i-1)}, \hat{\boldsymbol{\Theta}}^{(i-1)}), (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star))}
\end{aligned}
$$

Using the assumption $\mu_{4s} \leq \frac{1+\sqrt{3}}{2}$, and $\epsilon \leq B_{4s}(\boldsymbol{w}, \boldsymbol{\Theta})$ for some $\epsilon > 0$ for all $4s$-sparse $\boldsymbol{w}$, we have

$$\begin{aligned} \left\|(\hat{\boldsymbol{w}}^{(i)}, \hat{\boldsymbol{\Theta}}^{(i)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 \leq \frac{1}{2} &\left\|(\hat{\boldsymbol{w}}^{(i-1)}, \hat{\boldsymbol{\Theta}}^{(i-1)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 \\ &+ \frac{3+\sqrt{3}}{\epsilon} \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2 . \end{aligned} \tag{4.32}$$

Furthermore, since $\left\|\boldsymbol{w}^\star|_{\mathcal{S}\setminus\hat{\mathcal{S}}^{(i)}}\right\|_2 = \left\|(\hat{\boldsymbol{w}}^{(i)} - \boldsymbol{w}^\star)|_{\mathcal{S}\setminus\hat{\mathcal{S}}^{(i)}}\right\|_2 \leq \left\|(\hat{\boldsymbol{w}}^{(i)}, \hat{\boldsymbol{\Theta}}^{(i)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2$, we have

$$\left\|\boldsymbol{w}^\star|_{\mathcal{S}\setminus\hat{\mathcal{S}}^{(i)}}\right\|_2 \leq \frac{1}{2} \left\|(\hat{\boldsymbol{w}}^{(i-1)}, \hat{\boldsymbol{\Theta}}^{(i-1)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 + \frac{3+\sqrt{3}}{\epsilon} \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2 . \tag{4.33}$$

The theorem follows using the (4.32) and (4.33) recursively. ■

B.2 Proof of Corolary 4.4.1

Proof By Part b of Theorem 4.4.1,

$$\begin{aligned} \left\|\boldsymbol{w}^\star|_{\mathcal{S}\setminus\hat{\mathcal{S}}^{(i)}}\right\| &\leq 2^{-i} \left\|(\hat{\boldsymbol{w}}^{(0)}, \hat{\boldsymbol{\Theta}}^{(0)}) - (\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)\right\|_2 + \frac{6+2\sqrt{3}}{\epsilon} \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2 \\ &\leq 2^{-i}(\left\|\hat{\boldsymbol{w}}^{(0)} - \boldsymbol{w}^\star\right\|_2 + \left\|\hat{\boldsymbol{\Theta}}^{(0)} - \boldsymbol{\Theta}^\star\right\|_2) + \frac{6+2\sqrt{3}}{\epsilon} \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2 \\ &\leq 2^{-i}(2\sqrt{s}\bar{M} + c\sqrt{P}) + \xi m \\ &\leq m \quad \text{if} \quad i \geq \log_2 \frac{2\sqrt{s}\bar{M} + c\sqrt{P}}{(1-\xi)m}. \end{aligned} \tag{4.34}$$

The second inequality follows from triangle inequality. With $\max(\|\boldsymbol{\Theta}^\star\|_\infty, \|\hat{\boldsymbol{\Theta}}^{(0)}\|_\infty) \leq c/2$, $\left\|\hat{\boldsymbol{\Theta}}^{(0)} - \boldsymbol{\Theta}^\star\right\|_2 \leq c\sqrt{P}$ holds, where c is a constant. Combining this fact and the assumption $m \geq \frac{6+2\sqrt{3}}{\xi\epsilon} \left\|\nabla g(\boldsymbol{w}^\star, \boldsymbol{\Theta}^\star)|_{\mathcal{I}\cup\mathcal{R}_{\boldsymbol{\Theta}}}\right\|_2$ with $0 < \xi < 1$, the third inequality can be obtained. The last inequality follows after some algebra. This implies $\hat{\mathcal{S}}^{(i)} = \mathcal{S}$ if $i \geq \log_2 \frac{2\sqrt{s}\bar{M}+c\sqrt{P}}{(1-\xi)m}$. ■

5. CLASSIFICATION OF SHORT SINGLE LEAD ELECTROCARDIOGRAMS (ECGS) FOR ATRIAL FIBRILLATION DETECTION USING PIECEWISE LIEAR SPLINE AND XGBOOST

5.1 Introduction

Atrial Fibrillation(AF) is an abnormal heart rhythm characterized by rapid and irregular heartbeat. AF, usually associated with significant mortality and morbidity, is the most common sustained cardiac arrhythmia, occurring in 1-2% of the general population [2,3]. At least 2.7 million Americans are living with AF and, more than 12 million Europeans and North Americans are estimated to suffer from AF [3,4]. The incidence of AF increases with age, from less than 0.5% at 40-50 years, to 5-15% for 80 years of age [5]. Its prevalence will likely triple in the next 30-50 years, particularly, in the United States and other western countries with aging population demographics [6]. This growth may also be influenced by extended survival outcomes for patients with congestive heart failure (CHF), valvular heart disease, and coronary artery disease, as AF is common among patients with other forms of structural heart disease.

Accurate diagnosis of AF is the first step to address this problem and is essential in mitigating such serious concerns. There have been many previous research studies related to the classification of abnormal ECG beats for arrhythmia [88–93]. Most of the proposed models have used classical classifiers such as support vector machine (SVM) or neural network with discrete wavelet transformation of the signals to capture morphological characteristics of the ECG [91,93]. One of the key challenges for AF detection is that it can be episodic. Hence, AF classification from continuous monitoring data in real settings is important for its accurate diagnosis. With the advent of wearable monitoring devices and increased computational power, researchers have endeavored in developing comparable classification methods in more realistic settings. For example, [94] tackled the classification of ECG beats collected from mobile devices to identify AF and myocardial infarction. Due to the limited computational capacity of those mobile devices, the classification model was developed based on heart rate variability, instead of analyzing morphological features of ECG beats, therefore under-utilizing rich temporal data. In another study, the classification problem in wearable device environment was studied, however only the compressed sensed ECG signals (encoded by discrete wavelet transformation and basis pursuit denoising) were considered [95]. With the similar constraints on computational power, the SVM method was employed on heart rate variability. To make the algorithm computationally efficient, [96] proposed a method focusing on heart rate variability without dealing with high dimensional morphological features.

We tackled AF classification problem based on ECG recordings (Physionet Challenge 2017 data [97]) collected from AliveCor device, which is an ECG recording device in the mobile environment [98]. The proposed model can capture heart rate variability and morphological features without generating high dimensional features as wavelet analysis does. To circumvent representing the morphology in high dimension, we employed a signal fitting method called *piecewise linear function*. XGBoost [99], a gradient boosting method based classifier known for its performance in many data analysis competition, is used to have improved classification performance. We achieved an F_1 score of 81% on the test set from PhysioNet Challenge, which is comparable to other high-ranked competitive classifiers [98].

Overall, our approach focuses on identification of features from waveform morphology with piecewise linear splines and,

- Generates fewer number of features than the methods based on the discrete wavelet transformation, making this method more efficient and statistically robust.
- Can be applied to any type of ECG recordings. In comparison, most existing methods are trained on standard ECG database collected in hospital and not contaminated

by any external noises; they also focus only on classification between normal and AF rhythms.

- Uses the Kaggle [100][1] winning algorithm XGBoost for the classifier; this approach is highly efficient and flexible and can be easily used on distributed platforms for further computational efficiency.

5.2 Methods

5.2.1 Challenge data

ECG recordings, collected and band pass filtered using the AliveCor device, were sampled at 300 Hz. The training set contains 8,528 single lead ECG recordings ranging from 9 seconds to just over 60 seconds. The test set (withheld by the organizers) contains 3,658 ECG recordings of similar lengths. Each recording is labeled as either 'Normal', 'AF', 'Other' or 'Noisy'. All the labelling was performed by a single expert. During the various phases of the challenge, reference labels for the training data of the ECGs were updated with three reference versions provided by the organizers. The data profile in Table 5.1 is given based on the latest version. The test set was unavailable to the public and was not accessible to us.

5.2.2 AF classification algorithm overview

AF is defined as a "tachyarrhythmia characterized by predominantly uncoordinated atrial activation with consequent deterioration of atrial mechanical function" by the American College of Cardiology (ACC), the American Heart Association (AHA) and the European Society of Cardiology (ESC) [101].

Despite the enormity of this issue, AF detection remains problematic, as it can be episodic. The irregular rhythms in the ECG, which can be captured by the underlying pattern of R waves, is a key factor when diagnosing AF. Another important factor

[1]A machine learning challenge

Table 5.1. Data profile for the training set

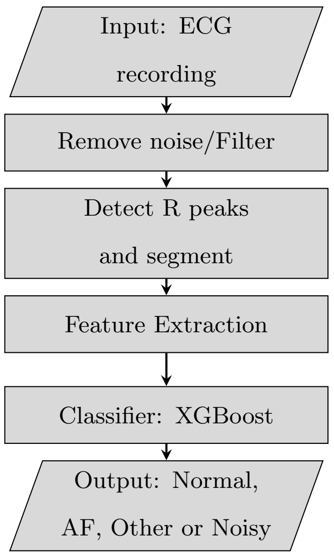
Type	# recording	Time length (s)				
		Mean	SD	Max	Median	Min
Normal	5076	32.11	9.97	60.95	30.0	9.05
AF	758	32.34	12.32	60.21	30.0	9.99
Other	2415	34.30	11.76	60.86	30.0	9.13
Noisy	279	24.38	10.41	60.0	30.0	9.36
Overall	**8528**	32.50	10.89	60.95	30.0	9.05

is the absence of P wave. AF detection by an algorithm can be considered as one of two school of thoughts: atrial activity analysis-based or ventricular response analysis-based methods. Atrial activity analysis-based AF detectors are based on the analysis of the absence of P waves or the presence of fibrillatory f waves in the TQ interval. In contrast, ventricular response analysis is based on the predictability of the inter-beat timing ('RR interval') of the QRS complexes in the ECG [97] (Figure 5.2). We have developed a hybrid method, where both approaches are combined for the selection of features. To extract information according to atrial activity and ventricular response, we break the method into several steps, which are shown in the flowchart (Figure 5.1).

We implemented the algorithm in Python 2.7, and incorporated some existing packages, 'biosppy' [102] and 'scipy' [103] for pre-processing, such as denoising and re-sampling for the wavelet method. The model was trained on the training dataset and stored as a separate file. To predict a new record, the model takes an ECG recording as an input and returns a class label of either 'Normal', 'AF', 'Other' or 'Noisy'. The evaluation was performed by running the algorithm on the server, equipped with virtual machines (VM), provided by the challenge organizers. Each VM is configured with a single-core AMD 64 processors, 2GB of RAM, a 2GB read-wirte/home partition, and a 500 MB read-write/tmp partition. Each classification task was limited to 2×10^{11} CPU instructions [104]. Our final submission consisted of 34 recordings that exceeded the computational limit and were classified as 'Noisy'.

5.2.3 R peaks detection and PQRST segmentation

A complete normal heartbeat produces four entities on ECG – a P wave, a QRS complex, a T wave and a U wave, where the U wave is not typically seen and its absence is generally ignored. Therefore, a PQRST interval as shown in Figure 5.2 is considered to represent a complete heartbeat wave on an ECG recording. Depending on the source lead of the ECG, this interval might be inverted (i.e., negative R peaks) for some waves.



Entire ECG leading

- RR interval length
- RR interval difference
- Heartbeat rate

Each segmentation of ECG leading

- Fourier Coefficient of each segmentation
- Piecewise linear spline
  - Location of PQRST wave
  - Amplitude of PQRST wave
  - Appearance of f waves

Fig. 5.1. Flowchart for different steps (left) and list of features (right).

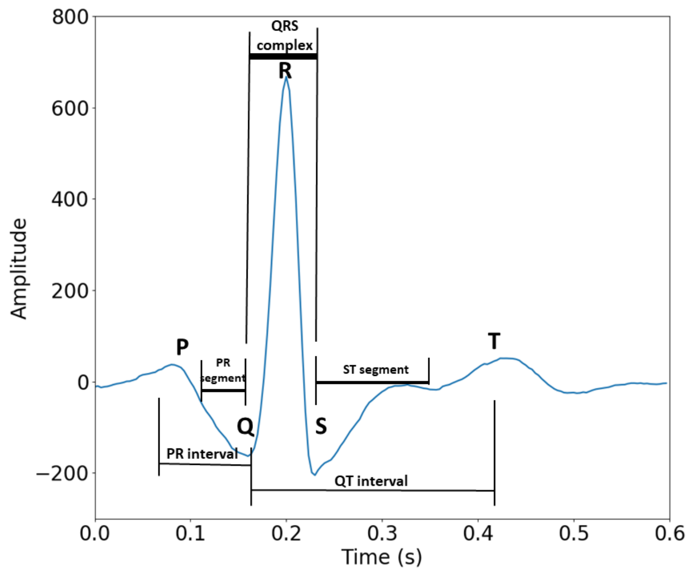


Fig. 5.2. A depiction of a typical ECG beat showing the PQRST interval

R peaks detection and PQRST segmentation is the first step for analyzing the ECG and classifying to different categories. Correctly identifying the R peaks and making proper PQRST segmentation could provide valuable information regarding different features including heart rate, and RR interval, which can also facilitate the discovery of the irregularity of the heart rhythm.

Comparison of different methods for R peaks detection

Different methods have been proposed to identify the R peaks. We compared five of the existing methods including Christov [105], Engelse and Zeelenberg [106], SSF (Slope Sum Function) [107], P. Hamilton [108], and H. Gamboa [109]. These five methods have shown an accuracy of 90% or more for QRS detection on Physionet database [110,111]. For the challenge dataset, the performance of these five methods does not differ significantly from each other in the case of positive R peaks [110]. However, Hamilton's [108] method performed better for automatically detecting the negative R peaks.

5.2.4 Fitting heartbeat

Both atrial activity analysis-based or ventricular response analysis-based methods require capturing the morphological features of the ECG wave. Traditional methods use the wavelet transformation, and then consider the coefficients as a representation of the ECG for further analysis. However, it is difficult to extract the P, Q, R, S and T locations and amplitude information from the wavelet coefficients, since the PQRST signals vary their positions randomly [112]. We introduce piecewise linear splines for capturing the waveform morphology at the heartbeat level.

Piecewise linear function

Piecewise linear functions, such as adaptive piecewise estimation are commonly used in non-parametric studies to fit a function [113]. The method uses a series of end-to-end straight lines to approximate the wave or function. The location of end points on the X axis are called knots. The goal is to minimize the sum of the least squared errors between the piecewise linear function and the true function to achieve a better fit. Figure 5.3 shows a simple example of a piecewise linear spline (red) for estimating a quadratic function whose parameters are unknown (blue). The mathematical form of the piecewise linear spline is shown in Equation 5.1. We define $f(x)$ to be the function on $[0, 1]$ without loss of generality:

$$f(x, \mathcal{B}) = \beta_0 + \beta_1 x + \sum_{i=1}^{k} \beta_{i+1}(x - t_i)_+ \tag{5.1}$$

where $\mathbf{t} = \{0 = t_0 < t_1 < \cdots < t_{k+1} = 1\}$ are the knots we choose, $\mathcal{B}$s are the coefficients for each spline.

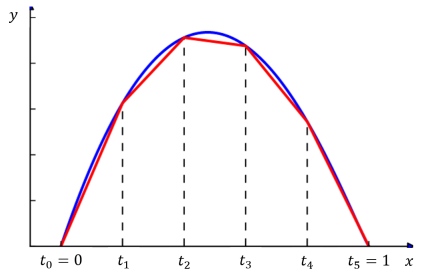


Fig. 5.3. Piecewise linear spline example: here we have a function $f(x)$(blue) is estimated by a piecewise linear spline(red) when the knots $\{x_0 = 0, \ldots, x_5 = 1\}$ is given [114]

The keys to a good approximation on a piecewise linear model are (i) the number of knots, and (ii) the location of the knots. With the number of knots and the location of the knots fixed, our question can be simplified to an optimization problem with a quadratic loss function, as in Equation 5.2.

$$L(y, f(x, \mathcal{B})) = \sum_{i=1}^{N} (y - f(x, \mathcal{B}))^2 \tag{5.2}$$

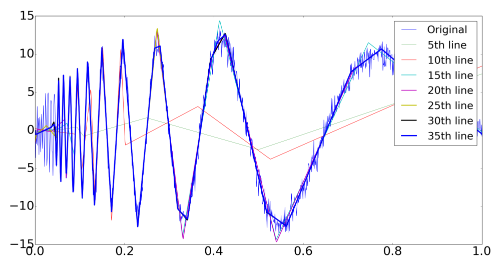
where $f(x, \mathcal{B})$ is defined in Equation 5.1. Our task is simplified to minimize Equation 5.2 on the space of $\mathcal{B}$, in Equation 5.3:

$$\hat{\mathcal{B}} = argmin_{\mathcal{B}}\ L(y, f(x, \mathcal{B})) \tag{5.3}$$

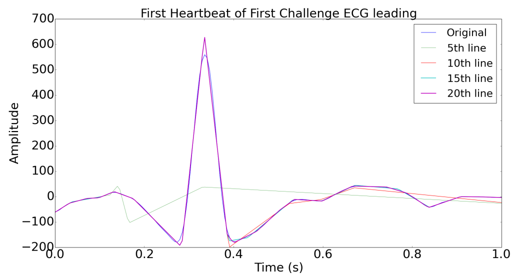
Here, we propose a forward step-wise algorithm and adaptively add new knot to current knots in the most likely position iteratively [115]. The details of each step are given in Algorithm 3. Two examples of the fitting of piecewise linear spline are shown in Figure 5.4. The examples include fitting the function for a Doppler function and PQRST segment of a heartbeat.

Algorithm 3 Fitting piecewise linear functions [115]

```
Given significant level $\alpha$, $f(x_i)$ $i = 1, ..., n$;
for $k = 1, ...$ : do
    Find $\{t_1, ..., t_k\} \in [t_0 = 0, t_{k+1} = 1]$ and corresponding $\hat{f}$ so that it minimize Equation 5.2
    Set residuals $\mathcal{R}_l = \{f(x_j) - \hat{f}(x_j) | x_j \in [t_{l-1}, t_l]\}$, for $l = 1, ..., k$
    Test the independence on $\mathcal{R}_l$ at significant level $\alpha$, $l = 1, ..., k$, collect p-value $p_l$ for each set $\mathcal{R}_l$
    if $\min\{p_i\} < \alpha$ then
        continue
    else
        Residuals are independent in all sets at significant level of $\alpha$
        break
    end if
end for
```



(a) Fitting the Doppler function with piecewise linear function



(b) Fitting a heartbeat with piecewise linear function

Fig. 5.4. The examples show the approximation of functions and waves with piecewise linear functions. For each heartbeat, it takes at most 30 iterations to find a well-fitted piecewise linear function.

The piecewise linear function approximated from the algorithm helps us correctly identify the position of the peaks in the ECG, which is critical for the analysis of the atrial activity in detecting the absence of P waves and the presence of f waves by capturing the morphology.

5.2.5 Feature extraction

To detect AF, clinicians seek for the rhythm of the heartbeat, absence of P wave and, presence of f wave (Figure 5.5). F wave is an atrial flutter wave on ECG which is more of a regular tachyarrhythmia and often superimposed with atrial fibrillation. It appears as a ‘sawtooth’ pattern in the leads II, III and aVF [116] (Figure 5.5).

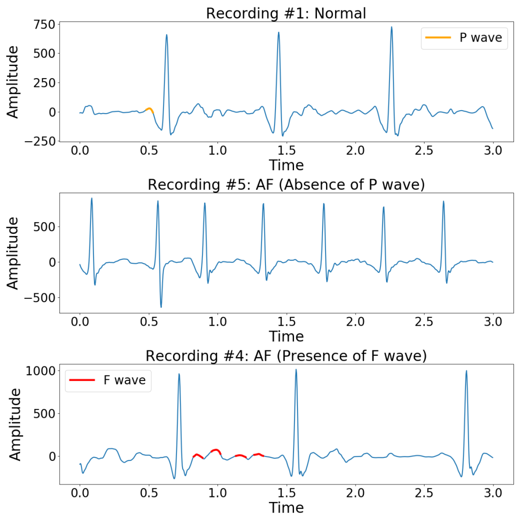


Fig. 5.5. Examples of ECGs of atrial fibrillation. On the top: the arrow indicates a P wave in normal sinus rhythm, which is lost in atrial fibrillation (middle); On the bottom: the morphology of f waves in ECG. The irregularity of heartbeat can be seen in both graphs.

We have developed a set of rules to extract the statistical features of heartbeat rate for the ECG signal and detect the P wave and f wave related features from the coefficients of the heartbeat approximation based on piecewise linear spline. These rules are described in detail in this section.

Heartbeat Rate and Rhythm

The detection of R peaks and PQRST segmentation help us count the number of heartbeat over a period of time. The heartbeat rate of each ECG is tracked along with number of R peaks in a fixed time window. With the window shifting over entire ECG, we get a sequence of heartbeat, which provides the heart rate and the heart rate variability(HRV). We define the set $\{R_i \mid i = 1, ..., n\}^2$ to be the sequence of R peak locations of the ECG leading record, and heartbeat rate is defined as the number of R peaks detected over one second. We compute the length of RR intervals, which are the time differences between two adjacent R peaks, as $\{RR_i = R_{i+1} - R_i \mid i = 1, ..., n-1\}$ and calculate the difference between RR interval lengths as $\{diff_i = RR_{i+1} - RR_i \mid i = 1, ..., n-2\}$. Statistical measurements of $\{RR_i\}$, $\{diff_i\}$ such as mean and standard deviation are extracted, and the variation of the sequence indicative of the rhythm of the heartbeat.

Absence of P waves

The absence of P waves is captured by first annotating each wave in the PQRST intervals. After fitting the PQRST intervals to piecewise linear functions, we mark the inflection points of the piecewise linear functions as our candidates for each peak. Since R peaks are located by the first step, according to the characteristic of each wave, we annotate the PQRST peaks among the candidates according to their amplitude and distance to the R peaks, as shown in Figure 5.6. With the annotation of PQRST, we can detect the absence of P wave by identifying the presence of an inflection point before the Q wave.

[2] n denotes the number of R peaks detected in each ECG recordings, and will vary for different recordings

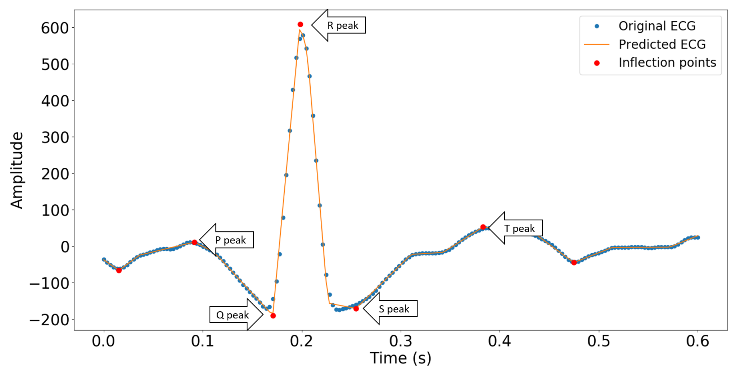


Fig. 5.6. Annotation of PQRST wave from fitted piecewise linear function

Presence of f waves

The presence of f waves in the ECG is another measure used by clinicians for detection of AF. It usually appears as a ‘sawtooth’ pattern [116] (Figure 5.5). To detect AF, we at first identify the coefficients of the piecewise linear function and then find the inflection points. This step is similar to the detection of the absence of P waves. After that, we compute the number of inflection points that follows the R peak. If there is an irregular number of inflection points after the R wave, it is more likely that there is a ‘sawtooth’ pattern for the presence of f waves. We compute the proportion of such waves in the ECG leading, and depending on the percentage, we detect the f waves.

Other Features

We also used other general features such as the RR interval, and the differences in RR interval and heartbeat rate, as described in Figure 5.1. The argument for using differences in RR interval can be explained by the Lorenz plot [117] (Figure 5.7). The slope gets closer to 1 if RR intervals are regular without much variation; in contrast, it becomes more random if RR intervals are irregularly changing as shown in Figure 5.7.

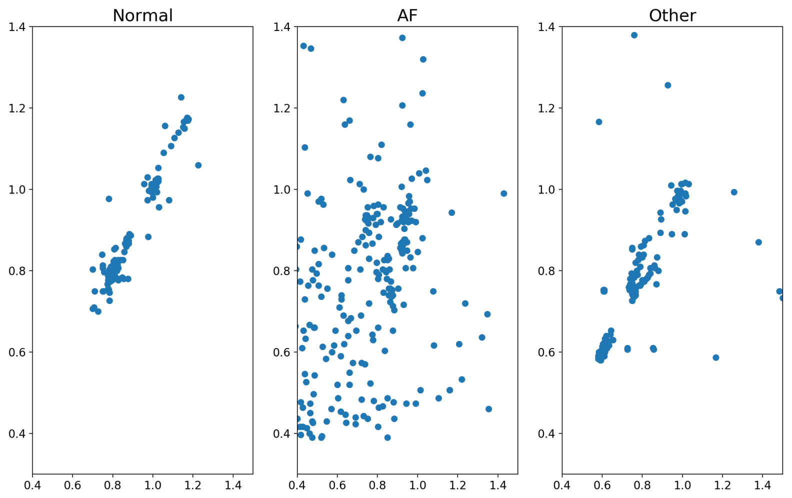


Fig. 5.7. Lorentz plot showing the difference between the differences of RR intervals for Normal, AF and Other classes for three samples. X axis are RR(n) - RR(n-1), and Y axis are RR(n+1) - RR(n) for each n, where n is an index of RR intervals.

5.2.6 Classification

The features extracted using piecewise linear function and other features that has been reported in the literature to classify AF as described in the previous sections were used for inputs for a classifier. The list of the features are: i) heart beat rate and rhythm, ii) presence or absence of P waves as identified by piecewise linear function, iii) presence or absence of f waves, iv) RR interval and v) differences in RR interval. For each samples, these features are extracted and an high dimensional vector was created. These vectors were then split randomly for training and testing with a 10 fold cross validation. We used 82% (7000) of the challenge dataset as our training data and the remaining 18% (1528) as our validation set. XGBoost, a gradient boosting algorithm was used as the classifier of the ECG signal from the features extracted [99]. Other methods such as neural network (NN) were also explored. The XGBoost, short for "Extreme Gradient Boosting", uses gradient boosted trees to solve the problem of supervised learning. Gradient boosted trees use decision trees of a fixed maximum size as base learners, and iteratively learns the base learners to add up to a final

Table 5.2. Counting rules for the numbers of the variables

		Predicted Classification				
		Normal	AF	Other	Noisy	Total
Reference Classification	Normal	Nn	Na	No	Np	$\sum$ N
	AF	An	Aa	Ao	Ap	$\sum$ A
	Other	On	Oa	Oo	Op	$\sum$ O
	Noisy	Pn	Pa	Po	Pp	$\sum$ P
	Total	$\sum$ n	$\sum$ a	$\sum$ o	$\sum$ p	

strong classifier [118]. A python open source package for XGBoost was used for the implementation [119].

5.2.7 Evaluation Metrics

The results are measured in terms of F_1 score and receiver operating characteristics (ROC) curve, described in this section. The definition of the F_1 score is based on the confusion matrix for the reference class and the predicted class. The confusion matrix is shown in Table 5.2.

The performance of the algorithm was evaluated as an F_1 measure as defined by the challenge organizers, which is an average of the three F_1 values from each classification type. F_1 values are defined with the following equations for the three categories.

- Nomral rhythm: $F_{1n} = \frac{2\times Nn}{\sum N+\sum n}$
- AF rhythm: $F_{1a} = \frac{2\times Aa}{\sum A+\sum a}$
- Other rhythm: $F_{1o} = \frac{2\times Oo}{\sum O+\sum o}$
- Noisy: $F_{1p} = \frac{2\times Pp}{\sum P+\sum p}$

The final score is calculated as:

$$F_1 = (F_{1n} + F_{1a} + F_{1o})/3$$

We also represented in other traditional statistical measures such as receiver operating characteristics (ROC) curve, precision-recall curve (PRC), positive predictive value (PPV) as well as sensitivity (a.k.a true positive rate) and specificity. The definitions of these parameters are listed below.

$$\begin{aligned} Specificity &= \frac{Number\ of\ true\ negative}{Number\ of\ actually\ negative\ samples} \\ Sensitivity &= \frac{Number\ of\ true\ positive}{Number\ of\ actually\ positive\ samples} \\ PPV &= \frac{Number\ of\ true\ positive}{Number\ of\ positive\ calls} \\ False\ positive\ rate &= \frac{Number\ of\ false\ positive}{Number\ of\ actually\ negative\ samples} \end{aligned}$$

The ROC curve and PRC are graphical plot that illustrate the diagnostic performance of a binary classifier as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate against the false positive rate at various thresholding settings. PRC is created by plotting the PPV against sensitivity, and shows the trade off between precision and recall for different threshold.

5.3 Results

5.3.1 Classification performance on the test set (hidden)

To build our model on the training data, we performed 10-fold cross validation with randomly selecting 7000 (82%) records as the training set and the remaining 1528 (18%) records as the validation set. For the 10-fold cross validation, we achieved an average F_1 score of 80.5%, accuracy of 83.8% with specificity of 98.3%. Among the F_1 score, the normal class, AF class and other class had a score of 0.90, 0.78 and 0.74 respectively. Especially for the 'AF' class, we have achieved a sensitivity of 0.78 and positive predictive value of 0.81. For the test set that was hidden from us, we

Table 5.3. Final F_1 Score of Piecewise Linear Spline/XGBoost method for the test set (hidden)

Class	F_1 score (%)
Normal	0.90
AF	0.80
Other	0.72
Overall	**0.81**

achieved similar performance statistics. The detailed F_1 score for the hidden test set is shown in Table 5.3.

5.3.2 Precision recall (PR) curve and receiver operating characteristic (ROC) curve

For a multi-class classification problem, we characterize the ROC curve for each class, by testing that class against all other classes. We used 82% (7000) of the challenge dataset as our training data and the remaining 18% (1528) as our validation set to generate the ROC curve, shown in Figure 5.8.

The ROC curves indicate good performance of the algorithm, when we consider the classification problem for each class as a binary classifier. We also studied the head-to-head comparison between each pair of classes and achieved the lowest area under the curve (AUC) for normal-other pair, which is still 0.94. The AUC for each class is above 0.9, and the class 'AF' has achieved an AUC of 0.98. As the data are imbalanced and precision recall curve (PR) better represents the performance in such cases, we used the PR curve. The AUCs of 3 major classes ('Normal', 'AF', 'Other') are all above 0.8 for the PR curve. The AUC of noise is low due to the high degrees of imbalance of noise data in the dataset.

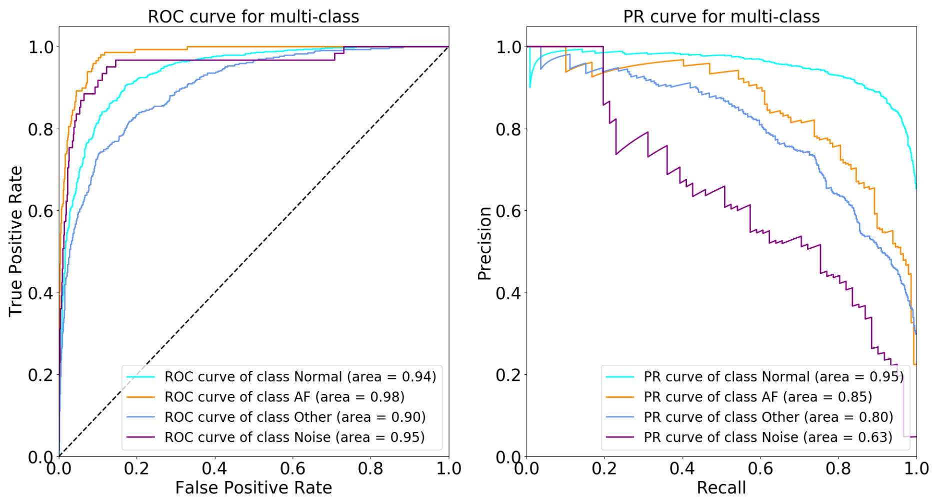


Fig. 5.8. Receiver operating characteristic (ROC) curve (left) and precision recall curve (PR) curve (right) for AF classification. For each curve, one class is tested against all other classes as a binary classification problem.

5.4 Discussion

We have shown that the proposed algorithm using piecewise linear coefficients from the ECG beats is capable of detecting AF from ECGs recorded from wearable devices (AliveCor) with high accuracy (ranked among top 10 challenge results) by identifying important features of waveform morphology for AF. Detailed information regarding the PQRST waves helped improve the result significantly. One of the most significant challenges we faced during the development of our algorithm was the uncertainty about the clinical reasoning for the samples that were labeled as the 'Other' class. Hence, more information regarding the other arrhythmia diseases as well as further study about them could help improve the F_1 score. During the challenge, we tested other existing methods on the challenge data set, including wavelet entropy. We also tested our algorithm on other publicly available ECG datasets with annotation of 'AF' such as the MIT-BIH arrhythmia database [111,120]. These findings are summarized in this section.

5.4.1 Other class

For our result, the class 'Other' has the lowest F_1 score, as shown in Table 5.3. Mostly because the 'Other' class includes any disease that is not AF. Our feature extraction primarily follows the diagnosis and clinical definition of AF. Some of the other arrhythmias include ventricular tachycardia, ventricular fibrillation, supraventricular tachycardias which can be sub classified based on atrial or AV nodal origin and AV nodal re-entry tachycardias [121] (Figure 5.9). Since our model is based on manually selecting features streamlined to detect AF without knowing the individual ECG characteristics of each specific type of arrhythmia in the 'Other' class, we have essentially limited the performance of the algorithm by over-fitting for the class 'AF'. For example, separation of noise and ventricular fibrillation was not investigated. Both of them are irregular rhythms and differentiating them tends to be a hard problems in pattern recognition.

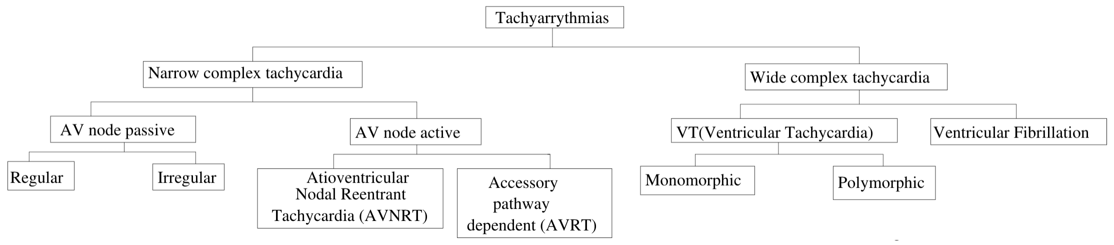

Tachyarrythmias
Narrow complex tachycardia
Wide complex tachycardia
AV node passive
AV node active
VT(Ventricular Tachycardia)
Ventricular Fibrillation
Regular
Irregular
Atioventricular Nodal Reentrant Tachycardia (AVNRT)
Accessory pathway dependent (AVRT)
Monomorphic
Polymorphic

Fig. 5.9. Classification diagram for tachyarrhythmias.[3]

5.4.2 Comparison to other methods

Wavelet transformation has been widely used in AF detection [122–124]. We included the RR-interval information (distance between two R peaks and the difference between the lengths of heartbeats) and the coefficient and entropy of the wavelet transformation. This method achieved an overall F_1 score of 0.72 and accuracy of 0.74. We also replaced wavelet coefficients with Fourier coefficients as the features, which achieved a similar level of accuracy and F_1 score. Table 5.4 illustrates detailed

[3]AV: atrioventricular

results, revealing that wavelet and Fourier coefficient methods have limited power on detecting AF on such datasets with the XGBoost classifier.

5.4.3 Other dataset

The method can easily be applied to other ECG datasets. We applied our model trained by challenge data on a subset of MIT-BIH arrhythmia database. For this dataset, the recordings are annotated as ‘AF’ or not AF (not ‘AF’ may include ‘Normal’, ‘Other’ or ‘Noisy’). The dataset includes 81 ECG samples, and their lengths are approximately 30 seconds. By using the built model from the challenge data, we have achieved an accuracy of 93.83%, sensitivity of 87.5% and specificity of 95.89% with this dataset.

5.4.4 Limitation and future direction

Our algorithm extracts features at the heartbeat level, which can be computationally expensive when applied to long ECG recordings. Although we have used large computational resources, which may not be currently available for wearable devices, the data can be sent to a cloud environment for processing. Additionally, we did not have access to patients’ demographics, clinical history, or lab results. Such clinical information can give a better context for a higher accuracy of the AF classification.

Table 5.4. Comparison table of different methods

Methods	F_1 score (%)				Specificity	Sensitivity	PPV
	Normal	AF	Other	Overall			
Wavelet Transformation	0.87	0.70	0.58	0.72	0.981	0.601	0.753
Fourier Transformation	**0.90**	0.63	0.66	0.73	**0.983**	0.570	0.770
Piecewise linear spline	**0.90**	**0.80**	**0.72**	**0.81**	0.982	**0.779**	**0.812**

6. LOCAL REGION SPARSE LEARNING FOR IMAGE-ON-SCALAR REGRESSION

6.1 Introduction

There has been significant research activity aimed at the association between image data and other scalar variables (e.g. cognitive score, diagnostic status) in the study of neurodegenerative and neuropsychiatric diseases, such as Alzheimer's disease (AD) [125]. The growing public threat of AD has raised the urgency to discover ROI of magnetic resonance images (MRI) that may identify subjects at greatest risk for future cognitive decline and accelerate the testing of preventive strategies. Machine learning methods have been developed and the penalized optimization is popular in the framework of the empirical risk minimization plus a penalty. However, spatially heterogeneous smoothness and local region selection greatly complicates the image analysis. To address these challenges, several regularization methods have been proposed to impose sparsity on both pixel values and their spatial derivatives. For instance, GraphNet [126] combines the Lasso penalty and an ℓ_2 penalty of image gradients, and TV-ℓ_1 [127,128] uses a weighted combination of the Lasso penalty and the TV penalty.

It is well-known that both Lasso and TV models have the inherent bias and often lead to less stable predictions [45]. For example, the spatially adaptive TV model [129] was proposed to remove the inherent bias in the TV model by utilizing a spatially varying weight function that is inversely proportional to the magnitude of image derivatives. It is a two-step procedure where the weight function obtained from the first step using standard TV is then used to guide smoothing in the second step. It is of interest to note that, in the statistical literature, the Smoothly Clipped Absolute Deviation (SCAD) penalty [130] has been proposed in the context of high-dimensional linear regression to address the shortcomings of Lasso (which is not consistent in variable selection). SCAD has some desired properties of the estimator such as continuity, asymptotic unbiasedness, sparsity, and the so-called oracle property (which behaves the same as when the zero coefficients are known in advance). There are a few papers on the use of SCAD for image analysis [128, 131]. None of them consider the local region learning. We will adapt the SCAD penalty for our local region selection problem in the framework of image-on-scalar regression.

In this chapter, we propose a novel regularization method called SCAD2TV, which combines the SCAD regularization, enforcing sparsity, and the SCAD of TV regularization, enforcing spatial contiguity, into one group, which segments contiguous spatial regions against zero-valued background. This chapter makes three main contributions:

- The new penalty, SCAD2TV, forces zeros on coordinates and spatial derivative jointly, which makes it easy to identify ROI for the image-on-scalar regression model. It solves the bias issue inherent in LASSO or TV methods.
- Our proposed algorithms are based on ADMM, which decomposes a non-convex problem with the non-convex penalty into two iterative optimization problems with explicit solutions. The divide and conquer learning algorithm is also developed, thereby allowing scaling to large images.
- Compared with GraphNet and TV-ℓ_1, SCAD2TV has better or competitive performance in either prediction or selection errors.

6.2 Image-on-Scalar Regression and SCAD2TV

6.2.1 Image-on-Scalar Regression

Regression models with image responses and scalar predictors are routinely encountered in many applications [132, 133]. Consider an image-on-scalar regression model with varying coefficients: $Y(s) = X^T\beta(s) + \eta(s) + \epsilon(s)$, where $X \in \mathbb{R}^p$ is the covariate, $Y(s) \in \mathbb{R}$ is the image response at pixel $s \in \mathcal{S}$ (a 2D or 3D domain), and $\beta(s) = (\beta_1(s), \dots, \beta_p(s))^T \in \mathbb{R}^p$ is the coefficient image vector. Here $\eta(\cdot)$ is a zero-mean spatial field which characterizes the spatial correlation, and $\epsilon(\cdot)$ is the white noise with mean zero and variance σ^2. In this chapter, we focus on $Y \in \mathbb{R}^{N \times N}$ a 2D image. Extension to 3D images is straightforward. The objective is to identify ROI in the response image which are associated with the corresponding covariate by estimating the coefficient images $\beta_1, \dots, \beta_p$. The available data are image and covariate pairs for n subjects, $(X_i, Y_i(\cdot))$, $i = 1, \dots, n$. We obtain the estimator by minimizing

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{s \in \mathcal{S}} \Big(Y_i(s) - X_i^T \beta(s) \Big)^2 + \sum_{j=1}^{p} \mathrm{pen}(\beta_\ell), \tag{6.1}$$

where pen$(\cdot)$ is a penalty function which favors estimators according to certain criteria. Our purpose is to recover nonzero active regions of $\beta_1, \dots, \beta_p$. The main challenges are that we need to impose sparsity of pixel values and extract active regions simultaneously.

6.2.2 Existing Regularizers

TV and SCAD. The TV analysis plays a fundamental role in various image analyses since the path-breaking works [134,135]. We focus on the anisotropic version of TV. For $\beta \in \mathbb{R}^{N \times N}$, define the discrete gradient $\nabla : \mathbb{R}^{N \times N} \to \mathbb{R}^{N \times N \times 2}$ is defined by

$$(\nabla \beta)_{jk} = \begin{cases} (\beta_{j+1,k} - \beta_{jk}, \beta_{j,k+1} - \beta_{jk}), & 1 \le j, k \le N-1, \\ (0, \beta_{j,k+1} - \beta_{jk}), & j = N, 1 \le k \le N-1, \\ (\beta_{j+1,k} - \beta_{jk}, 0), & 1 \le j \le N-1, k = N, \\ (0, 0), & k = j = N. \end{cases}$$

The TV norm $\|\beta\|_{TV}$ is just $\|\beta\|_{TV} = \sum_{j,k} \|(\nabla \beta)_{jk}\|_1$. The isotropic induced TV norm is $\sum_{j,k} \|(\nabla \beta)_{jk}\|_2$, which is equivalent to the anisotropic induced TV norms up to a factor of $\sqrt{2}$.

The SCAD penalty $\rho_\lambda(\cdot)$ is more conveniently defined its derivative

$$\rho'_\lambda(t) = \lambda\Big\{I(t \le \lambda) + \frac{(a\lambda - t)_+}{(a-1)\lambda} I(t > \lambda)\Big\}, \quad t > 0,$$

and $\rho_\lambda(0) = 0$. We use $a = 3.7$ by convention. Consider a penalized least squares problem: minimize $\frac{\varrho}{2}(z-\theta)^2 + \rho_\lambda(\theta)$. The solution is unique, explicit, and $\hat{\theta} = S_{\varrho,\lambda}(z)$, where $S_{\varrho,\lambda}$ is the thresholding function. Figure 6.1 displays the thresholding function for SCAD and the soft thresholding function for Lasso with $\varrho = 1$ and $\lambda = 2$. The SCAD penalty shrinks small coefficients to zero while keeping the large coefficients without shrinkage.

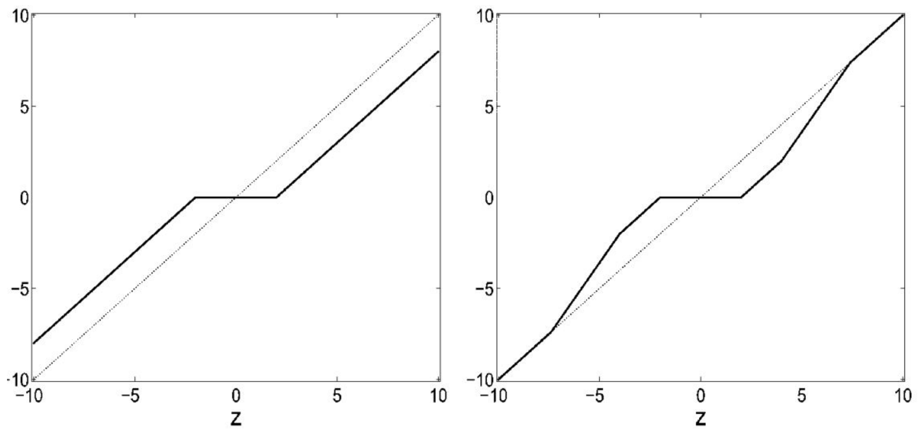


Fig. 6.1. Thresholding function for the Lasso penalty (left) and the SCAD penalty (right) with $\varrho = 1$ and $\lambda = 2$.

GraphNet and TV-ℓ_1. GraphNet and TV-ℓ_1 have been successful applied to medical images. GraphNet is the weighted average of an ℓ_1 penalty on all coordinates and a squared ℓ_2 penalty on the discrete gradient, while TV-ℓ_1 is the weighted average of an ℓ_1 penalty and a TV penalty:

$$\text{pen}_{GN}(\beta) = \lambda\Big(\gamma\|\nabla\beta\|_2^2 + (1-\gamma)\|\beta\|_1\Big)$$
$$\text{pen}_{TV-\ell_1}(\beta) = \lambda\Big(\gamma\|\beta\|_{TV} + (1-\gamma)\|\beta\|_1\Big).$$

For both penalties, $\lambda > 0$ is the smoothing parameter which controls the strength of regularization and $\gamma \in [0, 1]$ is another smoothing parameter controlling the trade-off between pixel sparsity and spatial regularity.

6.2.3 A New Penalty: SCAD2TV

For each coordinate β_{jk}, the discrete gradient $(\nabla\beta)_{jk} \in \mathbb{R}^2$ involves three coordinate values $\beta_{jk}, \beta_{j+1,k}, \beta_{j,k+1}$. Let $\widetilde{\beta}_{jk} = (\beta_{jk}, \beta_{j+1,k}, \beta_{j,k+1})^T$. The SCAD2TV penalty is defined by

$$\text{pen}_{S2TV}(\beta) = \sum_{j,k=1}^{N} \left\{ \gamma \sum_{l=1}^{2} \rho_\lambda\Big(|(\nabla\beta)_{jk,l}|\Big) + (1-\gamma) \sum_{l=1}^{3} \rho_\lambda\Big(|\widetilde{\beta}_{jk,l}|\Big) \right\}, \tag{6.2}$$

where $\lambda > 0$ and $\gamma \in [0, 1]$ are two tuning parameters, and ρ_λ is the SCAD function. The first term in the penalty allows adaptive estimation of the coefficient image and the second one enforces sparsity on coordinate values. One may also consider the functional version of (6.2). After some rescaling, (6.2) is equivalent to

$$\text{pen}_{S2TV}(\beta) = \gamma \int \rho_\lambda(|\dot{\beta}|) + (1-\gamma) \int \rho_\lambda(|\beta|). \tag{6.3}$$

The SCAD2TV solves the bias problem inherent in the TV and Lasso models. Note that this penalty function, unlike the L_1 penalty used in Lasso, is not convex, so that (6.1) is a non-convex objective function. We solve this problem based on the ADMM and convert it into two sub-problems with closed-form solutions. In general, ADMM has successful applications to convex problems. The behavior of ADMM applied to nonconvex problems has been a mystery. Recently, the global convergence of ADMM in non-convex optimization is discussed in [136], which shows that several ADMM algorithms including SCAD are guaranteed to converge.

6.3 Local Region Learning by SCAD2TV

6.3.1 Algorithm based on ADMM

Our proposed algorithm is based on ADMM [137]. We may write (6.1) as the matrix form by an abuse of notation:

$$\frac{1}{n}\sum_{i=1}^{n}\left\|Y_i - X_i^T\beta\right\|_2^2 + \sum_{j=1}^{p}\text{pen}_{S2TV}(\beta_j), \tag{6.4}$$

where $Y_i \in \mathbb{R}^{N^2}$ is the veritorized response image for subject i, $X_i \in \mathbb{R}^{N^2 \times pN^2}$ is the fixed extended design matrix related to the covariate for subject i, and $\beta \in \mathbb{R}^{pN^2}$ is the concatenated vectorized unknown coefficient image. Furthermore, one of the advantages of SCAD2TV in (6.2) is that we can write $\sum_{j=1}^{p}\text{pen}_{S2TV}(\beta_j)$ as $\|\rho_\lambda(D\beta)\|_1$ for a fixed $p(5(N-1)^2+6(N-1))$ by pN^2 matrix D depending only on γ, which greatly facilitates the efficiency of our algorithm. This fact can be easily seen since the elements involved in the $(j,k)^{th}$ term in (6.2) are

$$((\nabla\beta)_{jk}, \widetilde{\beta}_{jk})^T = (\beta_{j+1,k}-\beta_{j,k},\ \beta_{j,k+1}-\beta_{j,k},\ \beta_{j,k},\ \beta_{j+1,k},\ \beta_{j,k+1})^T = D_{jk}\widetilde{\beta}_{jk},$$

for a fixed matrix D_{jk}. So D is the concatenated version of D_{jk}.

Problem (6.4) is equivalent to

$$\begin{aligned} \min \quad & \frac{1}{n}\sum_{i=1}^{n}\left\|Y_i - X_i^T\beta\right\|_2^2 + \|\rho_\lambda(\alpha)\|_1 \\ s.t. \quad & \alpha = D\beta. \end{aligned}$$

We form the augmented Lagrangian as

$$L_\varrho(\beta,\alpha,\eta) = \frac{1}{n}\sum_{i=1}^{n}\left\|Y_i - X_i^T\beta\right\|_F^2 + \|\rho_\lambda(\alpha)\|_1 + \eta^T(\alpha - D\beta) + \frac{\varrho}{2}\left\|\alpha - D\beta\right\|_2^2.$$

The ADMM consists of the iterations

$$\beta^{(t+1)} = \arg\min_{\beta} L_{\varrho}(\beta, \alpha^{(t)}, \eta^{(t)}) \tag{6.5}$$

$$\alpha^{(t+1)} = \arg\min_{\alpha} L_{\varrho}(\beta^{(t+1)}, \alpha, \eta^{(t)}) \tag{6.6}$$

$$\eta^{(t+1)} = \eta^{(t)} + \varrho(\alpha^{(t+1)} - D\beta^{(t+1)}). \tag{6.7}$$

It should be emphasized that both (6.5) and (6.6) have the explicit solutions. Specifically, (6.5) is a ridge regression problem and (6.6) is a penalized least squares problem with the identity design matrix. The closed-form solutions for (6.5) and 6.6 are, respectively,

$$\beta^{(t+1)} = \frac{1}{2}\Big(\frac{1}{n}\mathbf{X}_i^T\mathbf{X}_i + \frac{\varrho}{2}D^TD\Big)^{-1}\Big(\frac{2}{n}\sum_{i=1}^{n}\mathbf{X}_i^T\mathbf{Y}_i + D^T\eta^{(t)} + \varrho D^T\alpha^{(t)}\Big)$$

$$\alpha^{(t+1)} = S_{\varrho,\lambda}\Big(D\beta^{(t+1)} - \frac{\eta^{(t)}}{\varrho}\Big).$$

The details of the algorithm is summarized in Algorithm 4.

Algorithm 4 Local Region Learning by SCAD2TV

```
**Input** Training samples $(X_1, Y_1), \ldots, (X_n, Y_n)$, tuning parameters $\lambda$, $\gamma$, $\varrho$, stopping criteria parameter $\epsilon^{pri}$ and $\epsilon^{dual}$.
Initialize $\beta^{(0)}$ as random uniform numbers; initialize primal and dual residuals $r^{(0)}$ and $s^{(0)}$;
**While** $\|r^{(k)}\|_2 > \epsilon^{pri}$ and $\|s^{(k)}\|_2 > \epsilon^{dual}$ Update $\alpha$ from (6.6): For given $\beta = \beta^{(k)}$ and $\eta = \eta^{(k)}$, $\alpha^{(k+1)} = S_{\varrho,\lambda}\Big(D\beta - \frac{\eta}{\varrho}\Big)$;
Update $\beta$ from (6.5): For given $\alpha = \alpha^{(k+1)}$ and $\eta = \eta^{(k)}$, $\beta^{(k+1)} = \frac{1}{2}\Big(\frac{1}{n}\mathbf{X}_i^T\mathbf{X}_i + \frac{\varrho}{2}D^TD\Big)^{-1}\Big(\frac{2}{n}\sum_{i=1}^{n}\mathbf{X}_i^T\mathbf{Y}_i + D^T\eta + \varrho D^T\alpha\Big)$;
Update $\eta$ by (6.7): For given $\alpha = \alpha^{(k+1)}$ and $\beta = \beta^{(k+1)}$, $\eta^{(k+1)} = \eta^{(k)} + \varrho(\alpha - D\beta)$;
Update $r^{(k)}$ and $s^{(k)}$: $r^{(k+1)} = \alpha^{(k+1)} - D\beta^{(k+1)}, \quad s^{(k+1)} = \varrho D^T(\alpha^{(k+1)} - \alpha^{(k)})$.
**Output** $\beta$ and $\alpha$
```

The output is either β or α. Note that α is a sparse solution and β may not be sparse. In practice, we extract the coefficient image estimator from the output α to obtain the sparse estimator.

6.3.2 Divide and Conquer Learning Algorithm for Large Image Size

To address the big data issue, a divide and conquer (D&C) algorithm is a solution by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem. In the above discussion, we assume that, for each subject i, all image coordinate values $Y_i \in \mathbb{R}^{N^2}$ are used together to infer the coefficient images β. However, in many applications N may be large and such a "batch" procedure is undesirable. In order to solve this issue, we develop a D&C algorithm for large image size.

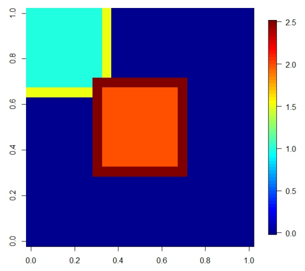


Fig. 6.2. Partition the image into overlapped sub-images.

Image data have their intrinsic structure and we need proceed the divide step with extra caution. For example, we may just partition each image as non-overlap sub-images $Y_i = Y_{i1} \cup Y_{i2} \cup \cdots \cup Y_{iJ}$, with the data processed sequentially. Due to the TV term in SCAD2TV, we lose the boundary information for all sub-images and this will give poor estimates of the boundary for all sub-images. We propose to partition Y_i with overlapped sub-images $Y_i = \tilde{Y}_{i1} \cup \tilde{Y}_{i2} \cup \cdots \cup \tilde{Y}_{iJ}$. For instance, Figure 6.2 displays a 24×24 image, and it is straightforward to partition them into nine nonoverlapped 8×8 sub-images. For our purpose, we extend each sub-image in four directions. Specifically, on the left up corner, we include both the light blue part and the light yellow part, which makes our first sub-image a 9×9 image. In the center, we take the inside 8×8 orange part together with the brown part, which makes it a 10×10 image. After this partition, we obtain nine overlapped sub-images. We perform Algorithm 4 on each sub-image and update the coefficient images. For each update we only keep the estimate for the original 8×8 sub-image.

The above D&C algorithm can be executed sequentially in a single machine. This algorithm is naturally adapted for execution in multi-processor machines, especially shared-memory systems where the communication of data between processors does not need to be planned in advance, because distinct sub-problems can be executed on different processors.

6.4 Empirical Results

6.4.1 Synthetic data

We design a synthetic data example to compare the performance among three approaches: SCAD2TV, GraphNet, and TV-ℓ_1 in terms of both prediction and selection errors.

Data Generation. In our setting, $\beta = (\beta_0, \beta_1, \beta_2)$ where each β_j is a 64×64 image (See the left panel of Figure 6.3). The covariate is $X = (1, X_1, X_2)^T$ and each X_j is generated from a uniform distribution between 0 and 2. The spatial field $\eta(\cdot)$ is generated from a zero mean Gaussian random field. The error process $\epsilon(\cdot)$ is the white noise with mean zero and variance σ^2. Two noise levels are adopted at $\sigma = 1, 0.1$. The sample size is $n = 100$.

Applying SCAD2TV. In order to examine the performances of three methods, SCAD2TV, GraphNet, and TV-ℓ_1, we have generated 100 datasets for each setting. For each dataset, we obtain the coefficient image estimates $\hat{\beta}$ from these three methods. The selection rate is define as

$$\mathrm{SR} = \frac{1}{|S|} \sum_{s \in S} \Big(I(\beta(s) = 0, \hat{\beta}(s) = 0) + I(\beta(s) \neq 0, \hat{\beta}(s) \neq 0) \Big), \tag{6.8}$$

and the mean squared error is defined as

$$\mathrm{MSE} = \frac{1}{n|S|} \sum_{i=1}^{n} \big\| \hat{Y}_i - Y_i \big\|_2^2, \tag{6.9}$$

where $|S| = 4096$ is the total number of pixels and the $\hat{Y}_i$ are the predicted images.

Practical Consideration. Smoothing parameters λ, γ can be selected by using the K-fold cross-validation (CV). However, its computational time can be long even under current computing facilities. In our experiment, we have tested a few different values for the tuning parameters such as $\lambda = 1, 2, \ldots, 10$ and $\gamma = 0.1, 0.2, \ldots, 0.9$. We find $\gamma = 0.5$ is a good balance for the estimation. The value of λ is related to our expectation of ROI. If the ROI has a sharp boundary and the values do not change much inside ROI, we can use a large λ. Otherwise, a smaller λ would be preferred. We choose $\lambda = 5$, $\gamma = 0.5$ and $\rho = 1$.

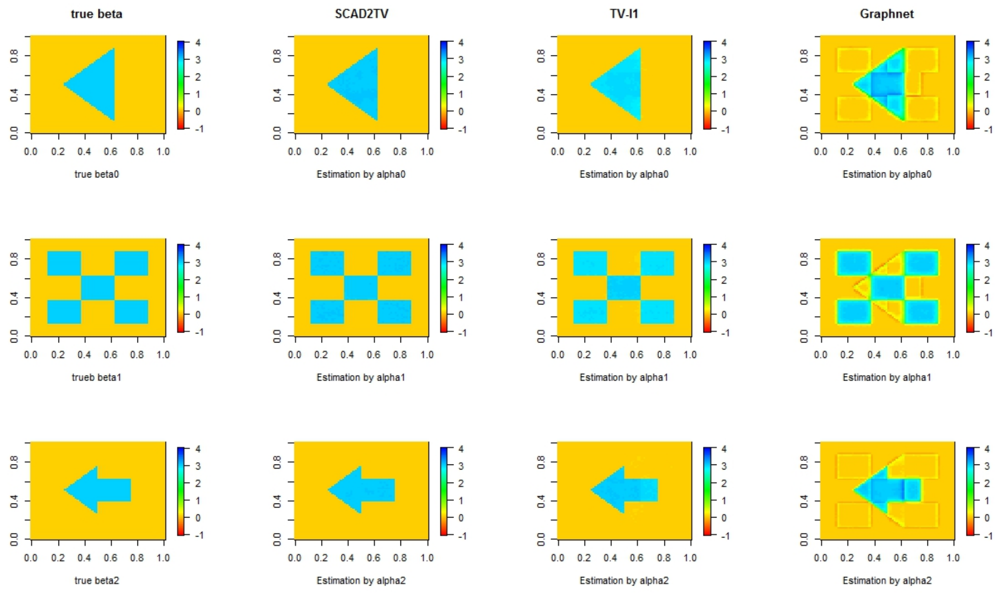


Fig. 6.3. The coefficient images and the estimates. Left:True images; 2nd column: SCAD2TV; 3rd column: TV-ℓ_1; Right: GraphNet.

Results. For each setting, the experiments using SCAD2TV, GraphNet, TV-ℓ_1 are repeated 100 times. Figure 6.3 displays the estimates of coefficient images from one realization. We note that SCAD2TV provides the solution almost exact the same as the truth. TV-ℓ_1 can keep the sharp boundary but provide biased estimates inside the active zone. GraphNet displays blurred estimates for both active zone and zero sub-regions. The average of the selection rates and the MSEs are reported in Table 6.1. It is noted that, in terms of the selection rate, SCAD2TV performs better consistently than the other two methods. On the other hand, in terms of the prediction error, SCAD2TV and TV-ℓ_1 are similar to each other, and GraphNet gives the highest MSE.

Table 6.1. Comparison Results of both SR and MSE for SCAD2TV, TV-ℓ_1, and GraphNet. The bold stands for the best among three methods.

		SCAD2TV		TV-ℓ_1		GraphNet	
		SR	MSE	SR	MSE	SR	MSE
$\sigma = 1$	β_0	**0.820**	**1.97**	0.783	1.98	0.534	1.98
	β_1	0.728		**0.740**		0.583	
	β_2	**0.677**		0.672		0.508	
$\sigma = 0.1$	β_0	**0.9995**	**0.020**	0.9675	0.024	0.8188	0.044
	β_1	**0.9983**		0.9417		0.8210	
	β_2	**0.9990**		0.9070		0.8472	

6.4.2 Hippocampus Data

Dataset. To illustrate the usefulness of our proposed model, consider anatomical MRI data collected at the baseline by the Alzheimer's Disease Neuroimaging Initiative (ADNI) study, which is a large scale multi-site study collecting clinical, imaging, and laboratory data at multiple time points from healthy controls, individuals with amnestic mild cognitive impairment, and subjects with Alzheimer's disease. Given the MRI scans, hippocampal substructures were segmented with FSL FIRST [138] and hippocampal surfaces were automatically reconstructed with the marching cube method [139]. We adopted a surface fluid registration based hippocampal subregional analysis package [140], which uses isothermal coodinates and uid registration to generate one-to-one hippocampal surface registration for surface statistics computation.

In the dataset, we have total 403 observations. For each subject, it includes a 150×100 2D representation of left hippocampus and 4 covariates: gender (female=0 and male=1), age (55-92), disease status (control=0 and AD=1), and behavior score (1-36). The goal is to identify local regions of the response image associated with each covariate.

Applying SCAD2TV. We have applied our D&C learning algorithm to this dataset. We divide each response image into 150 overlapped sub-images. We execute the algorithm sequentially in a single machine. The algorithm spends about 10 secs for each partition, and takes 25 minutes to get the final estimation of β's. The estimated coefficient images are presented in the top panel of Figure 6.4.

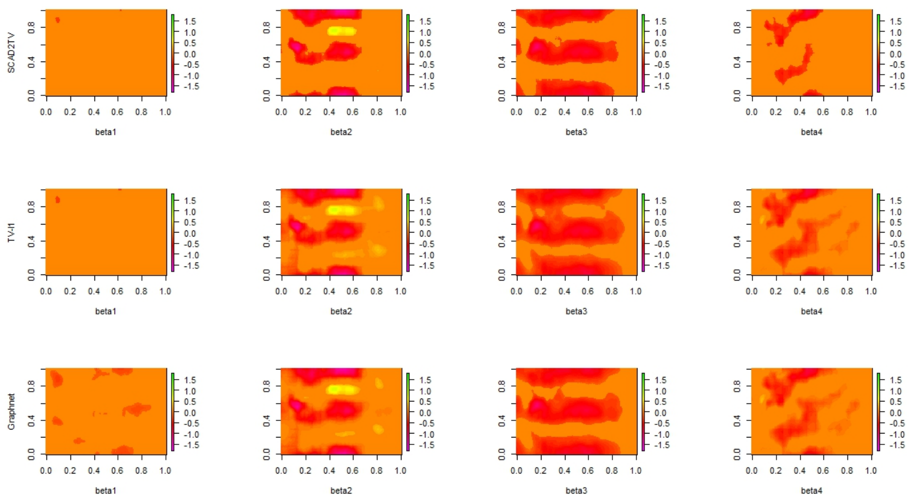


Fig. 6.4. The estimates of coefficient images from three methods. Top: SCAD2TV; Middle: TV-ℓ_1; Bottom: GraphNet.

Results. Our purpose is to identify the local regions where the response hippocampus image is associated with each individual covariate. Among all of the covariates, we are particularly interested in the association between the response hippocampus image and the disease status (Control vs AD). From the top panel of Figure 6.4, it is interesting to notice that gender has no effect on the response image, and for other three covariates SCAD2TV has been successfully identify the local active regions.

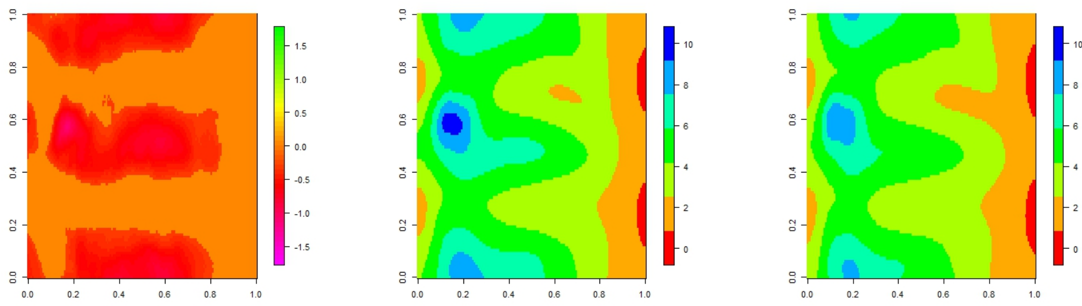


Fig. 6.5. Left: $\hat{\beta}_3$; Middle: Mean response image for health controls; Right: Mean response image for AD.

We take a close investigation on the coefficient image corresponding to the disease status. The left panel of Figure 6.5 displays the estimate of β_3. The sub-regions in red indicate the active zone and the region in orange is the zero sub-region. In general, the AD patients have lower pixel values in the response hippocampus image. The right two panels are the mean response images for both health control and AD. The mean difference is consistent with our estimation of β_3.

We extract the pixels within the ROI for health controls and AD, and apply hypothesis testing to test if their difference is significant. By applying hypothesis testing on each pixel in the ROI, all of them are different between health controls and AD at the significance level 5%, and 99.85% of the pixels in the ROI are different at the significance level 1%. The result justifies our ROI selection is indeed the region to differentiate between health controls and AD.

Comparison with TV-ℓ_1 and GraphNet. We obtain the estimates of the coefficient images for TV-ℓ_1 and GraphNet, which are presented in the middle and bottom panels of Figure 6.4. These three methods overall detects similar regions. Both TV-ℓ_1 and GraphNet display more blocky active regions, whereas SCAD2TV keep the active zone with sharp boundaries. We also divide our dataset into 5 parts to compare the prediction performance where each dataset contains around 80 observations. Every time we use 4 of them as the training data, and make prediction on the remaining testing data. The averages of the MSEs are computed for each methods, which are reported in Table 6.2. The MSEs are similar to each other and SCAD2TV displays a slightly better prediction power.

Table 6.2. The MSEs for three methods

	SCAD2TV	TV-ℓ_1	Graphnet
MSE	**0.5476**	0.5482	0.5491

6.5 Conclusion

We have introduced a new region-selecting sparse non-convex penalty, SCAD2TV, which enforces large regions of zero sub-images and extracts non-zero active zones simultaneously. Efficient algorithm and the distributed algorithm have been developed. Numerical examples are presented and the experimental results are superior or competitive with other state-of-the-art approaches such as GraphNet and TV-ℓ_1. We have so-far focused on 2D images. It should be noted that our method works for 3D images as well. We are currently implementing our algorithm in the distributed platform such as Apache Spark. We have discussed the application for image-on-scalar regression models. This new framework may also be applied to the image clustering and image classification problems, which assume that only small regions of the images have significant effects on clustering and classification.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[2] Gregory Y H Lip, Laurent Fauchier, Saul B Freedman, Isabelle Van Gelder, Andrea Natale, Carola Gianni, Stanley Nattel, Tatjana Potpara, Michiel Rienstra, Hung Fat Tse, and Deirdre A Lane. Atrial fibrillation. *Nature Reviews Disease Primers*, 2:1–26, 3 2016.

[3] A John Camm, Paulus Kirchhof, Gregory YH Lip, Ulrich Schotten, Irene Savelieva, Sabine Ernst, Isabelle C Van Gelder, Nawwar Al-Attar, Gerhard Hindricks, Bernard Prendergast, et al. Guidelines for the management of atrial fibrillation: the task force for the management of atrial fibrillation of the european society of cardiology (esc). *European heart journal*, 31(19):2369–429, 2010.

[4] Roberta Colloca, Luca Minardi, and Gari D Clifford. Implementation and testing of atrial fibrillation detectors for a mobile phone application. *M.Sc.Thesis, Politecnico di Milano and University of Oxford*, 2013.

[5] Gerald V Naccarelli, Helen Varker, Jay Lin, and Kathy L Schulman. Increasing prevalence of atrial fibrillation and flutter in the united states. *The American journal of cardiology*, 104(11):1534–1539, 2009.

[6] Irina Savelieva and John Camm. Update on atrial fibrillation: part i. *Clinical cardiology*, 31(2):55–62, 2008.

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] Yujia Li, Kevin Swersky, and Richard Zemel. Generative moment matching networks. *arXiv preprint arXiv:1502.02761*, 2015.

[10] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

[11] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

[12] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[14] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning*, pages 1558–1566, 2016.

[15] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. It takes (only) two: Adversarial generator-encoder networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[16] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[17] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[18] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2391–2400. JMLR. org, 2017.

[19] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P Xing. On unifying deep generative models. *arXiv preprint arXiv:1706.00550*, 2017.

[20] Xu Chen, Jiang Wang, and Hao Ge. Training generative adversarial networks via primal-dual subgradient methods: a lagrangian perspective on gan. *arXiv preprint arXiv:1802.01765*, 2018.

[21] Shengjia Zhao, Jiaming Song, and Stefano Ermon. The information autoencoding family: A lagrangian perspective on latent variable generative models. *arXiv preprint arXiv:1806.06514*, 2018.

[22] Mevlana Gemici, Zeynep Akata, and Max Welling. Primal-dual wasserstein gan. *arXiv preprint arXiv:1805.09575*, 2018.

[23] Farzan Farnia and David Tse. A convex duality framework for gans. In *Advances in Neural Information Processing Systems*, pages 5248–5258, 2018.

[24] Paulina Grnarova, Kfir Y Levy, Aurelien Lucchi, Nathanael Perraudin, Thomas Hofmann, and Andreas Krause. Evaluating gans via duality. *arXiv preprint arXiv:1811.05512*, 2018.

[25] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

[26] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[27] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 224–232. JMLR. org, 2017.

[28] Haoming Jiang, Zhehui Chen, Minshuo Chen, Feng Liu, Dingding Wang, and Tuo Zhao. On computation and generalization of generative adversarial networks under spectrum control. 2018.

[29] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[30] Xingguo Li, Junwei Lu, Zhaoran Wang, Jarvis Haupt, and Tuo Zhao. On tighter generalization bound for deep neural networks: Cnns, resnets, and beyond. *arXiv preprint arXiv:1806.05159*, 2018.

[31] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[32] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[33] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[34] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[35] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[36] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[37] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018.

[38] John Nash. The embedding problem for riemannina manifolds. *Annals of Mathematics*, pages 20–63, 1956.

[39] Matthias Günther. Isometric embeddings of riemannian manifolds. In *Proceedings of the International Congress of Mathematicians*, pages 1137–1143, 1991.

[40] Umberto Cherubini, Elisa Luciano, and Walter Vecchiato. *Copula methods in finance*. John Wiley & Sons, 2004.

[41] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[42] Colin L Mallows. Some comments on C_p. *Technometrics*, 15(4):661–675, 1973.

[43] Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer, 1998.

[44] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[45] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[46] 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.

[47] 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.

[48] Cun-Hui Zhang et al. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.

[49] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[50] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666, 2007.

[51] Deanna Needell and Joel A Tropp. Cosamp: iterative signal recovery from incomplete and inaccurate samples. *Communications of the ACM*, 53(12):93–100, 2010.

[52] Florentina Bunea et al. Honest variable selection in linear and logistic regression models via ℓ_1 and $\ell_1 + \ell_2$ penalization. *Electronic Journal of Statistics*, 2:1153–1194, 2008.

[53] Sham Kakade, Ohad Shamir, Karthik Sindharan, and Ambuj Tewari. Learning exponential families in high-dimensions: Strong convexity and sparsity. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 381–388, 2010.

[54] Martin J Wainwright. Information-theoretic limits on sparsity recovery in the high-dimensional and noisy setting. *IEEE Transactions on Information Theory*, 55(12):5728–5741, 2009.

[55] Ghislaine Gayraud, Yuri Ingster, et al. Detection of sparse additive functions. *Electronic Journal of Statistics*, 6:1409–1448, 2012.

[56] Vladimir Koltchinskii and Ming Yuan. Sparsity in multiple kernel learning. *The Annals of Statistics*, pages 3660–3695, 2010.

[57] Garvesh Raskutti, Martin J Wainwright, and Bin Yu. Minimax-optimal rates for sparse additive models over kernel classes via convex programming. *Journal of Machine Learning Research*, 13(Feb):389–427, 2012.

[58] Laëtitia Comminges, Arnak S Dalalyan, et al. Tight conditions for consistency of variable selection in the context of high dimensionality. *The Annals of Statistics*, 40(5):2667–2696, 2012.

[59] Yifeng Li, Chih-Yu Chen, and Wyeth W Wasserman. Deep feature selection: Theory and application to identify enhancers and promoters. In *International Conference on Research in Computational Molecular Biology*, pages 205–217. Springer, 2015.

[60] Faming Liang, Qizhai Li, and Lei Zhou. Bayesian neural networks for selection of drug sensitive genes. *Journal of the American Statistical Association*, pages 1–18, 2018.

[61] Jean Feng and Noah Simon. Sparse-input neural networks for high-dimensional nonparametric regression and classification. *arXiv preprint arXiv:1711.07592*, 2017.

[62] Genevera I Allen. Automatic feature selection via weighted kernels and regularization. *Journal of Computational and Graphical Statistics*, 22(2):284–299, 2013.

[63] Jingxiang Chen, Chong Zhang, Michael R Kosorok, and Yufeng Liu. Double sparsity kernel learning with automatic variable selection and data extraction. *Statistics and its interface*, 11(3):401–420, 2018.

[64] Alex Lapanowski and Irina Gaynanova. Sparse feature selection in kernel discriminant analysis via optimal scoring. In *AISTATS*, 2019.

[65] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

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

[67] Sohail Bahmani, Bhiksha Raj, and Petros T Boufounos. Greedy sparsity-constrained optimization. *Journal of Machine Learning Research*, 14(Mar):807–841, 2013.

[68] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

[69] Cun-Hui Zhang, Tong Zhang, et al. A general theory of concave regularization for high-dimensional sparse estimation problems. *Statistical Science*, 27(4):576–593, 2012.

[70] Jiahua Chen and Zehua Chen. Extended bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008.

[71] Xin Gao and Peter X-K Song. Composite likelihood bayesian information criteria for model selection in high-dimensional data. *Journal of the American Statistical Association*, 105(492):1531–1540, 2010.

[72] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[73] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[74] P Breheny. ncvreg: Regularization paths for scad-and mcp-penalized regression models. *R package version*, pages 2–6, 2013.

[75] Pradeep Ravikumar, Han Liu, John Lafferty, and Larry Wasserman. Spam: Sparse additive models. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 1201–1208. Curran Associates Inc., 2007.

[76] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[77] Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

[78] Justin Bleich, Adam Kapelner, Edward I George, and Shane T Jensen. Variable selection for bart: An application to gene regulation. *The Annals of Applied Statistics*, pages 1750–1781, 2014.

[79] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, Joseph Lehár, Gregory V Kryukov, Dmitriy Sonkin, et al. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603, 2012.

[80] 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.

[81] Katie E Hadley and Denver T Hendricks. Use of nqo1 status as a selective biomarker for oesophageal squamous cell carcinomas with greater sensitivity to 17-aag. *BMC cancer*, 14(1):334, 2014.

[82] Ho Jeong Lee, Masaki Hanibuchi, Sun-Jin Kim, Hyunkyung Yu, Mark Seungwook Kim, Junqin He, Robert R Langley, François Lehembre, Urs Regenass, and Isaiah J Fidler. Treatment of experimental human breast cancer and lung cancer brain metastases in mice by macitentan, a dual antagonist of endothelin receptors, combined with paclitaxel. *Neuro-oncology*, 18(4):486–496, 2016.

[83] Stephanie N Dorman, Katherina Baranova, Joan HM Knoll, Brad L Urquhart, Gabriella Mariani, Maria Luisa Carcangiu, and Peter K Rogan. Genomic signatures for paclitaxel and gemcitabine resistance in breast cancer derived by machine learning. *Molecular oncology*, 10(1):85–100, 2016.

[84] Gabriele Zoppoli, Marie Regairaz, Elisabetta Leo, William C Reinhold, Sudhir Varma, Alberto Ballestrero, James H Doroshow, and Yves Pommier. Putative dna/rna helicase schlafen-11 (slfn11) sensitizes cancer cells to dna-damaging agents. *Proceedings of the National Academy of Sciences*, 109(37):15030–15035, 2012.

[85] Jonas Mockus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.

[86] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[87] Jiří Matoušek. Improved upper bounds for approximation by zonotopes. *Acta Mathematica*, 177(1):55–73, 1996.

[88] Elif Derya Übeyli. ECG beats classification using multiclass support vector machines with error correcting output codes. *Digital Signal Processing*, 17(3):675–684, 2007.

[89] Elif Derya Übeyli. Combining recurrent neural networks with eigenvector methods for classification of ECG beats. *Digital Signal Processing*, 19(2):320–329, 2009.

[90] Yonghan Jung and Heeyoung Kim. Detection of PVC by using a wavelet-based statistical ECG monitoring procedure. *Biomedical Signal Processing and Control*, 36:176–182, 2017.

[91] Khaled Daqrouq, A Alkhateeb, MN Ajour, and Ali Morfeq. Neural network and wavelet average framing percentage energy for atrial fibrillation classification. *Computer methods and programs in biomedicine*, 113(3):919–926, 2014.

[92] Adfal Afdala, Nuryani Nuryani, and Anto Satrio Nugroho. Detection of atrial fibrillation using artifical neural network with power spectrum density of rr interval of electrocardiogram. In *Journal of Physics: Conference Series*, volume 795, page 012073. IOP Publishing, 2017.

[93] Pranav Rajpurkar, Awni Y Hannun, Masoumeh Haghpanahi, Codie Bourn, and Andrew Y Ng. Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*, 2017.

[94] Ziad Sankari and Hojjat Adeli. Heartsaver: a mobile cardiac monitoring system for auto-detection of atrial fibrillation, myocardial infarction, and atrio-ventricular block. *Computers in biology and medicine*, 41(4):211–220, 2011.

[95] Giulia Da Poian, Chengyu Liu, Riccardo Bernardini, Roberto Rinaldo, and Gari D Clifford. Atrial fibrillation detection on compressed sensed ECG. *Physiological measurement*, 38(7):1405, 2017.

[96] Maryam Mohebbi and Hassan Ghassemian. Prediction of paroxysmal atrial fibrillation using recurrence plot-based features of the rr-interval signal. *Physiological measurement*, 32(8):1147, 2011.

[97] Physionet. AF classification from a short single lead ECG recording: The Physionnet/computing in cardiology challenge 2017, 2017. https://www.physionet.org/challenge/2017/.

[98] Gari Clifford, Chengyu Liu, Benjamin Moody, L Lehman, Ikaro Silva, Qiao Li, Alistair Johnson, and Roger Mark. AF classification from a short single lead ECG recording: The Physionet computing in cardiology challenge 2017. *Computing in Cardiology*, 44, 2017.

[99] 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. ACM, 2016.

[100] Kaggle Inc. Kaggle: The home of data science & machine learning, 2018. https://www.kaggle .com/.

[101] Valentin Fuster, Lars E Rydén, Richard W Asinger, David S Cannom, Harry J Crijns, Robert L Frye, Jonathan L Halperin, G Neal Kay, Werner W Klein, Samuel Lévy, et al. ACC/AHA/ESC guidelines for the management of patients with atrial fibrillation: executive summary a report of the american college of cardiology/american heart association task force on practice guidelines and the european society of cardiology committee for practice guidelines and policy conferences (committee to develop guidelines for the management of patients with atrial fibrillation) developed in collaboration with the north american society of pacing and electrophysiology. *Circulation*, 104(17):2118–2150, 2001.

[102] Carlos Carreiras. BioSPPy: Biosignal processing in Python, 2015. https://github.com/PIA-Group/BioSPPy.

[103] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. http://www.scipy.org/.

[104] Physionet. Sandbox evaluation of entries in Physionet/computing in cardiology challenges, 2017. https://www.physionet.org/challenge/sandbox/.

[105] Ivaylo I Christov. Real time electrocardiogram qrs detection using combined adaptive threshold. *Biomedical engineering Misc*, 3(1):28, 2004.

[106] WAH Engelse and C Zeelenberg. A single scan algorithm for qrs-detection and feature extraction. *Computers in cardiology*, 6(1979):37–42, 1979.

[107] W Zong, T Heldt, GB Moody, and RG Mark. An open-source algorithm to detect onset of arterial blood pressure pulses. In *Computers in Cardiology, 2003*, pages 259–262. IEEE, 2003.

[108] Pat Hamilton. Open source ECG analysis. In *Computers in Cardiology, 2002*, pages 101–104. IEEE, 2002.

[109] Hugo Gamboa. *Multi-Modal Behavioral Biometrics Based on HCI and Electrophysiology*. PhD thesis, PhD thesis, Universidade Técnica de Lisboa, Instituto Superior Técnico, 2008.

[110] Filipe Canento, André Lourenço, Hugo Silva, and Ana Fred. Review and comparison of real time electrocardiogram segmentation algorithms for biometric applications. In *Proceedings of the 6th Int'l Conference on Health Informatics (HEALTHINF)*, 2013.

[111] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet. *Circulation*, 101(23):e215–e220, 2000.

[112] TR Gopalakrishnan Nair, AP Geetha, and M Asharani. Adaptive wavelet based identification and extraction of pqrst combination in randomly stretching ECG sequence. In *Signal and Information Processing (ChinaSIP), 2013 IEEE China Summit & International Conference on*, pages 278–282. IEEE, 2013.

[113] Ryan J Tibshirani et al. Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42(1):285–323, 2014.

[114] Krishnavedala. A function (blue) and a piecewise linear approximation to it (red), 2011. https://upload.wikimedia.org/wikipedia/commons/5/53/SinusRhythmLabels.png.

[115] Xiao Wang and Chuanhai Liu. Adaptive piecewise linear regression and deep learning. *manuscript*, 2018. in preparation.

[116] The McGraw-Hill Companies Inc. Mcgraw-hill concise dictionary of modern medicine. https://medical-dictionary.thefreedictionary.com/F+wave.

[117] Katerina Hnatkova, Xavier Copie, Anne Staunton, and Marek Malik. Numeric processing of lorenz plots of rr intervals from long-term ecgs: comparison with time-domain measures of heart rate variability for risk stratification after myocardial infarction. *Journal of electrocardiology*, 28:74–80, 1995.

[118] 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*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[119] GitHub. XGBoost python package. https://github.com/dmlc/xgboost.

[120] George B Moody and Roger G Mark. The impact of the mit-bih arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

[121] S.M. Berg, E.A. Bittner, and K.H. Zhao. *Anesthesia Review: Blasting the Boards*. Lippincott Williams & Wilkins, 2016.

[122] R Alcaraz, C Vayá, R Cervigón, C Sánchez, and JJ Rieta. Wavelet sample entropy: A new approach to predict termination of atrial fibrillation. In *Computers in Cardiology, 2006*, pages 597–600. IEEE, 2006.

[123] Juan Ródenas, Manuel García, Raúl Alcaraz, and José J Rieta. Wavelet entropy automatically detects episodes of atrial fibrillation from single-lead electrocardiograms. *Entropy*, 17(9):6179–6199, 2015.

[124] Manuel García, Juan Ródenas, Raúl Alcaraz, and José J Rieta. Application of the relative wavelet energy to heart rate independent detection of atrial fibrillation. *computer methods and programs in biomedicine*, 131:157–168, 2016.

[125] Yangling Mu and Fred H Gage. Adult hippocampal neurogenesis and its role in alzheimer's disease. *Molecular neurodegeneration*, 6(1):85, 2011.

[126] Michael Eickenberg, Elvis Dohmatob, Bertrand Thirion, and Gaël Varoquaux. Grouping total variation and sparsity: statistical learning with segmenting penalties. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 685–693. Springer, 2015.

[127] Elvis Dopgima Dohmatob, Alexandre Gramfort, Bertrand Thirion, and Gael Varoquaux. Benchmarking solvers for tv-ℓ_1 least-squares and logistic regression in brain imaging. In *2014 International Workshop on Pattern Recognition in Neuroimaging*, pages 1–4. IEEE, 2014.

[128] Abolfazl Mehranian, Hamidreza Saligheh Rad, Arman Rahmim, Mohammad Reza Ay, and Habib Zaidi. Smoothly clipped absolute deviation (scad) regularization for compressed sensing mri using an augmented lagrangian scheme. *Magnetic resonance imaging*, 31(8):1399–1411, 2013.

[129] David M Strong, Peter Blomgren, and Tony F Chan. Spatially adaptive local-feature-driven total variation minimizing image restoration. In *Statistical and Stochastic Methods in Image Processing II*, volume 3167, pages 222–233. International Society for Optics and Photonics, 1997.

[130] 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.

[131] Aditya Chopra and Heng Lian. Total variation, adaptive total variation and nonconvex smoothly clipped absolute deviation penalty for denoising blocky images. *Pattern Recognition*, 43(8):2609–2619, 2010.

[132] Yakuan Chen, Jeff Goldsmith, and R Todd Ogden. Variable selection in function-on-scalar regression. *Stat*, 5(1):88–101, 2016.

[133] Jeff Goldsmith, Vadim Zipunnikov, and Jennifer Schrack. Generalized multilevel function-on-scalar regression and principal component analysis. *Biometrics*, 71(2):344–353, 2015.

[134] Leonid I Rudin and Stanley Osher. Total variation based image restoration with free local constraints. In *Proceedings of 1st International Conference on Image Processing*, volume 1, pages 31–35. IEEE, 1994.

[135] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[136] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63, 2019.

[137] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[138] Brian Patenaude, Stephen M Smith, David N Kennedy, and Mark Jenkinson. A bayesian model of shape and appearance for subcortical brain segmentation. *Neuroimage*, 56(3):907–922, 2011.

[139] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[140] Jie Shi, Paul M Thompson, Boris Gutman, Yalin Wang, Alzheimer's Disease Neuroimaging Initiative, et al. Surface fluid registration of conformal representation: Application to detect disease burden and genetic influence on hippocampus. *NeuroImage*, 78:111–134, 2013.

VITA

Yao Chen received a bachelor's degree in Statistics from the University of Science and Technology of China in 2013, a master's degree in Actuarial Science from the University of Iowa in 2015 and earned a doctoral degree in Statistics from Purdue University in 2020. His research interests include deep learning, machine learning, sparse learning and high dimensional statistics.