Trainable Sparse Coding with ℓ_p-norm-based Regularization

Haoli Zhao

A DISSERTATION

SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

IN COMPUTER SCIENCE AND ENGINEERING

Graduate Department of Computer and Information Systems

The University of Aizu

2019

The thesis titled

Trainable Sparse Coding with ℓ_p-norm-based Regularization

by

Haoli Zhao

is reviewed and approved by:

Chief Referee

Associate Professor

Xiang Li Xiang Li

Senior Associate Professor

Xin Zhu Xin Zhu

Associate Professor

Yuichi Okuyama

Senior Associate Professor

Konstantin Markov Konstantin Markov

Professor

Shuxue Ding 丁数学

THE UNIVERSITY OF AIZU

September 2019

Contents

List of Figures

List of Tables

List of Abbreviations

BP	Basis Pursuit
CDA	the Coordinate Descent Algorithm
CS	Compressed Sensing
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
DNN-SC	Deep Neural Network structured Sparse coding
DNN-ISTA	DNN-structured ISTA, the same are DNN-IHTA, DNN-WISTA and so on
LISTA	Supervised Learned ISTA, 'L' stands for supervised learning, the same are LIHTA, LWISTA and so on
TISTA	Unsupervised Trained ISTA, 'T' stands for unsupervised learning, the same are TIHTA, TWISTA and so on
FOCUSS	Focal Underdetermined System Solver
FOCUSSDL	FOCUSS based dictionary learning algorithm
HDLWL	Hierarchical Dictionary Learning with Weighted ℓ_1 norm
II-DNN-SC	independently interpretable DNN-SC
IHT	Iterative Hard Thresholding
IHTA	Iterative Half Thresholding Algorithm
II-ISTA	Independently Interpretable ISTA
IILasso	Independently Interpretable Lasso
IIWLasso	Independently Interpretable Weighted Lasso
IIWL	one special case of IIWLasso that choosing $p \in (0, 1)$ and $q = 1$
IIWR	one special case of IIWLasso that choosing $p = 1$ and $q \in (0, 1)$
IIWW	one special case of IIWLasso that choosing $p = q \in (0, 1)$
ISTA	Iterative Shrinkage Thresholding Algorithm
K-SVD	K-means Singular Value Decomposition
Lasso	Least Absolute Shrinkage and Selection Operator
MOD	Method of Optimal Directions
OMP	Orthogonal Matching Pursuit
PDLA	The Proximal Dictionary Learning Algorithm
PO	Proximal Operator
PSNR	Peak Signal-to-Noise Ratio
SNR	Signal-to-Noise Ratio
SVD	Singular Value Decomposition
WISTA	Weighted Iterative Shrinkage Thresholding Algorithm
WISTA0.9	The number 0.9 after WISTA stands for the p value used in WISTA, The same are the other weighted algorithms
2/3PO	the algorithm which use PO to solve $\ell_{\frac{2}{3}}$ norm regularization
IIPO	Independently Interpretable PO
II2/3PO	Independently Interpretable 2/3PO

List of Symbols

$\mathbf{A}$	Matrix
$\mathbf{a}$	Vector
$\mathbf{a}_i$	The i-th element of vector $\mathbf{a}$
$\mathbf{A}_{i:}$	The i-th row of $\mathbf{A}$
$\mathbf{A}_{:j}$	The j-th column of $\mathbf{A}$
$\mathbf{A}^{\mathrm{T}}$	Transposed matrix of $\mathbf{A}$
$\mathbf{A}^{-1}$	The inverse matrix of the matrix $\mathbf{A}$
$\mathbf{A}_{ij}$	The entry at the i-th row and j-th column of $\mathbf{A}$
$\|\mathbf{a}\|_0$	ℓ_0 norm of $\mathbf{a}$
$\|\mathbf{a}\|_1$	ℓ_1 norm of $\mathbf{a}$
$\|\mathbf{a}\|_2$	ℓ_2 norm of $\mathbf{a}$
$\|\mathbf{a}\|_p$	ℓ_p norm of $\mathbf{a}$
$\|\mathbf{A}\|_F$	Frobenius norm of $\mathbf{A}$
$\|\mathbf{A}\|_1$	ℓ_1 norm of $\mathbf{A}$ ($\sum_{i,j} \|\mathbf{A}_{ij}\|$)
$\mathbb{R}$	Real numbers
$\mathbf{I}$	Identity matrix
$\mathbf{D}$	Synthesis dictionary, sized as $m \times n$
$\mathbf{x}$	Input data vector, sized as $m \times 1$
$\mathbf{X}$	Input data set, sized as $m \times N$, $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)$
$\mathbf{z}$	Sparse representation vector, sized as $n \times 1$
$\mathbf{Z}$	Sparse representation set, sized as $n \times N$, $\mathbf{Z} = (\mathbf{z}_1, ..., \mathbf{z}_N)$
$\mathbf{z}^{(k)}$	k in the bracket stands for number of iterations or layers in algorithms
$\mathbf{R}$	coherence based matrix for regularization, sized as $n \times n$
$\pi_1(x, t)$	Soft thresholding operator, the same is $\pi_t(x)$
$\pi_{\frac{1}{2}}(x, t)$	Half thresholding operator for $\ell_{\frac{1}{2}}$ norm regularization
$\pi_{\frac{2}{3}}(x, t)$	$\ell_{\frac{2}{3}}$ norm regularization proximal operator

Acknowledgment

The completion of this study could not have been possible without the expertise of Professor Shuxe Ding and Professor Xiang Li, my mentors and supervisors during the doctoral project. Your guidance, patience, and continuous encouragement help me going through difficulties and obstacles in this study.

I would also like to use this opportunity to give my thanks and appreciation to the members of my dissertation committee, Professor Xin Zhu, Professor Yuichi Okuyama and Professor Konstantin Markov. Thanks for your suggestions and advises for this thesis.

A debt of gratitude is also owed to colleagues at Cognitive Science Laboratory, University of Aizu, Yujie Li, Zhenni Li, Benyin Tan, Huakun Huang and Lingjun Zhao. Thanks for their dedicated assistance and infinite support throughout my studies.

Last but not the least, this thesis would not have been possible without the most devoted, supportive, and caring family. To my parents, Gang Zhao and Zhen Tang, I would like to send my deepest gratitude, appreciation and affection for the unconditional support, persistent encouragement, inspirational advice, and endless love. To my dear girlfriend, Tiantian Sun, thanks for your patience and understanding. You have been a guide, a motivator, a comfort, and a role model, and for all these and more I am deeply indebted to you.

Abstract

Sparse representation, which aims at finding appropriate sparse representations of data with an overcomplete dictionary set, has been proven to be a powerful tool for analysis and processing of various signals. Performance of sparse representation mainly depends on a well-defined dictionary and an appropriate sparse constraint for corresponding data.

In this thesis, we concentrate on emphasizing the importance of ℓ_p norm ($0 < p < 1$) in three different directions in sparse representation to show its potential in enhancing sparsity and accuracy. At first, we consider a dictionary learning problem with ℓ_p norm ($0 < p < 1$) regularization. The algorithms use two approximations for ℓ_p norm ($0 < p < 1$) which make the optimization problem convex and smooth during iterations, and thus we can use gradient descent to update both dictionary and sparse representation sets. We validate in synthetic data experiments that the proposed dictionary learning algorithm can recover dictionary to 100% while obtaining accurate sparse representation set.

Then, we propose to construct a family of ℓ_p norm ($0 < p < 1$) based Deep Neural Network structured Sparse Coding (DNN-SC) algorithms. DNN-SC uses parameter training methodology in Recurrent Neural Network (RNN) to train parameters in a truncated sparse coding algorithm. The encoder with well-trained parameters can perform as good as converged sparse coding algorithms while obviously enhancing efficiency. In both synthetic data experiments and denoising experiments for real images and videos, we show that ℓ_p norm ($0 < p < 1$) based DNN-SC algorithms can obtain better performances. We also validate that by using unsupervised ℓ_p norm ($0 < p < 1$) based DNN-SC algorithm, it is possible to conduct online video denoising for 25 frames/s (360×480 pixels per frame) gray-scaled videos using CPU only.

What is more, we show how a well-posed sparsity constraint can affect performances in highly correlated data by introducing ℓ_p norm ($0 < p < 1$) into the regularization part of Independently Interpretable Lasso (IILasso). The regularization of IILasso introduces coherence information in dictionary to implement the strategy of selecting uncorrelated variables which functions well in various highly correlated data. We show that the new independently interpretable regularization with ℓ_p norm ($0 < p < 1$) can obtain smaller relative norm error and support error. Furthermore, we construct DNN-SC algorithms based on independently interpretable algorithms. Synthetic data experiments show that DNN-SC can also help enhancing efficiency of independently interpretable algorithms.

Chapter 1

Introduction

1.1 Sparse Models for Signal Representation

Many natural and measure signals are distributed in a high dimensional space. They can be modelled using low-dimensional structures and be represented using just a few variables in an appropriate model [1]. Signal models are the cores of various signal processing tasks, such as compression, denoising, sampling, classification and so on [2]. Signal models are a fundamental tool for facilitating the distinctiveness of the interesting signals. A signal model formulates a mathematical description of the family of interesting signals, which is the guarantee to distinguish them from the rest of the signal space. Essentially, a signal model is a set of mathematical properties that the data is believed to satisfy [2]. Moreover, a good signal model should be simple while matching the signals.

With advancements in mathematics, linear representation methods have been well studied and have recently received considerable attention [3, 4]. In the past decade, the sparse and redundant representation model has been proved to be important and useful [5–7]. Sparse representation, from the viewpoint of its origin, is directly related to compressed sensing (CS) [8–10], which is one of the most popular topics recently. In [8], Donoho first proposed the original concept of compressed sensing. CS theory suggests that if a signal is sparse or compressive, the original signal can be reconstructed by exploiting a few measured values, which are much less than the ones suggested by the theories used previously. Candès et al. [9], from the mathematical perspective, demonstrated the rationale of CS theory, i.e. the original signal could be precisely reconstructed by utilizing a small portion of Fourier transformation coefficients. These literature [8, 9, 11–13] laid the foundation of CS theory and provided the theoretical basis for future

research. What is more, CS theory always includes the three basic components: sparse representation, encoding measuring, and reconstructing algorithm. As an indispensable prerequisite of CS theory, the sparse representation theory is the most outstanding technique used to conquer difficulties which appear in many fields. Sparse representation has attracted much attention In the past decade. It has also been proven to be an extraordinary powerful solution to a wide range of application fields, especially in signal processing, image processing, machine learning, and computer vision, such as denoising, debluring, inpainting, restoration, super-resolution, visual tracking, classification and segmentation [14–20].

The goal of sparse representation is to approximate a signal by a linear combination of a small number of elementary components, called atoms, which are chosen from an overcomplete dictionary (the number of atoms is greater than the dimension of the signal). Approaches of sparse representation is conducted by solving the underdetermined linear system,

$$\mathbf{X} = \mathbf{DZ}, \tag{1.1}$$

or

$$\mathbf{X} \approx \mathbf{DZ}, \quad s.t. \ \|\mathbf{X} - \mathbf{DZ}\|_F^2 \leq \varepsilon, \tag{1.2}$$

where $\mathbf{X} \in \mathbb{R}^{m \times N}$ is the signal set that we are to model and process, $\mathbf{D} \in \mathbb{R}^{m \times n}$, $(m < n)$, is a possible dictionary [2]. Since $m < n$, where it contains n atoms of size $m \times 1$, the dictionary is column overcomplete, i.e., redundant. Here $\mathbf{Z} \in \mathbb{R}^{n \times N}$ is the representation coefficient matrix. In the model of sparse representation, it is assumed that we can use few atoms to represent every signal in $\mathbf{X}$, thus the representation $\mathbf{Z}$ is supposed to be sparse, that is, most elements are 0-valued. The name "synthesis" comes from the relation (1.1), with the interpretation that the model describes a way to use $\mathbf{D}$ and $\mathbf{Z}$ to synthesize signals $\mathbf{X}$ [21].

1.2 Related works

Since $\mathbf{D}$ is overcomplete, this signal reconstruction task is ill-posed with infinite solutions if there is no restriction on $\mathbf{Z}$. Assuming that we have an appropriate dictionary, to find a sparsest one in the infinite solutions can be a meaningful mode and can make the problem well-posed, which is the essence of sparse coding. Therefore, the optimization problem is commonly used

to search for the optimal solution of the underdetermined linear system,

$$\min_{\mathbf{Z}} g(\mathbf{Z}) \text{ subject to } \|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2 \leq \epsilon, \tag{1.3}$$

where $g(\mathbf{Z})$ functions as the sparsity constraint. $g(\mathbf{Z}) = \|\mathbf{Z}\|_p^p$ is the most commonly used sparse regularization. The sparsest solution can be guaranteed when using the ℓ_0 norm, $p = 0$, where the regularization counts the number of nonzero entries in $\mathbf{Z}$. Orthogonal matching pursuit (OMP) is successful sparse coding algorithm with ℓ_0 norm [21], which can always find sufficiently sparse solutions. Though OMP is not computationally prohibited with large data size, it still faces the problem of high computational complexity. By accepting relaxation on the sparse constraint, we have the the ℓ_1 norm, $p = 1$, which is often selected as the sparsity constraint for its reasonable sparsity-pursuing ability and convex property, for example, the Iterative Shrinkage Thresholding Algorithm (ISTA) [22], Least Absolute Shrinkage and Selection Operator (Lasso) using CDA [23, 24], Basis Pursuit (BP) [25] and FOCal Underdetermined System Solver (FOCUSS) [26]. While searching for a sparser and more accurate representations, the ℓ_p norm, where $p \in (0, 1)$, can be an alternative choice as a relaxation approach. Although the ℓ_p norm $(0 < p < 1)$ makes the optimization problem nonconvex, it has proven its possibility in enhancing the sparsity and accuracy of solutions compared to the ℓ_1 norm. For example, the $\ell_{1/2}$ norm regularization [27,28], $\ell_{2/3}$ norm regularization [28,29], weighted approximation for ℓ_p norm $(0 < p < 1)$ [30–34]. Except the most commonly used sparse regularization, there are some other sparse reglarizations which use different functions to define the sparsity of the coefficient matrix. For example, one can use the combination of logarithm and the absolute value function to define the sparsity constraint, $g(\mathbf{Z}) = \sum_{j=1}^{N} \sum_{i=1}^{n} \log(1 + |\mathbf{Z}_{ij}|/\delta)$ [35, 36], where δ is a positive constant. The log-regularizer functions well as a sparsity constraint but it is also nonconvex and nonsmooth.

However, the above sparse regularization only consider low coherence conditions of data, and typical theoretical supports are also based on small correlation assumptions such as restricted eigenvalue condition [37, 38]. When signals are highly correlated, the coefficient also tends to be jointly correlated. Consequently, sparse coding with normal sparse regularization cannot provide efficient evidence for judging independent variable contribution and interpreting the highly correlated model. The process of interpreting models is named as "decomposability" in [39], representing for the ability of how a model can be decomposed into several parts and

its components can be interpreted separately. In 2018, Takada et al. propose a new regularization, named Independently Interpretable Lasso (IILasso), which is composed with coherence between dictionary columns to enhance the ability of choosing uncorrelated variables in sparse coding [40]. IILasso has proven to be efficient in highly correlated data and provides smaller misclassification error than several other sparse coding algorithms. However, IILasso has the same problem as the other ℓ_1 norm based regularization that its result is not sparse and accurate enough.

To ensure the effectiveness of the model, the dictionary $\mathbf{D}$ is supposed to fit the signal set $\mathbf{X}$ well, whereas we can hardly use few atoms to represent signals. Building dictionary $\mathbf{D}$ can be generally divided into two categories. One is choosing a prespecified matrix as dictionary, for example, the discrete cosine transform for JPEG, the discrete wavelet transform for JPEG2000 and the discrete Fourier transform [41, 42]. Another one considers learning a self-adaptive dictionary based on signal set during procedure, which can usually result in better matching to the contents of the signals and then result in better performance in sparsity and accuracy. Dictionary learning is a classical methodology to train signal-adaptive parameter for sparse coding, and the optimization problem becomes,

$$\min_{\mathbf{D},\mathbf{Z}} g(\mathbf{Z}) \text{ subject to } \|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2 \leq \epsilon, \tag{1.4}$$

Dictionary learning is usually achieved by alternatively updating sparse representation set and dictionary, that is, a iteration in dictionary learning algorithms usually contain a sparse coding stage and a dictionary learning stage. K-means Singular Value Decomposition (K-SVD) [43, 44] and the Method of Optimal Directions (MOD) [45] are two classical dictionary learning algorithms which both utilize OMP for their sparse coding stage, but use different method for their dictionary updating. FOCUSS-CNDL [26] is a ℓ_1 norm based dictionary learning example. Moreover, the Proximal Dictionary Learning Algortihm (PDLA) [36] use the log-regularizer as the sparsity constraint.

To further enhance efficiency of obtaining sparse representations based on the sparse model, it is possible to use the DNN structure and corresponding learning procedure in sparse coding based the structure similarity between iterative shrinkage sparse coding algorithms and Recurrent Neural Network (RNN). Deep Neural Network structured Sparse Coding (DNN-SC) is a methodology which aims at training a encoder from a truncated sparse coding algorithm to

approximate converged performance of the original algorithm. In 2010, Gregor and LeCun introduced the idea with ISTA, where the learned DNN encoder can perform nearly 10 times faster than the original sparse coding algorithm, i.e., ISTA [46]. In 2015, Sprechmann et al. showed that it is possible to learn the DNN without requiring the true answers to serve as the training labels, i.e., the unsupervised learning is possible [47]. Furthermore, the methodology of DNN-SC have been applied in different sparse coding algorithms with similar structure as ISTA, for example, Iterative Hard Thresholding (IHT) [48], Approximate Message Passing (AMP) [49] and so on.

1.3 Motivations and Contributions

As mentioned above, studies on sparse representation model have led to several directions for optimizing sparse representation algorithms, namely developing new sparsity constraints, developing new dictionary learning algorithms, and developing DNN-SCs. ℓ_1 norm is attractive and have been implemented in the three directions. However, ℓ_1 norm regularizations usually have the problem that results are not sparse and accurate enough. Therefore, we try to address importance of ℓ_p norm $(0 < p < 1)$ in the three directions for optimizing sparse representation algorithms.

In Chpater 3, we presented a dictionary learning algorithm to train a data adaptive dictionary with ℓ_p norm $(0 < p < 1)$ regularization. The algorithm utilizes two approximations in the regularization, namely weighted ℓ_1 norm for ℓ_p norm and a smoothed approximation for the absolute value, to make it possible to use gradient descent method to update both sparse coefficient set and dictionary.

In Chapter 4, we proposed to implement parameter training methods of RNN in ℓ_p norm $(0 < p < 1)$ based iterative shrinkage algorithms. We show how we can formulate ℓ_p norm $(0 < p < 1)$ based DNN-SCs by unfolding truncated iterations of their original algorithms. Moreover, we present how we can train DNN-SCs supervisedly or unsupervisedly. Furthermore, we validate that DNN-SCs are possible to conduct online video denoising in experiments.

In Chapter 5, we introduce ℓ_p norm $(0 < p < 1)$ to the coherence related regularization of IILasso for processing highly correlated data. To efficiently solve the non-convex problem brought by ℓ_p norm $(0 < p < 1)$, we use CDA with weighted ℓ_1 norm and the Proximal Operator (PO) to solve the optimization problem with the new regularization. Moreover, we validate that

our proposed algorithm can interpret the "decomposability" of highly correlated gene expression datasets and thus present reasonable suggestions for diseases and developmental stages on gene expression.

In Chapter 6, we further enhance efficiency of different independently interpretable algorithms. We present how to build the structures and train parameters for independently interpretable DNN-SC algorithms.

1.4 Thesis Outline

We start by providing a more detailed introduction about sparse coding, dictionary learning, DNN-SC in Chapter 2. In this Chapter, we mainly review some methods and techniques that can be used for sparse coding, dictionary learning and DNN-SC. In Chapter 3, we present a dictionary learning algorithm with ℓ_p norm $(0 < p < 1)$. In Chapter 4, we propose to construct ℓ_p norm $(0 < p < 1)$ based DNN-SCs for online processing. In Chapter 5, we introduce ℓ_p norm $(0 < p < 1)$ in the regularization part of IILasso for processing highly correlated data. In Chapter 6, we further construct DNN-SCs for algorithms in Chapter 5 to enhance efficiency of algorithms. In Chapter 7 we summarize our contributions in the thesis and present some directions for further research. A brief thesis structure diagram is present in Figure 1.1.

1.5 Publications

The following papers have been published or submitted and currently under review in peer reviewed journals and conferences. Most of the results presented in Chapter 3, 4 and 5 are published in these works.

Journals

[1] **Haoli Zhao**, Shuxue Ding, Xiang Li and Huakun Huang, "Deep Neural Network Structured Sparse Coding for Online Processing," *IEEE Access*, 6, pp.74778-74791, 2018.

[2] **Haoli Zhao**, Shuxue Ding, Xiang Li and Lingjun Zhao, "ℓ_p Norm Independently Interpretable Regularization based Sparse Coding for Highly Correlated Data," *IEEE Access*, 7, pp.53542-53554, 2019.

[3] Huakun Huang, **Haoli Zhao**, Xiang Li, Shuxue Ding, Lingjun Zhao and Zhenni Li, "An Accurate and Efficient Device-Free Localization Approach Based on Sparse Coding in Subspace," *IEEE Access*, 6, pp.61782-61799, 2018.

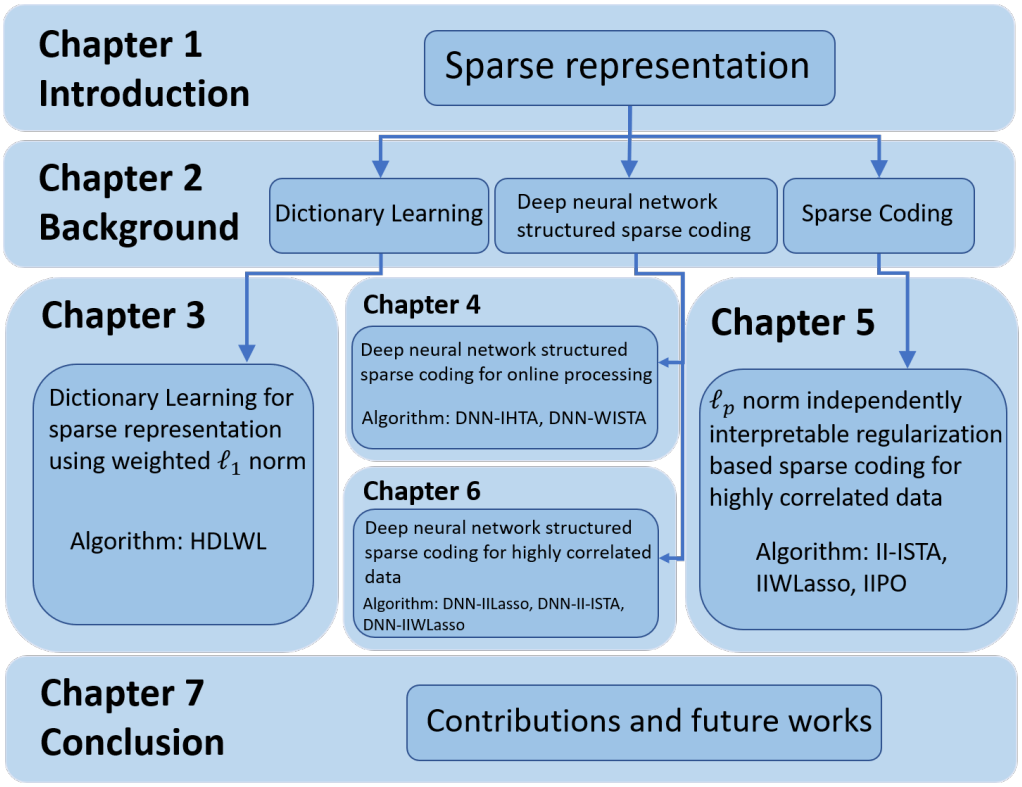


Figure 1.1: The architecture of the thesis.

[4] Yujie Li, Shuxue Ding, Benying Tan, **Haoli Zhao** and Zhenni Li, "Sparse Representation Based on the Analysis Model With Optimization on the Stiefel Manifold," *IEEE Access*, 7, pp.8385-8397, 2019.

[5] Lingjun Zhao, Huakun Huang, Xiang Li, Shuxue Ding, **Haoli Zhao** and Zhaoyang Han, "Accurate and Robust Approach of Device-Free Localization with Convolutional Autoencoder," *IEEE Internet of Things Journal*, accepted, 2019.

Conferences

[1] **Haoli Zhao**, Shuxue Ding, Yujie Li, Zhenni Li and Xiang Li, "Dictionary Learning for sparserepresentation using weighted ℓ_1 norm," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 292-296, 2016.

[2] Yujie, Li, Shuxue Ding, Benying TanXiang Li Zhenni Li and **Haoli Zhao**, "Nonnegative sparse representation based on the determinant measure," in *2016 IEEE International Conference on Digital Signal Processing (DSP)*, pp. 599-603, 2016.

Chapter 2

Background

Situated at the heart of signal and image processing, data models are fundamental for stabilizing the solution, and enabling other tasks, such as signal compression, denoising, sampling, classification and so on [50]. A model is a set of mathematical relations that the data is believed to satisfy. Models are central in signal processing. How to choose a highly simple and reliable model is an essential task. Among the many ways, the sparse-based model has been proved to be important and useful. Development of sparse representation for signals is driven by the fact that a great deal of real world signals and data can be represented by a linear combination of a few representative elements from a dictionary base in certain signal models and hence the efficiency and capacity can be enhanced during processing. Reviewing the object of sparse representation, one is required to recover a sparse estimation $\mathbf{z} \in \mathbb{R}^n$ from a lower-dimensional measured signal $\mathbf{x} \in \mathbb{R}^m$, $n > m$, in the following linear relationship,

$$\mathbf{x} = \mathbf{D}\mathbf{z} + \mathbf{v}, \tag{2.1}$$

where $\mathbf{D} \in \mathbb{R}^{m \times n}$ is a dictionary matrix, and $\mathbf{v} \in \mathbb{R}^m$ is the measurement noise. Since $\mathbf{D}$ is overcomplete, this signal reconstruction task is ill-posed with infinite solutions if there is no restriction on $\mathbf{z}$. In various applications, to find a sparse one in the infinite solutions can be a meaningful mode and can make the problem well-posed, which is the essence of sparse coding. Therefore, the optimization problem is commonly used to search for the optimal solution of the linear signal model,

$$\min_{\mathbf{z}} g(\mathbf{z}) \text{ subject to } \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 \leq \epsilon, \tag{2.2}$$

where $g(\mathbf{z})$ functions as the sparsity constraint.

Building appropriate $\mathbf{D}$ for sparse representation is a key issue in the optimization problem, which can be generally divided into two categories. One is choosing a prespecified matrix as dictionary, thus the optimization problem only concentrate on obtaining sparse and accurate coefficient $\mathbf{Z}$, namely it is a sparse coding problem. The second category considers designing a self-adaptive dictionary based on signal examples during learning procedure, which can usually result in better matching to the contents of the signals and then result in better performance in sparsity and accuracy. The methodology of building a self-adaptive dictionary is named as dictionary learning for sparse coding, which can be regard as a way to train better parameter for sparse coding.

Furthermore, there is an another methodology to train parameters for sparse coding, which is based on the similar structure between iterative shrinkage sparse coding algorithms and Recurrent Neural Network (RNN). A typical RNN receives features and subjects them to a deep structure of many layers for processing, where each layer consists of a linear transformation followed by a component-wise nonlinearity transformation. The structure of one layer in a typical RNN is similar to that of one iteration in iterative shrinkage sparse coding algorithms, which provides the possibility of using gradient-based learning methods in RNNs to train parameters in truncated sparse coding algorithms. This methodology is called Deep Neural Network structured Sparse Coding (DNN-SC) algorithm.

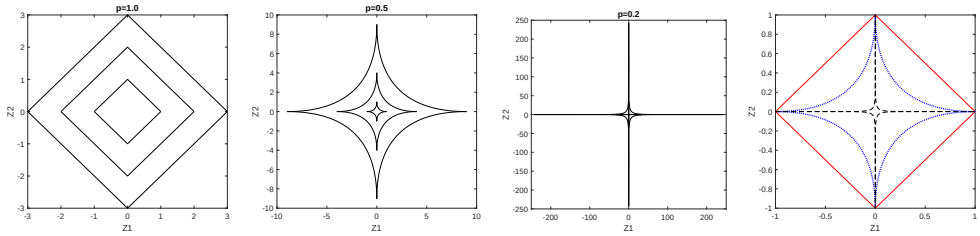
In the following sections of this chapter, we will discuss in detail about sparse coding in Section 2.1, dictionary learning in Section 2.2, and DNN-SC in Section 2.3.

2.1 Sparse Coding

Reviewing sparse coding, we tend to find a proper approach to obtain an optimal sparse solution $\mathbf{z} \in \mathbb{R}^n$ from given noisy data $\mathbf{x} \in \mathbb{R}^m$ based on the following optimization problem,

$$\min_{\mathbf{z}} L(\mathbf{z}) = \frac{1}{2}\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda g(\mathbf{z}), \tag{2.3}$$

where $\mathbf{D} \in \mathbb{R}^{m \times n}$ is an given overcomplete dictionary matrix with $n > m$, and $g(\mathbf{z})$ functions as the sparsity regularization. Effectiveness of sparse regularization can directly influence performance of sparse coding algorithms.



(a) (b) (c) (d)

Figure 2.1: (a)-(c) Contours of the constraint when $g(\mathbf{z}) = \|\mathbf{z}\|_p^p = 1, 2, 3$ with $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2]$ and different p values from left to right; (d) Contours comparison among the previous 3 cases when $g(\mathbf{z}) = 1$ with $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2]$; respectively red straight line for (a) $p = 1$, blue dot line for (b) $p = 0.5$ and black dash line for (c) $p = 0.2$.

The most commonly used sparse regularization setting is,

$$g(\mathbf{z}) = \|\mathbf{z}\|_p^p, \tag{2.4}$$

Based on the sparse regularization, the sparsest solution can be guaranteed when using the ℓ_0 norm, namely $p = 0$, that the regularization counts the number of nonzero entries in $\mathbf{z}$. However, this is a combinatorial optimization problem, where searching for an accurate sparse representation with large n is computationally prohibited. [42]. One typical alternative method is using a greedy algorithm to find the most related variables such as OMP [21, 51], which can always find sufficiently sparse solutions. Though OMP is not computationally prohibited with large data size, it still faces the problem of high computational complexity.

Another idea to solve the optimization problem relies on relaxation-based approaches. For example, the ℓ_1 norm, where $p = 1$, shown in Figure 2.1(a), is often selected as the sparsity constraint for the optimization problem, e.g., the FOCal Underdetermined System Solverthe (FOCUSS) [26] using gradient descent method by assuming the existence of a gradient factorization of $g(\mathbf{z})$, Iterative Shrinkage Thresholding Algorithm (ISTA) [22] using PO and Least Absolute Shrinkage and Selection Operator (Lasso) using CDA [23, 24]. The ℓ_1 norm is attractive because it is a convex problem and can achieve reasonable performance.

While searching for sparser and more accurate representations, the ℓ_p norm, where $p \in (0, 1)$, shown in Figure 2.1, can be an alternative choice as a relaxation approach. The contours is getting more concave with smaller p value, indicating that using ℓ_p norm regularization tends to choose fewer variables and thus enhancing sparsity. Although the ℓ_p norm $(0 < p < 1)$ makes the optimization problem nonconvex, it has proven its possibility in enhancing the sparsity and accuracy of solutions compared to the ℓ_1 norm. For example, we can solve ℓ_p norm $(0 < p <$

1) regularization by PO, e.g., the $\ell_{1/2}$ norm regularization [27, 28], $\ell_{2/3}$ norm regularization [28,29]; and we can also use weighted ℓ_1 norm to approximate the effect of ℓ_p norm $(0 < p < 1)$ [30–34].

Furthermore, we can also use different functions to define the sparsity of the coefficient matrix. By combine logarithm and the absolute value functions, we can use the log-regularizer to define the sparsity,

$$g(\mathbf{Z}) = \sum_{j=1}^{N} \sum_{i=1}^{n} \log(1 + |\mathbf{Z}_{ij}|/\delta), \tag{2.5}$$

where δ is a positive constant. To solve the nonconvex and nonsmooth optimization problem with log-regularizer, we can again use the proximal operator, e.g., the iterative log thresholding [35].

However, the above sparse regularizations in equation (2.4) mainly focus on low coherence conditions of data, and typical theoretical supports are also based on small correlation assumptions such as restricted eigenvalue condition [37, 38]. When signals are highly correlated, the coefficient also tends to be jointly correlated. Consequently, sparse coding with normal sparse regularization cannot provide efficient evidence for judging independent variable contribution and interpreting the highly correlated model. The process of interpreting models is named as "decomposability" in [39], representing for the ability of how a model can be decomposed into several parts and its components can be interpreted separately. To resolve the problem with correlated data, there are several sparse coding methods have been proposed based on the idea of selecting uncorrelated variables, and thus obtain decomposability. Uncorrelated Lasso (ULasso) [52] intends to construct a model with uncorrelated variables and form the regularization with coherence related parameter $\mathbf{R}$,

$$g(\mathbf{z}) = \|\mathbf{z}\|_1 + \mathbf{z}^{\mathrm{T}}\mathbf{R}\mathbf{z}. \tag{2.6}$$

However, the regularization still tends to select "negatively" correlated variables and hence the correlation problem is not resolved. Exclusive Group Lasso (EGLasso) [53] is proposed to solve correlated problem by defining groups with high coherence beforehand and gives the group-related regularization,

$$g(\mathbf{z}) = \|\mathbf{z}\|_1 + \sum_{k=1}^{K} \|\mathbf{z}_k\|_1^2, \tag{2.7}$$

where K is the number of groups. This regularization can be effective for correlated data when it is properly grouped, but it is necessary to group correlated variables over a determined threshold beforehand, which makes this algorithm unstable. IILasso reforms the regularization of ULasso by $\mathbf{z}$ with its absolute value,

$$g(\mathbf{z}) = \|\mathbf{z}\|_1 + |\mathbf{z}|^{\mathrm{T}}\mathbf{R}|\mathbf{z}|, \tag{2.8}$$

where $|\mathbf{z}|$ stands for using absolute value vector of $\mathbf{z}$. This change successfully makes it efficient in selecting uncorrelated variables and perform better than ULasso and EGLasso in experiments. However, IILasso still suffers the same problem as the other ℓ_1 norm based regularizations that its result is usually not sparse and accurate enough.

2.2 Dictionary Learning for Sparse Representation

Reviewing the object of dictionary learning for sparse representation, we tend to find a proper approach to obtain both an optimal overcomplete dictionary matrix $\mathbf{D} \in \mathbb{R}^{m\times n}$ with $n > m$ and corresponding sparse solution set $\mathbf{Z} \in \mathbb{R}^{n\times N}$ from given noisy data set $\mathbf{X} \in \mathbb{R}^{m\times N}$ based on the following optimization problem,

$$\min_{\mathbf{D},\mathbf{Z}} L(\mathbf{D}, \mathbf{Z}) = \frac{1}{2}\|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2 + \lambda g(\mathbf{Z}), \tag{2.9}$$

where $g(\mathbf{Z})$ functions as the sparsity regularization.

The most commonly used methodology for solving dictionary learning problem is usually separated into two stages:

1. Sparse coding stage: assuming that the dictionary $\mathbf{D}$ is known, find the proper sparse representation set $\mathbf{Z}$ by using the data set $\mathbf{X}$;

2. Dictionary learning stage: assuming that the proper sparse representation set $\mathbf{Z}$ is known, use the data set $\mathbf{X}$ to find the most satisfied dictionary $\mathbf{D}$.

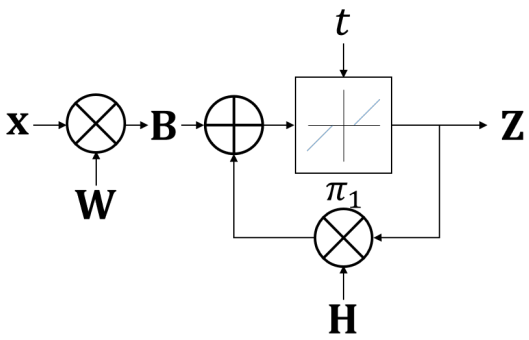
In the sparse coding stage, we use updated dictionary from previous iteration to update sparse representation set $\mathbf{Z}$ with sparse coding algorithms mentioned in the previous section. While in the dictionary learning stage, the optimization problem is,

$$\min_{\mathbf{D}} L(\mathbf{D}) = \frac{1}{2}\|\mathbf{X} - \mathbf{D}\mathbf{Z}^{(k-1)})\|_F^2, \tag{2.10}$$

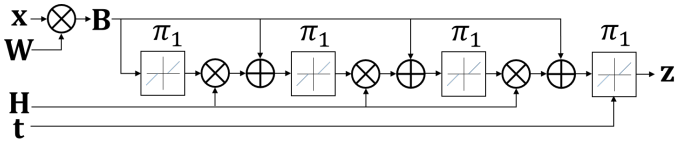
where k stands for iteration number. Since the sub-optimization problem for dictionary learning is convex, it is possible to directly use gradient descend method to update dictionary, which is employed in FOCUSS-CNDL [26] with FOCUSS in its sparse coding stage. Moreover, another classical dictionary learning algorithm is the Method of Optimal Directions (MOD) [45], which updates the overall set of atoms simultaneously by optimizing a least squares problem, but facing the problem that its convergence could not be guaranteed. Furthermore, singular value decomposition (SVD) is also a possible route for updating dictionary, which is employed in K-SVD [43, 44]. K-SVD updates columns of dictionary one by one using corresponding largest singular vector, which can usually learn a good dictionary and improve the convergence. However, SVD is computationally expensive in high dimension data. What is more, by introducing a coherence related term in the dictionary learning stage, PDLA use the proximal operator to update dictionary columns based on the assumption that dictionary columns are not correlated [36].

2.3 Deep Neural Network structured Sparse Representation

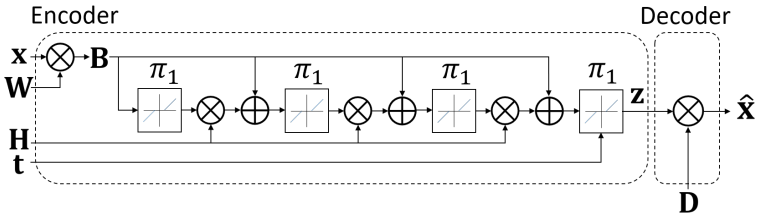
DNN-structured sparse representation algorithms are based on the idea of unfolding an iterative algorithm with shared parameters through specific layers based on the processing similarity between the iterative algorithm and the Recurrent Neural Network (RNN), which was developed by Domke and applied to the tree-reweighted belief propagation and mean-field inference [54, 55]. Gregor and LeCun were the first to implement this idea in the sparse coding algorithm [46]. They unfolded the structure of ISTA, as shown in Figure. 2.2(a), to form a feed-forward neural network named Learned ISTA (LISTA), which is illustrated in Figure. 2.2(b). LISTA is a supervised learned neural network that requires inputting a dataset of signals and corresponding sparse representations (as labels for training) pairs to a truncated unfolded ISTA structure to train the weights and bias. The learned DNN from LISTA has proven its efficiency in estimating spare representations of other signals besides the ones used in training, which can reach the converged result standard of ISTA with much fewer layers than the number of ISTA convergent iterations. Sprechmann et al. proposed the unsupervised Trained ISTA (TISTA) [47]. TISTA has a similar neural network structure to LISTA because both of them form from a truncated unfolded ISTA structure, as shown in Figure. 2.2(c), but TISTA changes the learning procedure by adding a decoder at the end of the network. By requiring the output as close as



(a)



(b)



(c)

Figure 2.2: (a) Illustration of the ISTA algorithm for sparse coding. The optimal sparse representation can be obtained by the recursive structure $\mathbf{z}^{(k)} = \pi_1(\mathbf{W}\mathbf{x} + \mathbf{H}\mathbf{z}^{(k-1)}, t)$, where $\mathbf{x}$ is the input signal, $\pi_1(\mathbf{x}, t)$ is the soft thresholding function with threshold t, $\mathbf{W} = \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}$, $\mathbf{H} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}\mathbf{D}$, and α is a restriction parameter for ISTA. (b) Network structure of the supervised learned DNN-ISTA, which is named LISTA, formed from unfolded ISTA and truncated to a fixed number of iterations (3 here). $\mathbf{W}$, $\mathbf{H}$, $\mathbf{t}$ are trainable parameters in the network to give an approximate sparse representation on a given dataset. (c) Network structure of the unsupervised learned DNN-ISTA, which is named TISTA. TISTA has a similar propagation structure to LISTA, and $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{t}$ are targeted trainable parameters. The key difference is that TISTA uses a decoder to output $\mathbf{x}$ as the learning objective, where the original $\mathbf{x}$ is the known input. On the contrary, original $\mathbf{z}$, which is required for supervised learning, is a priori knowledge.

possible to the input, it becomes possible to directly learn the weights and bias from the loss function of the original sparse representation problem. Consequently, TISTA avoids using the true sparse representations for a separated training and enables the online network learning and processing procedure simultaneously. The unfolded ISTA also shows that DNN-SC can help learning the optimal nonlinear threshold functions for iterative sparse coding to achieve better performances in known datasets [56, 57]. Kamilov and Mansour [56] proposed to learn the nonlinear activation function, which is modeled using cubic B-splines through DNN-structured ISTA. The learned nonlinear threshold can result in better accuracy than ISTA. Mahapatra et al. [57] proposed a more parsimonious representation of the thresholding function using a linear expansion of thresholds during learning. Furthermore, the DNN can be applied to other iterative spare coding algorithms such as Iterative Hard Thresholding (IHT) [48] and Approximate Message Passing (AMP) [49]. Both IHT and AMP have the similar structure as ISTA: all functions in these sparse coding algorithms are continuous and overall differentiable throughout; thus, they can be unfolded, and the parameters in their structures can be learned as a DNN. Moreover, Moreau and Bruna presented mathematical explanations about the acceleration of DNN-structured algorithms by analyzing the specific matrix factorization in the Gram kernel of dictionaries [58]. The findings show that the learning procedure in DNN-SC attempts to diagonalize the kernel with a basis, which produces a small perturbation of the original ℓ_1 space, and the learning may fail if there is no factorization. Efforts are also made to combine sparse coding algorithms with different DNN structures. The convolutional neural networks (CNN), which have proven its superiority in image processing tasks, have been widely combined with sparse coding algorithms for image classification [59–61] and image restoration [62–64]. These deep learning architectures are significantly different from DNN-SC since they are not truncated iterative SC algorithms but using sparse coding structure to reconstruct signal for specific tasks. A notable difference is that the output signal is, e.g., class labels in the general deep learning architectures. In this paper, the output signal is $\mathbf{z}$, since the purpose is to present signal $\mathbf{x}$ with a sparse $\mathbf{z}$.

Chapter 3

Dictionary Learning for Sparse Representation using Weighted ℓ_1 Norm

3.1 Introduction

Dictionary learning is a classical methodology to enhance performance of sparse coding. Based on the signal model (2.1), we tend to use few variables in dictionary to represent signals, thus an appropriate dictionary which fit data well can directly help improve performance of sparse coding. In this chapter, we propose to employ the ℓ_p norm ($0 < p < 1$) as sparsity constraint in the dictionary learning problem. To solve the problem brought by the nonconvex property of the the ℓ_p norm ($0 < p < 1$), we introduce the weighted ℓ_1 norm to convexly approximate ℓ_p norm ($0 < p < 1$) and combined with the hierarchically alternating update strategy [65] which is an idea firstly applied in dictionary learning for nonnegative signals with the other sparsity constraints. To further solve the problem that the absolute value function is not derivable at zero, we propose to use logarithm and hyperbolic function to approximate absolute value function. We present an efficient algorithm for learning dictionary with the weighted ℓ_1 norm as sparsity constraint, including two alternating phases: sparse coding and dictionary update. This algorithm presents good robustness to noise when signal-to-noise ratio (SNR) is higher than 10 dB that dictionaries can be recovered to nearly 100%.

The remainder of this chapter is organized as follows. The formulation of our algorithm is presented in Section 3.2. This section describes details of the targeted minimization problem, the

weighted ℓ_1 norm and formation of the hierarchically alternating update strategy. In section 3.3, the idea and structure of the algorithm are shown. Procedure of the algorithm throughout one iteration is presented. Then we present results of numerical experiments conducted in section 3.4 to evaluate the algorithms ability of recovering dictionary and finding sparse representation, then comparison with the other classical algorithms are shown to present the potential advantage of this algorithm. Finally, chapter summary is drawn in Section 3.5.

3.2 Problem formulation

The algorithm we present in this chapter is for learning a proper dictionary, which aims at representing data sparsely. The algorithm is based on the signal model as follow,

$$\mathbf{X} = \mathbf{DZ} + \mathbf{V}, \tag{3.1}$$

where $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N) \in \mathbb{R}^{m \times N}$ is the given data set. During procedure of the algorithm, we try to approach to a dictionary $\mathbf{D} \in \mathbb{R}^{m \times n}$ which can perform the best possible sparse representation based on the given data and the chosen sparsity constraint in this study. Those corresponding sparse representation set is labeled as $\mathbf{Z} = (\mathbf{z}_1, ..., \mathbf{z}_N) \in \mathbb{R}^{n \times N}$ and $\mathbf{V} = (\mathbf{v}_1, ..., \mathbf{v}_N) \in \mathbb{R}^{m \times N}$ is corresponding Gaussian distributed noise. The problem is reconstructed as the following problem using maximum a posteriori estimation [26],

$$(\mathbf{D}_{\text{MAP}}, \mathbf{Z}_{\text{MAP}}) = \arg\max_{\mathbf{D},\mathbf{Z}} P(\mathbf{D}, \mathbf{Z}|\mathbf{X}) = \arg\min_{\mathbf{D},\mathbf{Z}} L(\mathbf{D}, \mathbf{Z}), \tag{3.2}$$

$$L(\mathbf{D}, \mathbf{Z}) = \frac{1}{2}\|\mathbf{X} - \mathbf{DZ}\|_F^2 + \lambda d_p(\mathbf{Z}), \tag{3.3}$$

where $\lambda > 0$ is a tuning parameter to adjust the effect of the sparse constraint $d_p(\mathbf{z})$, which is a ℓ_p norm $(0 < p < 1)$ regularization formulated as follow,

$$d_p(\mathbf{z}) = \|\mathbf{z}\|_p^p = \sum_{i=1}^{n} |\mathbf{z}_i|^p. \tag{3.4}$$

We choose ℓ_p norm $(0 < p < 1)$ regularization because of its property in pursuing good sparsity and accuracy, but this choice makes the optimization problem a non-convex one, which means the problem may have multiple local minimum. Thus, we propose to use weighted ℓ_1

norm to approximate the ℓ_p norm $(0 < p < 1)$ regularization, which can be reformed as follows,

$$d_p(\mathbf{z}) = \|\mathbf{z}\|_p^p \approx \sum_i^n |\mathbf{z}_i^{(k-1)}|^{p-1}|\mathbf{z}_i|, \tag{3.5}$$

where k represents the number of iterations. Benefited from the approximation, the regularization becomes convex, while the absolute value of $\mathbf{z}_i$ is still a barrier to calculate the loss function directly because it is not derivable at zero. To solve this problem, we proposed to introduce a smoothed approximation for the absolute value,

$$|z| = \frac{1}{c}\log\cosh(cz), \tag{3.6}$$

where higher value of the constant c makes the approximation more similar to the absolute value function. With this approximation, the sparsity constraint function is derivable in any points.

For the reason of finding results more accurately and efficiently, the hierarchically alternating update strategy is employed. This procedure is detailed as column-wisely updated dictionary $\mathbf{D}$; correspondingly, the sparse representation set $\mathbf{Z}$ is updated row-wisely. Thereby the problem should be reconstructed as below,

$$\begin{aligned}\min_{\mathbf{D}_{:i},\mathbf{Z}_{i:}} L(\mathbf{D}_{:i},\mathbf{Z}_{i:}) &= \frac{1}{2}\|(\mathbf{X} - \sum_{j=1,j\neq i}^{n}\mathbf{D}_{:j}\mathbf{Z}_{j:}) - \mathbf{D}_{:i}\mathbf{Z}_{i:}\|_F^2 + \lambda d_p(\mathbf{Z}_{i:}) \\ &= \frac{1}{2}\|\hat{\mathbf{X}}_i - \mathbf{D}_{:i}\mathbf{Z}_{i:}\|_2^2 + \lambda d_p(\mathbf{Z}_{i:}),\end{aligned} \tag{3.7}$$

where $\mathbf{D}_{:i}$ is ith column of the dictionary $\mathbf{D}$ and $\mathbf{Z}_{i:}$ is ith row of the coefficient matrix $\mathbf{Z}$.

3.3 Algorithm

This iterative dictionary learning algorithm is inspired by MOD, in which the learning procedure is separated into two independent stages, sparse coding phase for sparse vector selection and dictionary update phase for dictionary learning. This can be summarized as:

1. Sparse coding phase: assume that the dictionary $\mathbf{D}$ is known and use the data set $\mathbf{X}$, to find the proper sparse representation set $\mathbf{Z}$;

2. Dictionary learning phase: assume that the proper sparse representation set $\mathbf{Z}$ is known and use the data set $\mathbf{X}$ to find the most satisfied dictionary $\mathbf{D}$.

The difference is that we update $\mathbf{D}$ column-wisely and $\mathbf{Z}$ row-wisely, namely the hierarchically alternating update strategy, rather than update them entirely as in MOD. In the proposed algorithm, the updating at each iteration uses the local optimal. During sparse coding stage, the dictionary $\mathbf{D}$ is supposed to be known, so the derivative of every entry of $\mathbf{Z}$ based on the loss function $L(\mathbf{D}_{:i}, \mathbf{Z}_{i:})$ is,

$$\partial_{\mathbf{Z}_{i:}} L(\mathbf{Z}_{i:}) = \sum_{j=1}^{N} ((\mathbf{D}_{:i}\mathbf{Z}_{ij} - \hat{\mathbf{X}}_i(:,j))^{\mathrm{T}}\mathbf{D}_{:i} + \lambda \partial_{\mathbf{Z}_{ij}} d_p(\mathbf{Z}_{ij})), \tag{3.8}$$

where $\mathbf{Z}_{ij}$ locates at i row and j column in sparse solution set $\mathbf{Z}$. To obtain the minimum of $L(\mathbf{Z}_{ij})$, the equation below which represents for $\partial_{\mathbf{Z}_{ij}} L(\mathbf{Z}_{ij}) = 0$ should be solved.

$$f(\mathbf{Z}_{ij}) = \mathbf{D}_{:i}^{\mathrm{T}}\mathbf{D}_{:i}\mathbf{Z}_{ij} - \mathbf{D}_{:i}^{\mathrm{T}}\hat{\mathbf{X}}_i(:,j) + \lambda |\mathbf{Z}_{ij}^{(k-1)}|^{p-1} \tanh(c\mathbf{Z}_{ij}) = 0, \tag{3.9}$$

where the solution represents for the local minimum of the loss function. However, it is not easy to directly obtain the solution from the combination of a hyperbolic function and a line function, thus we use the Newton method to obtain the approximate solution by repeating,

$$\mathbf{Z}_{ij} \leftarrow \mathbf{Z}_{ij} - \frac{f(\mathbf{Z}_{ij})}{f'(\mathbf{Z}_{ij})}, \tag{3.10}$$

where $f'(\mathbf{Z}_{ij})$ is,

$$f'(\mathbf{Z}_{ij}) = \mathbf{D}_{:i}^{\mathrm{T}}\mathbf{D}_{:i} + c\lambda |\mathbf{Z}_{ij}^{(k-1)}|^{p-1}(1 - \tanh^2(c\mathbf{Z}_{ij})), \tag{3.11}$$

In the dictionary update phase, the sparse representation set $\mathbf{Z}$ is assumed known, thus updating $\mathbf{D}$ is only affected by the ℓ_2 norm error and the derivative of the loss function is,

$$\partial_{\mathbf{D}_{:i}} L(\mathbf{D}_{:i}) = (\mathbf{D}_{:i}\mathbf{Z}_{i:} - \hat{\mathbf{X}}_i)^{\mathrm{T}}\mathbf{D}_{:i}. \tag{3.12}$$

Therefore, the update rule of $\mathbf{D}_{:i}$ can be directly obtained by calculating $\partial_{\mathbf{D}_{:i}} L(\mathbf{D}_{:i}) = 0$,

$$\mathbf{D}_{:i} \leftarrow \frac{\hat{\mathbf{X}}_i \mathbf{Z}_{i:}^{\mathrm{T}}}{\mathbf{Z}_{i:}\mathbf{Z}_{i:}^{\mathrm{T}}}. \tag{3.13}$$

Furthermore, it should be aware that updating dictionary $\mathbf{D}$ should be bounded in a certain

region to avoid bad scale of its value and enforce the algorithm to converge to a local minimum. In this algorithm, we use column normalization to restrict values in $\mathbf{D}$ as follows,

$$\mathbf{D} \in \{\mathbf{D} | \|\mathbf{D}_{:i}\|_2^2 = 1, i = 1, ..., n, \mathbf{D} \in \mathbb{R}^{m \times n}\}. \tag{3.14}$$

Algorithm 3.1: HDLWL

Input: data set $\mathbf{X}$, proper positive parameters λ and c.
Initialization: initialize $\mathbf{D}^{(0)}$, $\mathbf{Z}^{(0)}$, $k = 0$.
Main iteration: increment k by 1
• *Spare representation phase*
use the Newton method to solve the following equation by repeating equation (3.10) to converge.
$\mathbf{D}_{:i}^{\mathrm{T}}\mathbf{D}_{:i}\mathbf{Z}_{i:} - \mathbf{D}_{:i}^{\mathrm{T}}\hat{\mathbf{X}}_i + \lambda|\mathbf{Z}_{i:}^{(k-1)}|^{p-1}\tanh(c\mathbf{Z}_{i:}) = 0, i = 1, ..., n$
• *Dictionary learning phase*
$\mathbf{D}_{:i} \leftarrow \frac{\hat{\mathbf{X}}_i \mathbf{Z}_{i:}^{\mathrm{T}}}{\mathbf{Z}_{i:}\mathbf{Z}_{i:}^{\mathrm{T}}}, i = 1, ..., n$
Stopping rule: stop if $\mathbf{D}^{(k)}$ and $\mathbf{Z}^{(k)}$ have converged
Output: $\mathbf{D} = \mathbf{D}^{(k)}$ and $\mathbf{Z} = \mathbf{Z}^{(k)}$

In summary, the algorithm proposed in this chapter is present in specific steps principally in Algorithm 3.1, named as Hierarchical Dictionary Learning with Weighted ℓ_1 norm (HDLWL).

3.4 Numerical Experiments

In this section, we present the results of the experiments for evaluating the proposed algorithms. From the experiments we tested whether this algorithm can recover the grand true dictionary. All experiments were performed via Matlab R2018b, and programs were run on a PC with a 2.7 GHz Intel core and 12G RAM.

In the synthetic data simulation experiments, we first formed the dictionary $\mathbf{D}_{\text{orig}}$, which is sized as a 20×50 matrix, generated by randomly drawing value from a normal distribution $N(0, 1)$ and finally column-normalized. In every sparse representation vector, non-zero values were set in 3 random positions. There were 1000 ground-truth generated sparse representation vectors for learning dictionary $\mathbf{Z}_{\text{orig}} = (\mathbf{z}_{\text{orig}1}, ..., \mathbf{z}_{\text{orig}1000})$ in the experiments. Correspondingly, data set $\mathbf{X}_{\text{orig}}$ had 1000 samples, which were generated by $\mathbf{D}_{\text{orig}}$ and $\mathbf{Z}_{\text{orig}}$ based on equation (3.1). The noise set $\mathbf{V}$ were added to generate input data set $\mathbf{X}$ based on Gaussian random entries with various SNR levels. To evaluate the anti-noise performance and robustness of our proposed algorithm, experiments were conduct in 3 noise situations with SNR level of 10

dB, 20 dB and no noise. For different SNR level, 10 trials were conducted during experiments.

To finish the initialization before the algorithm started, the first n data vectors were used to form the initial dictionary $\mathbf{D}_{init}$,

$$\mathbf{D}_{init} = [\mathbf{X}_{:1}, ..., \mathbf{X}_{:n}], \{\|\mathbf{D}_{init}(:, i)\|_2^2 = 1, i = 1, ..., n\}. \tag{3.15}$$

The sparse representation set $\mathbf{Z}_{init}$ was initialized by the pseudo inverse of initial dictionary $\mathbf{D}_{init}$ and data set $\mathbf{X}$,

$$\mathbf{Z}_{init} = \frac{\mathbf{D}_{init}^{\mathrm{T}}}{\mathbf{D}_{init}\mathbf{D}_{init}^{\mathrm{T}}}\mathbf{X}. \tag{3.16}$$

For measuring performances of learnt dictionary in the synthetic data experiments, we used recovery ratio and mean dictionary distance for evaluation by comparing with the ground true dictionary $\mathbf{D}_{\text{orig}}$. The ith column of learnt dictionary $\mathbf{D}(:, i)$ was counted as a recovered one if it can find a close match to one column in the original dictionary, which means,

$$\{1 - \frac{|\mathbf{D}(:, i)^{\mathrm{T}}\mathbf{D}_{\text{orig}}(:, j)|}{\|\mathbf{D}(:, i)\|\|\mathbf{D}_{\text{orig}}(:, j)\|} \leq 0.01, j = 1, ..., n\}, \tag{3.17}$$

and each column in $\mathbf{D}_{\text{orig}}$ will not be count again after finding a matched column in $\mathbf{D}$. When all columns in learnt dictionary $\mathbf{D}$ are paired with the original dictionary $\mathbf{D}_{\text{orig}}$, we can calculate dictionary distances between matched pairs,

$$\text{Mean dictionary distance} = \frac{1}{n}\sum_{i=1}^{n}(\{1 - \frac{|\mathbf{D}(:, i)^{\mathrm{T}}\mathbf{D}_{\text{orig}}(:, \mathbf{l}(i))|}{\|\mathbf{D}(:, i)\|\|\mathbf{D}_{\text{orig}}(:, \mathbf{l}(i))\|}), \tag{3.18}$$

where $\mathbf{l}$ is label set of paired columns in $\mathbf{D}_{\text{orig}}$ for corresponding columns in $\mathbf{D}$.

Sparsity measurement used Hoyer sparsity measure [66] based on the relationship between the ℓ_1-norm and the ℓ_2-norm, which can give a well defined sparsity. The Hoyer sparsity measure is formulated as following,

$$\text{Hoyer sparsity}(\mathbf{z}) = \frac{\sqrt{n} - (\sum |\mathbf{z}_i|)/\sqrt{\sum {\mathbf{z}_i}^2}}{\sqrt{n} - 1}, \tag{3.19}$$

where n represents the dimension of $\mathbf{z}$, and when the value of the equation is closer to 1, the $\mathbf{z}$ vector is approaching to a sparser vector.

Figure 3.1 shows performance of proposed algorithm HDLWL in a range of λ with different p settings for 20 dB SNR data. Except $p = 0.9$, the other p settings can all recover the dic-

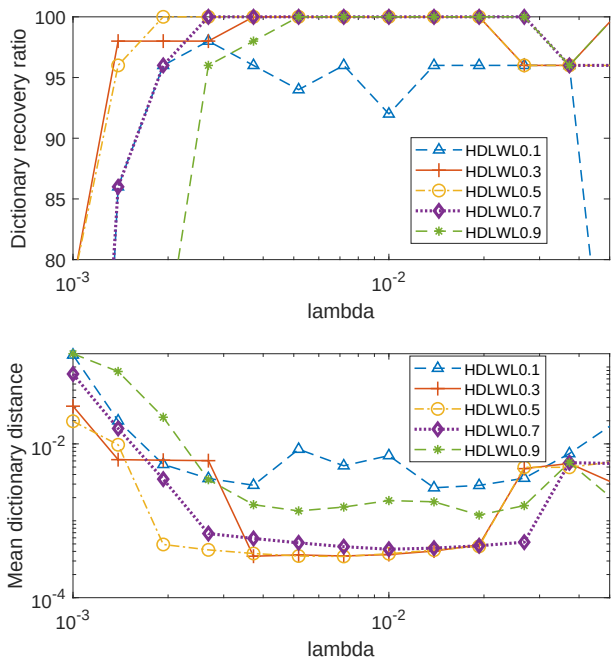


Figure 3.1: Dictionary recovery ratio and mean dictionary distance of proposed algorithm with different p values in a range of λ from data with 20 dB SNR

tionary to 100% in a range of varying lambda with good mean dictionary distances. In detail, HDLWL0.3 and HDLWL0.5 can obtain relatively the best mean dictionary distances with 100% dictionary recovery ratio. We further present performance of proposed algorithm HDLWL for 10 dB SNR data in Figure. 3.2. We can see that only HDLWL0.5 obtain 100% dictionary recovery ratio in this case with 10 dB. We also notice obvious narrowing in the lambda range for obtaining 100% dictionary recovery ratio compared with performances in 20 dB. A more intuitive comparison between reordered learnt dictionary and the ground true original dictionary is shown in Figure 3.3. We can see the learnt dictionary can fit the ground true dictionary for every atom.

Dictionary recovery ratio, mean dictionary distance and Hoyer sparsity convergence graph of proposed algorithm with different p value is shown in Figure 3.4. In general, HDLWL with these p settings can converge at 100% dictionary recovery ratio in 20dB with small mean dictionary distance. Moreover, HDLWL with these p settings can also converge at the Hoyer sparsity near to the ground true one. In detail, we can see slight differences in convergence speed among different p settings that reducing p tend to increase iteration numbers before convergence. Furthermore, smaller p values also result in closer converged Hoyer sparsity to the original one.

Based on previous experimental results, we choose $p = 0.5$ for HDLWL to compare with

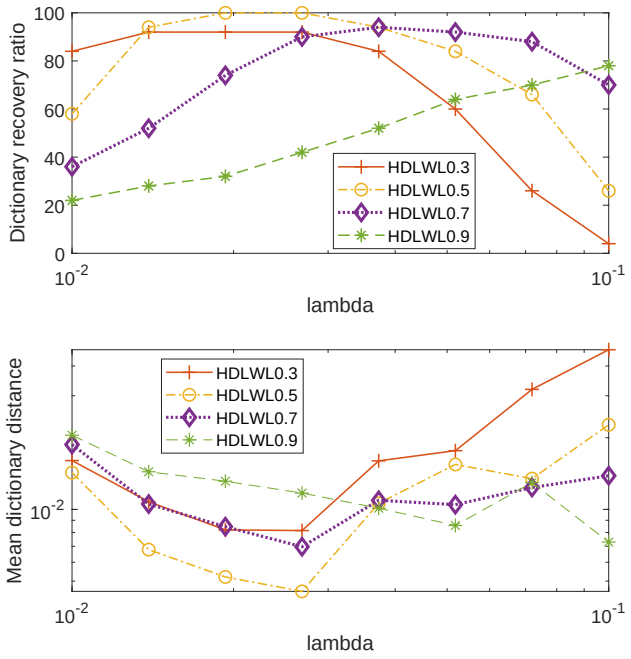
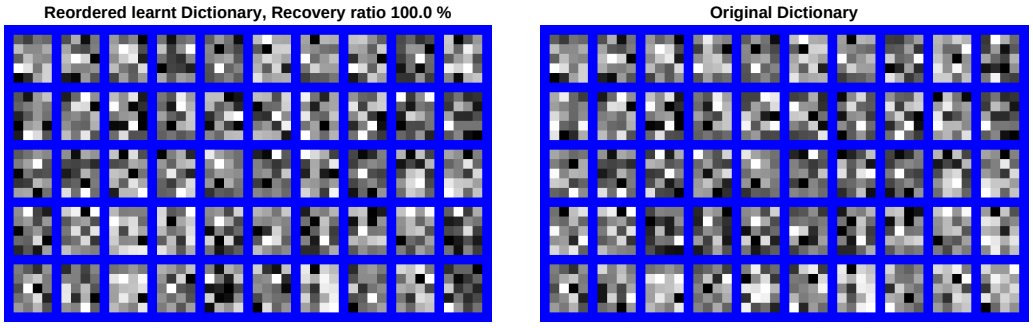


Figure 3.2: Dictionary recovery ratio and mean dictionary distance of proposed algorithm with different p values in a range of λ from data with 10 dB SNR



(a) (b)

Figure 3.3: Reordered learnt Dictionary (a) from 20dB signals compared with the ground true dictionary (b), present in 4×5 dimensional subspaces.

the other classical dictionary learning algorithms. Dictionary recovery ratio and mean dictionary distance convergence graph of different algorithms in time scale is shown in Figure 3.5. In general, all algorithms can converge in reasonable time scale. We can see that HDLWL0.5 can converge at higher dictionary recovery ratio with similar computation time compared with KSVD, MOD and PDLA. Compared with FOCUSSDL, HDLWL0.5 can converge faster with smaller mean dictionary distance.

Detail performance comparison between different algorithm in different noise condition for 10 trails is shown in Table 3.4. In general, recovery ratio and mean dictionary distance of all

algorithms tend to worsen with higher noise with one exception that performances of K-SVD in no noise and 20dB condition are similar. In detail, HDLWL0.5 can obtain best recovery ratios and mean dictionary distances in different noise conditions among all algorithms. Moreover, convergence time of HDLWL0.5 is reasonable good that it is similar with KSVD, MOD and PDLA when noise level is higher than 20dB and far smaller than FOCUSSDL in all noise condition. The reason that convergence time for HDLWL0.5 and FOCUSSDL in noiseless condition is longer than the other cases with noise is that optimized λ for noiseless condition is usually much smaller than the others and smaller λ tend to have higher convergence iteration numbers.

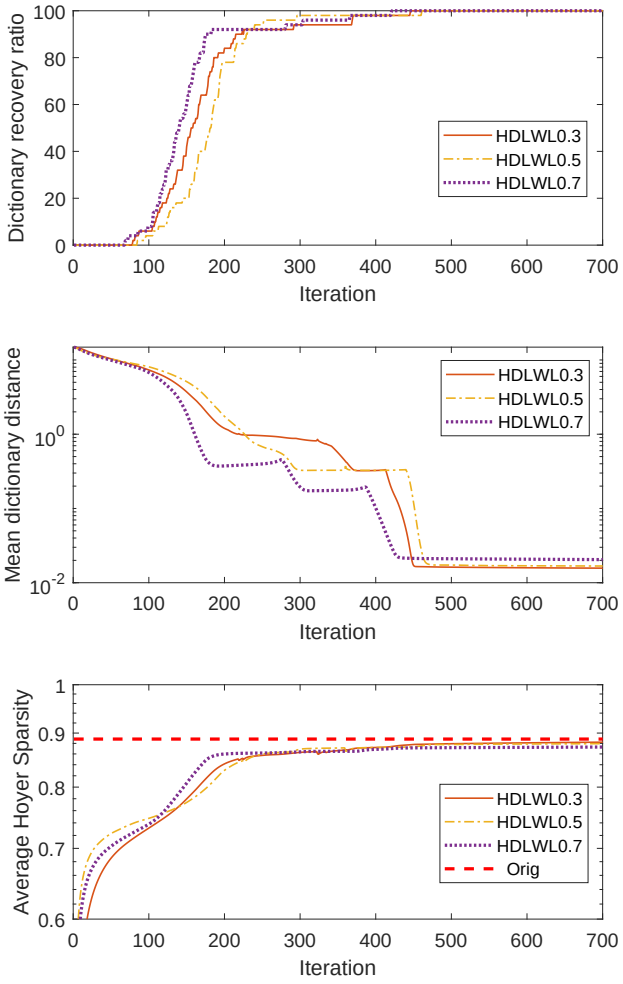


Figure 3.4: Dictionary recovery ratio, mean dictionary distance and Hoyer sparsity convergence graph of proposed algorithm with different p values in iterations with optimized parameters from data with 20 dB SNR

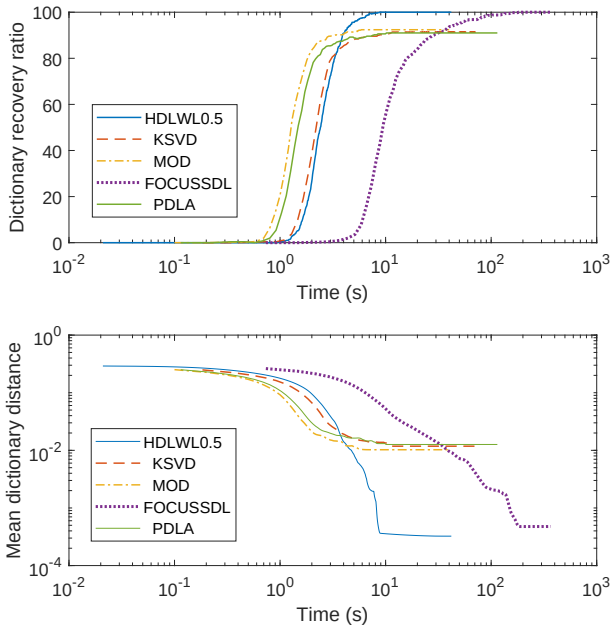


Figure 3.5: Average dictionary recovery ratio and mean dictionary distance convergence graph of different algorithms in time scale with optimized parameters from data with 20 dB SNR

Table 3.1: Performance of different algorithms in various noise condition for 10 trails

Algorithms	Recovery ratio (%)			Mean dictionary distance			Convergence time (s)		
	no noise	20 dB	10 dB	no noise	20 dB	10 dB	no noise	20 dB	10 dB
HDLWL0.5	100.0 ± 0.0	100.0± 0.0	95.8 ± 4.7	2.16 ± 0.63 $\times 10^{-6}$	3.24 ± 0.29 $\times 10^{-4}$	7.15 ± 2.42 $\times 10^{-3}$	26.7 ± 8.4	9.5 ± 2.7	14.2 ± 4.0
K-SVD	90.2 ± 3.8	91.4 ± 3.8	79.8 ± 8.08	1.30 ± 0.48 $\times 10^{-2}$	1.18 ± 0.54 $\times 10^{-2}$	2.04 ± 0.58 $\times 10^{-2}$	8.8 ± 2.0	8.9 ± 2.7	16.0 ± 3.4
MOD	92.0 ± 4.2	92.4 ± 3.4	79.6 ± 9.7	1.10 ± 0.59 $\times 10^{-2}$	1.02 ± 0.44 $\times 10^{-2}$	1.70 ± 0.59 $\times 10^{-2}$	5.2 ± 1.2	5.4 ± 1.4	8.1 ± 1.9
FOCUSSDL	100.0 ± 0.0	100.0± 0.0	85.8 ± 4.0	3.97 ± 1.61 $\times 10^{-5}$	4.76 ± 0.49 $\times 10^{-4}$	1.59 ± 0.35 $\times 10^{-2}$	157.5 ± 31.2	103.9± 44.7	119.7± 36.0
PDLA	96.2 ± 4.2	92.2 ± 6.3	72.4 ± 7.3	5.29 ± 6.10 $\times 10^{-3}$	1.26 ± 0.77 $\times 10^{-2}$	2.70 ± 0.65 $\times 10^{-2}$	5.02 ± 2.48	5.79 ± 2.66	7.84 ± 1.95

3.5 Chapter Summary

In conclusion of this chapter, we present a dictionary learning algorithm with the ℓ_p norm $(0 < p < 1)$ as sparsity constraint in this paper. To solve the complex nonconvex optimization problem caused by the ℓ_p norm, we have successfully applied the combination of hierarchically alternating update strategy and weighted ℓ_1 norm method. The algorithm was validated to be effective in synthetic data, for training an overcomplete dictionary which suits a set of given signals by numerical experiments. Three contributions can be summarized here. Firstly,

the algorithm proposed in this paper can recover dictionary to nearly 100% in finite iterations for data having a SNR higher than 10 dB. Secondly, the weighted ℓ_1 norm is valid to reduce sparseness when reduce value of p. Furthermore, the good robustness of this algorithm has been confirmed in numerical experiments in comparison with the other three conventional dictionary learning algorithms. In the furture, we will employ the proposed algorithm for image denoising, inpainting and other applications.

Chapter 4

Deep Neural Network Structured Sparse Coding for Online Processing

4.1 Introduction

Sparse coding has proven its effectiveness in various signal processing applicaions, such as image denoising [43, 44, 67], inpainting [42, 68], super-resolution [69, 70], etc.. However, the efficiency and capacity of the current sparse coding algorithms have not reached the requirement of certain missions that require fast processing, such as real-time video denoising; thus, we propose our sparse coding algorithms, which use the DNN structure and corresponding learning procedure in sparse coding to achieve the goal.

Deep neural network structured sparse coding (DNN-SC) is a methodology to train parameters for enhancing efficiency of sparse coding, which have been developed in recent years. Building DNN-SC bases on the structure similarity between iterative shrinkage sparse coding algorithms and the Recurrent Neural Network (RNN). In 2010, Gregor and LeCun introduced the idea with Iterative Shrinkage Thresholding Algorithm (ISTA), where the learned DNN can perform nearly 10 times faster than the original sparse coding algorithm, i.e., ISTA [46]. However, its learning procedures require one to know the true answers of the sparse representation. In 2015, Sprechmann et al. showed that it is possible to learn the DNN without requiring the true answers to serve as the training labels, i.e., the unsupervised learning is possible [47]. With unsupervised network learning, the DNN-SC algorithms can be more suitable for online processing without prior training when the network can be efficiently and quickly learned online to make the processing enjoy the acceleration introduced by the DNN structure.

In this chapter, we propose to build ℓ_p norm ($0 < p < 1$) regularization based DNN-SCs to obtain better sparsity and accuracy compared to existed ℓ_1 norm regularization based DNN-SCs. The remainder of this chapter is organized as follows. In section 4.2, we first describe problem formulation and original sparse coding algorithms. In section 4.3, we show how to unfold IHTA and WISTA to form a feed-forward neural networks, where the parameters can be learned by back-propagation. Both unfolded WISTA and IHTA have similar structures to ISTA and the recurrent neural network, where each layer is comprised by a differentiable combination of linear and nonlinear operators. For the setting of the loss function, all DNN-SC algorithms can be learned through supervised and unsupervised schemes (section 4.3.1). The unsupervised learning procedure is the key to apply DNN-SC in online learning processing.

Subsequently, we give an experimental validation of proposed algorithms in section 4.4. Synthetic data experiments (section 4.4.1) present a performance comparison in terms of the relative norm error and accuracy among the sparse coding algorithms ISTA, IHTA and WISTA and their DNN-structured versions.

Real-world graphic experiments are shown in section 4.4.2. Both image and video denoising experiments concentrate on the acceleration of the DNN-SC algorithm in denoising while maintaining reasonably good performances. Benefiting from the acceleration of the DNN structured processing, we show that DNN-SC algorithms can conduct real-time video denoising with only CPU for 25 frames per second (FPS) 360×480-pixel gray-scaled videos.

4.2 Sparse Coding

The paper considers a sparse representation problem, where we tend to find a proper approach to obtain an optimal sparse solution $\mathbf{z} \in \mathbb{R}^n$ from given noisy data $\mathbf{x} \in \mathbb{R}^m$ based on the linear signal model described in the following equation:

$$\mathbf{x} = \mathbf{D}\mathbf{z} + \mathbf{v}, \tag{4.1}$$

where $\mathbf{D} \in \mathbb{R}^{m \times n}$ is an overcomplete dictionary matrix with $n > m$, and $\mathbf{v} \in \mathbb{R}^m$ is an additive white Gaussian distributed noise vector. The above equation (4.1) defines an underdetermined linear system. Because the dictionary in the equation is supposed to be a full row-rank matrix, this model should have infinite solutions. To achieve the required sparse answer, the sparse constraint is introduced. Therefore, the general minimization model with the square data fitting

error and a sparse constraint is applied to solve the linear signal model,

$$\min_{\mathbf{z}} f(\mathbf{z}) = \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda d_p(\mathbf{z}), \tag{4.2}$$

where $\lambda > 0$ is a tuning parameter to adjust the effect of the sparse constraint; the function $d_p(\mathbf{z})$ is a sparse penalty term formulated as follows,

$$d_p(\mathbf{z}) = \|\mathbf{z}\|_p^p = \sum_{j=1}^{n} |z_j|^p, \tag{4.3}$$

4.2.1 ISTA

Algorithm 4.1: ISTA

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper parameters λ and α.
Restriction: $\alpha >$ largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$
Initialization: $t = \frac{\lambda}{\alpha}$, $\mathbf{z}^{(0)} = \mathbf{0}$, $k = 0$.
Main iteration: increment k by 1
$\mathbf{z}^{(k)} = \pi_1(\mathbf{z}^{(k-1)} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{z}^{(k-1)} - \mathbf{x}), t)$
Stopping rule: stop if $\mathbf{z}^{(k)}$ has converged
Output: $\mathbf{z} = \mathbf{z}^{(k)}$

ISTA [22] is one of the best known iterative algorithms to solve the sparse linear problem. The q value in equation (4.2) of ISTA is 1, i.e., this is a convex problem where a local minimum is the global minimum. The detail of ISTA is shown in Algorithm 4.1, and the diagram is presented in Figure 2.2(a). For input vector $\mathbf{x}$, ISTA iterates the following recursive equation to approach the optimal:

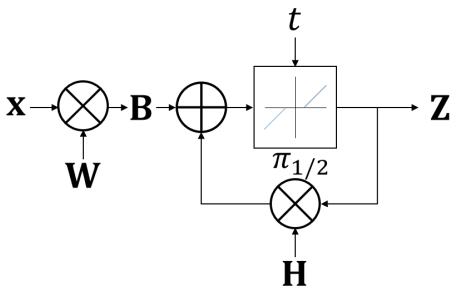
$$\mathbf{z}^{(k)} = \pi_1(\mathbf{W}\mathbf{x} + \mathbf{H}\mathbf{z}^{(k-1)}, t), \tag{4.4}$$

where $\mathbf{W} = \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}$, $\mathbf{H} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}\mathbf{D}$ and α is a restriction parameter, which should be larger than the largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$. The operator $\pi_1(\mathbf{x}, t)$ is a nonlinear soft thresholding operator, which is defined in equation (4.5).

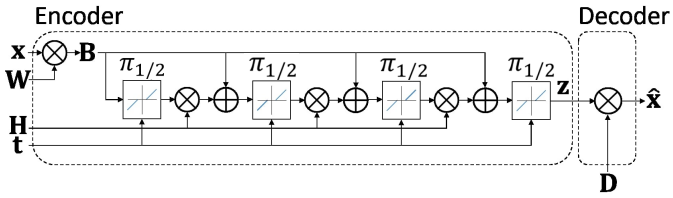
$$[\pi_1(\mathbf{x}, t)]_j = \mathrm{sign}(x_j) \max\{|x_j| - t, 0\} \tag{4.5}$$

4.2.2 IHTA

IHTA [27] concentrates on the nonconvex and nonsmooth optimization model, where the q value of equation (4.2) is 0.5. The detail of IHTA is shown in Algorithm 4.2, and the diagram is presented in Figure 4.1(a).



(a)



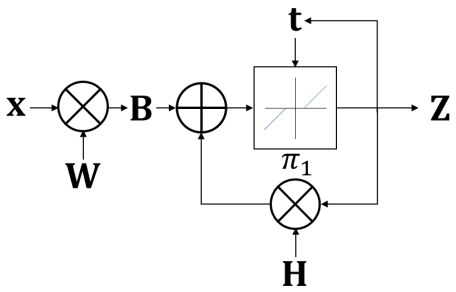
(b)

Figure 4.1: (a) Illustration of the IHTA structure for sparse coding. The optimal sparse representation can be obtained by the recursive structure $\mathbf{z}^{(k)} = \pi_{\frac{1}{2}}(\mathbf{W}\mathbf{x} + \mathbf{H}\mathbf{z}^{(k-1)}, t)$, where $\mathbf{x}$ is the input signal, $\pi_{\frac{1}{2}}(\mathbf{x}, t)$ is the half thresholding operator with threshold t, $\mathbf{W} = \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}$, $\mathbf{H} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}\mathbf{D}$ and α is a restriction parameter for IHTA. (b) The network structure of DNN-IHTA is formed from unfolded IHTA and truncated to a fixed number of iterations (3 here). $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{t}$ are trainable parameters in the network to provide an approximate sparse representation on a given dataset. The network can be trained supervisedly with only the encoder and unsupervisedly using both encoder and decoder.

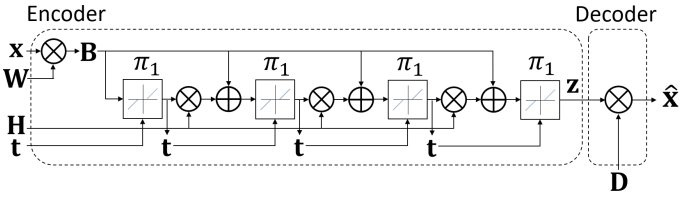
Algorithm 4.2: IHTA

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper parameters λ and α.
Restriction: $\alpha >$ largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$
Initialization: $t = \frac{\lambda}{\alpha}$, $\mathbf{z}^{(0)} = \mathbf{0}$, $k = 0$.
Main iteration: increase k by 1
$\mathbf{z}^{(k)} = \pi_{\frac{1}{2}}(\mathbf{z}^{(k-1)} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{z}^{(k-1)} - \mathbf{x}), t)$
Stopping rule: stop if $\mathbf{z}^{(k)}$ has converged
Output: $\mathbf{z} = \mathbf{z}^{(k)}$

In Algorithm 4.2, $\pi_{\frac{1}{2}}(\mathbf{x}, t)$ is the half thresholding operator, which is the key difference of IHTA compared to ISTA. Applying $p = 0.5$ may cause a convergence issue since it is nonconvex, whereas the lower $p \in (0, 1)$ value tends to more rapidly and efficiently achieve a sparser representation, meanwhile, the algorithm IHTA can lead to a converged result when λ is sufficiently small, and dictionary $\mathbf{D}$ satisfies a certain concentration assumption [71]. By applying the $\ell_{0.5}$ norm sparsity constraint, the half thresholding operator $\pi_{\frac{1}{2}}(\mathbf{x}, t)$ can be formulated as an analytical expression of the well-defined resolvent operator on its loss function [27], which



(a)



(b)

Figure 4.2: (a) Illustration of the WISTA structure for sparse coding. The optimal sparse representation can be recursively obtained in two steps: $\mathbf{z}^{(k)} = \pi_1(\mathbf{W}\mathbf{x} + \mathbf{H}\mathbf{z}^{(k-1)}, \mathbf{t}^{(k-1)})$, $\mathbf{t}^{(k)} = \frac{\lambda}{\alpha}|\mathbf{z}^{(k)}|^{p-1}$, where $\mathbf{x}$ is the input signal, $\pi_1(\mathbf{x}, \mathbf{t})$ is the soft thresholding operator with a changing threshold $\mathbf{t}$ during the iterations, $\mathbf{W} = \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}$, $\mathbf{H} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}\mathbf{D}$, and α is a restriction parameter for WISTA. (b) The network structure of DNN-IHTA is formed from the unfolded WISTA and truncated to a fixed number of iterations (3 here). $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{t}$ are trainable parameters in the network to provide an approximate sparse representation on a given dataset. The network can be trained supervisedly with only the encoder and unsupervisedly using both encoder and decoder.

is defined in (4.6).

$$\begin{aligned}\left[\pi_{\frac{1}{2}}(\mathbf{x}, t)\right]_j =& \frac{2}{3}x_j(1 + \cos(\frac{2\pi}{3} - \frac{2}{3}\arccos(\frac{t}{8}(\frac{|x_j|}{3})^{-1.5}))) \\ & \operatorname{sign}(|x_j| - \frac{\sqrt[3]{54}}{4}t^{\frac{2}{3}})\end{aligned} \tag{4.6}$$

4.2.3 WISTA

The proposed Weighted Iterative Shrinkage Thresholding Algorithm (WISTA) in this paper also concentrates on the nonconvex and nonsmooth ℓ_p regularization ($p \in (0, 1)$) optimization model considering the benefit in sparsity-inducing and efficiency. The word 'weighted' refers to the idea of restraining the ℓ_1 sparsity constraint with information from the previous iteration, which can approximately function as an ℓ_p norm. Namely, the sparsity constraint can be reformed as follows:

$$\lambda\|\mathbf{z}\|_p^p = \lambda\sum_i^n |z_i|^p \approx \lambda\sum_i^n |z_i^{(\text{iteration}-1)}|^{p-1}|z_i|. \tag{4.7}$$

Because the component-wise weighted part from the previous iteration can be considered constant during the iterations, the algorithm is transformed to a sequence of weighted ℓ_1 minimization problems from the ℓ_p minimization problem [32], which is convex for the calculation. The detail of WISTA is shown in Algorithm 4.3, and the diagram is presented in Figure 4.2(a).

Algorithm 4.3: WISTA

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper parameters λ and α.
Restriction: $\alpha >$ largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$
Initialization: $\mathbf{t} = \frac{\lambda}{\alpha}\mathbf{1}$, $\mathbf{z}^{(0)} = \mathbf{0}$, $k = 0$.
Main iteration: increase k by 1
$\mathbf{z}^{(k)} = \pi_1(\mathbf{z}^{(k-1)} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{z}^{(k-1)} - \mathbf{x}), \mathbf{t})$
$\mathbf{t} = \frac{\lambda}{\alpha}|\mathbf{z}^{(k)}|^{p-1}$
Stopping rule: stop if $\mathbf{z}^{(k)}$ has converged
Output: $\mathbf{z} = \mathbf{z}^{(k)}$

Different from [32, 33], where the gradient descent method is applied for the optimization, the proximal operator is applied in WISTA. Based on the difference in the sparse constraint, the form of soft thresholding operator $\pi_1(\mathbf{x}, \mathbf{t})$ in WISTA is slightly different from the one in ISTA, in which the input vector $\mathbf{z}$ is handled with vector $\mathbf{t}$ component-wise during the iterations, as defined in equation (4.8).

$$[\pi_1(\mathbf{x}, \mathbf{t})]_j = \text{sign}(x_j) \max\{|x_j| - t_j, 0\} \tag{4.8}$$

4.3 Deep Neural Network structured Sparse Coding

In the conventional deep learning [72], the training data, which comprise of pairs of feature and label, are used to learn the parameters of a deep neural network to predict unknown labels using the new given features. Typically, a Recurrent Neural Network (RNN) receives features and subjects them to a deep structure of many layers for processing, where each layer consists of a linear transformation followed by a component-wise nonlinearity transformation. The unfolded ISTA is shown in Figure 2.2 with unfolded IHTA (Figure 4.1) and unfolded WISTA (Figure 4.2); they hold similar structures to RNN, which enables the use of the parameter training methods in deep learning to train a DNN-SC algorithm. However, we should be clear about the differences between the two structures. The deep neural networks can be divided into two main types by their purposes, classification and regression. In a classification DNN, labels are generally discrete, e.g., when features are images, the labels will be their classes {dog, cat, ... , chicken}. In a contract, for a regression DNN and DNN-SC, the labels are numerical values,

which are geneallly continuous and high-dimensional.

For the objective of building a DNN-SC algorithm, it is essential to ensure that all functinos in the encoders are continuous and overall differentiable throughout the network structure, which provides the possibility of using gradient-based learning methods to train network parameters. For example, an overall differentiable structure ensure that back-propagation can provide gradients throughout.

In this paper, we would like to propose two novel DNN-SC algorithms: DNN-structured IHTA (DNN-IHTA) and DNN-structured WISTA (DNN-WISTA), which can be trained to compute approximate sparse codes. The two algorithms are based on IHTA and WISTA. We have applied two training modes in our proposed encoders: supervised and unsupervised.

4.3.1 Supervised and unsupervised learning

To train a DNN that is formed from a truncated sparse coding algorithm, referring to Figure 2.2(b), with training data $\mathbf{x}$ as the input, the encoder outputs $\mathbf{z}$ after the defined depth of the DNN. We use the gradient decent to train the parameters to minimize the loss function $L(\mathbf{z})$, which is defined as the squared error between the predicted code $\mathbf{z}$ from the DNN forward-propagation result and the corresponding optimal code $\mathbf{z}^*$ [46], as shown in equation (4.9). In practice, $\mathbf{z}^*$ can be obtained from the converged results of the corresponding sparse coding algorithms.

$$L(\mathbf{z}) = \|\mathbf{z} - \mathbf{z}^*\|_2^2 \tag{4.9}$$

Therefore, as the true answer $\mathbf{z}^*$ is functioned as training label, the loss function results in a supervised learning procedure. Then, for a T-layer DNN-SC, the gradient of $\mathbf{z}^{(T)}$ in supervised learning is

$$\delta\mathbf{z}^{(T)} = \frac{\partial L(\mathbf{z}^{(T)})}{\partial \mathbf{z}} = \mathbf{z}^{(T)} - \mathbf{z}^*. \tag{4.10}$$

In comparison, unsupervised learning aims at training the parameters in the DNN without requiring the true answers to serve as the training labels, i.e., optimal code $\mathbf{z}^*$ in DNN-SC. By adding a decoder in the network, as shown in Figure 2.2(c), we change the output of the DNN to $\hat{\mathbf{x}} = \mathbf{D}\mathbf{z}$. Therefore, we can use the general sparse representation loss function to avoid using $\mathbf{z}^*$ [47]. The loss function of unsupervised learning and the corresponding gradient of $\mathbf{z}^{(T)}$ is shown below.

$$L(\mathbf{z}) = \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda d_p(\mathbf{z}) \tag{4.11}$$

$$\delta \mathbf{z}^{(T)} = \frac{\partial L(\mathbf{z}^{(T)})}{\partial \mathbf{z}} = \mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{z}^{(T)} - \mathbf{x}) + \lambda \frac{\partial d_p(\mathbf{z}^{(T)})}{\partial \mathbf{z}} \tag{4.12}$$

Between two learning procedures resulted from different loss function settings, the key difference concerns that prior training is essential in supervised learning, which makes it difficult to implement the supervised DNN-SC in online processing and applications without training labels. On the contrary, by avoiding using $\mathbf{z}^*$, unsupervised learning enables the learned DNN for online processing when the learned DNN is efficient and learning procedure is fast enough.

4.3.2 DNN-structured IHTA

By unfolding the iterations of IHTA from Algorithm 4.2, we can construct neural network structures. The algorithm can be rewritten as a network structure as follows, and the unfolded network structure is shown in Figure 4.1(b).

Algorithm 4.4: DNN-IHTA Forward propagation

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper parameters λ and α, network layer T.
Restriction: $\alpha >$ largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$
Initialization: $\mathbf{t} = \frac{\lambda}{\alpha}\mathbf{1}$, $\mathbf{z}^{(0)} = \mathbf{0}$, $\mathbf{H} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}\mathbf{D}$, $\mathbf{W} = \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}$, $\mathbf{B} = \mathbf{W}\mathbf{x}$.
For $k = 1$ to T
  $\mathbf{c}^{(k-1)} = \mathbf{B} + \mathbf{H}\mathbf{z}^{(k-1)}$
  $\mathbf{z}^{(k)} = \pi_{\frac{1}{2}}(\mathbf{c}^{(k-1)}, \mathbf{t})$
End
Output: $\mathbf{C} = \{\mathbf{c}^{(0)}, ..., \mathbf{c}^{(T-1)}\}$, $\mathbf{Z} = \{\mathbf{z}^{(0)}, ..., \mathbf{z}^{(T)}\}$

Algorithm 4.5: DNN-IHTA Back propagation

Input: $\mathbf{x}$, $\mathbf{D}$, $\delta\mathbf{z}^{(T)}$, $\mathbf{Z}$, $\mathbf{C}$, $\mathbf{B}$, $\mathbf{H}$, $\mathbf{t}$, λ and α.
Initialization: $\delta\mathbf{t}^{(T)} = \mathbf{0}$, $\delta\mathbf{B}^{(T)} = \mathbf{0}$, $\delta\mathbf{H}^{(T)} = \mathbf{0}$.
For $k = T - 1$ down to 0
  $\delta\mathbf{t}^{(k)} = \delta\mathbf{t}^{(k+1)} + \frac{\partial \pi_{\frac{1}{2}}((c)^{(k)}, \mathbf{t})}{\partial \mathbf{t}} \delta\mathbf{z}^{(k+1)}$
  $\delta\mathbf{c}^{(k)} = \frac{\partial \pi_{\frac{1}{2}}((c)^{(k)}, \mathbf{t})}{\partial \mathbf{c}} \delta\mathbf{z}^{(k+1)}$
  $\delta\mathbf{B}^{(k)} = \delta\mathbf{B}^{(k+1)} + \delta\mathbf{c}^{(k)}$
  $\delta\mathbf{H}^{(k)} = \delta\mathbf{H}^{(k+1)} + \delta\mathbf{c}^{(k)}{\mathbf{z}^{(k)}}^{\mathrm{T}}$
  $\delta\mathbf{z}^{(k)} = \mathbf{H}^{\mathrm{T}}\delta\mathbf{c}^{(k)}$
End
$\delta\mathbf{W} = \delta\mathbf{B}^{(0)}\mathbf{x}^{\mathrm{T}}$
Output: $\delta\mathbf{W}$, $\delta\mathbf{H}^{(0)}$, $\delta\mathbf{t}^{(0)}$

In the network, $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{t}$ are parameters to train. After forward propagating with determined network layer T, $\mathbf{C}$ and $\mathbf{Z}$ of each layer in the IHTA network are saved for the back-propagation parameter training. The gradient-based parameter learning schedules of the network are shown in Algorithm 4.5. The learning procedure can be either adapted to super-

vised or unsupervised learning by respectively choosing $\delta \mathbf{z}^{(T)}$ from equation (4.10) or equation (4.12).

4.3.3 DNN-structured WISTA

Using the schedule in the previous section, by unfolding the iterations of WISTA from Algorithm 4.3, we can also construct a neural network structure with linear transformations and a simple nonlinear transformation in each layer, as shown in Figure 4.2(b).

Algorithm 4.6: DNN-WISTA Forward propagation

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper parameters λ and α, network layer T.
Restriction: $\alpha >$ largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$
Initialization: $\mathbf{t}^{(0)} = \frac{\lambda}{\alpha}\mathbf{1}$, $\mathbf{z}^{(0)} = \mathbf{0}$, $\mathbf{H} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}\mathbf{D}$, $\mathbf{W} = \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}$, $\mathbf{B} = \mathbf{W}\mathbf{x}$.
For $k = 1$ to T
  $\mathbf{c}^{(k-1)} = \mathbf{B} + \mathbf{H}\mathbf{z}^{(k-1)}$
  $\mathbf{z}^{(k)} = \pi_1(\mathbf{c}^{(k-1)}, \mathbf{t}^{(k-1)})$
  $\mathbf{t}^{(k)} = \frac{\lambda}{\alpha}|\mathbf{z}^{(k)}|^{p-1}$
End
Output: $\mathbf{C} = \{\mathbf{c}^{(0)}, ..., \mathbf{c}^{(T-1)}\}$, $\mathbf{Z} = \{\mathbf{z}^{(0)}, ..., \mathbf{z}^{(T)}\}$

Algorithm 4.7: DNN-WISTA Back propagation

Input: $\mathbf{x}$, $\mathbf{D}$, $\delta \mathbf{z}^{(T)}$, $\mathbf{Z}$, $\mathbf{C}$, $\mathbf{B}$, $\mathbf{H}$, $\mathbf{T}$, λ and α.
Initialization: $\delta \mathbf{B}^{(T)} = \mathbf{0}$, $\delta \mathbf{H}^{(T)} = \mathbf{0}$.
For $k = T - 1$ down to 0
  $\delta \mathbf{t}^{(k)} = \frac{\partial \pi_1((c)^{(k)}, \mathbf{t}^{(k)})}{\partial \mathbf{t}} \delta \mathbf{z}^{(k+1)}$
  $\delta \mathbf{c}^{(k)} = \frac{\partial \pi_1((c)^{(k)}, \mathbf{t}^{(k)})}{\partial \mathbf{c}} \delta \mathbf{z}^{(k+1)}$
  $\delta \mathbf{B}^{(k)} = \delta \mathbf{B}^{(k+1)} + \delta \mathbf{c}^{(k)}$
  $\delta \mathbf{H}^{(k)} = \delta \mathbf{H}^{(k+1)} + \delta \mathbf{c}^{(k)} {\mathbf{z}^{(k)}}^{\mathrm{T}}$
  $\delta \mathbf{z}^{(k)} = \mathbf{H}^{\mathrm{T}} \delta \mathbf{c}^{(k)} + \frac{\lambda(p-1)|\mathbf{z}^{(k)}|^{p-2}\mathrm{sign}(\mathbf{z}^{(k)})}{\alpha} \delta \mathbf{t}^{(k)}$
End
$\delta \mathbf{W} = \delta \mathbf{B}^{(0)} \mathbf{x}^{\mathrm{T}}$
Output: $\delta \mathbf{W}$, $\delta \mathbf{H}^{(0)}$, $\delta \mathbf{t}^{(0)}$

In this network, $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{t}$ are targeted trainable parameters. After forward propagating with determined network layer T, $\mathbf{C}$ and $\mathbf{Z}$ of each layer in the WISTA network are saved for the back-propagation parameter training. The gradient-based parameter learning schedules of the two networks are shown in Algorithm 4.7. The learning procedure can be either adapted to supervised learning by defining $\delta \mathbf{z}^{(T)}$ as equation (4.10) or unsupervised learning referring to equation (4.12).

4.4 Experiments

4.4.1 Synthetic data experiments

In this section, we present our experiment results of the proposed algorithms in finding the ground truth sparse representations. Synthetically generated data were used throughout the experiments, which were built by randomly generated ground true dictionaries. All experiments were performed with Matlab R2016a, and the programs were run on a PC with a 3.4 GHz Intel core and 64 G of RAM.

There are two main parts that we would like to show in synthetic data experiments: comparison of performances among the ISTA, IHTA and WISTA and the performances of different DNN-SC algorithms. In the synthetic data simulation experiments, we first formed the dictionary $\mathbf{D}_{\text{orig}}$, which is sized as a 250×500 matrix, generated by randomly drawing value from a normal distribution $N(0, 1)$ and finally column-normalized. In every sparse representation vector, non-zero values were set in random positions determined by the Bernoulli distribution of possibility 0.05. The random values were selected from the normal distribution $N(0, 1)$. There were 100 ground truth-generated sparse representation vectors $\mathbf{Z}_{\text{orig}} = (\mathbf{z}_{\text{orig}1}, ..., \mathbf{z}_{\text{orig}100})$ in the experiments. Correspondingly, data set $\mathbf{X}_{\text{orig}}$ had 100 samples, which were generated by $\mathbf{D}_{\text{orig}}$ and $\mathbf{Z}_{\text{orig}}$ based on equation (4.1), The noise vectors $\mathbf{v}$ were added based on Gaussian random entries with 20 dB SNR.

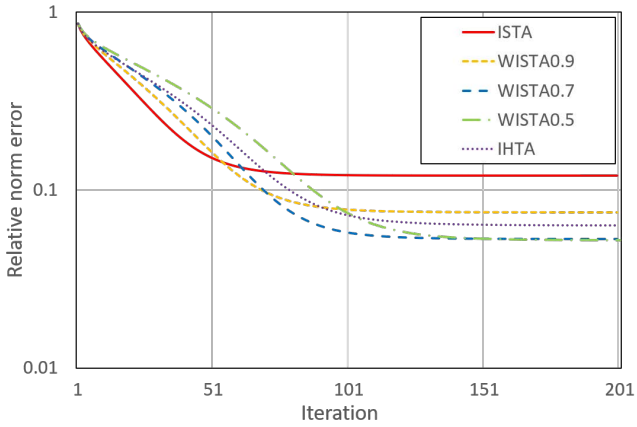
Performances of sparse coding algorithms

The sparsity measurement used the Hoyer measure [66] based on the relationship between the ℓ_1-norm and the ℓ_2-norm, which can provide a well-defined sparsity. Hoyer sparsity measure is formulated as follows:

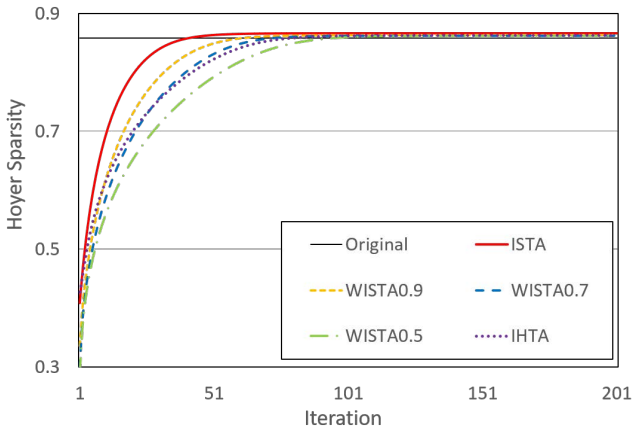
$$\text{Hoyer sparsity}(\mathbf{z}) = \frac{\sqrt{n} - (\sum |z_i|)/\sqrt{\sum {z_i}^2}}{\sqrt{n} - 1}, \tag{4.13}$$

where n is the dimension of $\mathbf{z}$; when the value of the equation is closer to 1, the $\mathbf{z}$ vector approaches a sparse vector.

Figure 4.3 and Figure 4.4 show the ability of recovering accurate sparse representations using the three sparse coding algorithms. Three p values are used in our proposed algorithm WISTA, which is referred by the number after WISTA. The sparse representation error use the



(a)



(b)

Figure 4.3: Average sparse representation errors and Hoyer sparsity convergence graph of different sparse coding algorithms.

relative norm error compared to $\mathbf{Z}_{\text{orig}}$, which is defined in equation (4.14).

$$\text{Relative norm error}(\mathbf{z}_i) = \frac{\|\mathbf{z}_{\text{orig}i} - \mathbf{z}_i\|_2^2}{\|\mathbf{z}_{\text{orig}i}\|_2^2} \tag{4.14}$$

Figure 4.3 shows that all algorithms can achieve convergence results with small error and high sparsity in reasonable iterations. ISTA tends to converge most rapidly with a larger relative norm error than the other algorithms. With decreasing p, WISTA converges slower and result in a smaller relative error. When $p = 0.7$, WISTA surpasses IHTA in both converge speed and relative error. Figure 4.4 shows that all algorithms can generally find accurate and sparse representations but ignore several small values compared to the ground true representation (red). ISTA has the largest amount of small value mistakes, so it has a larger relative error than the other algorithms. For WISTA, the number of support mistakes, where the blue points do not overlap with the red ones, tend to diminish when p decreases from 0.9 to 0.7, while one small

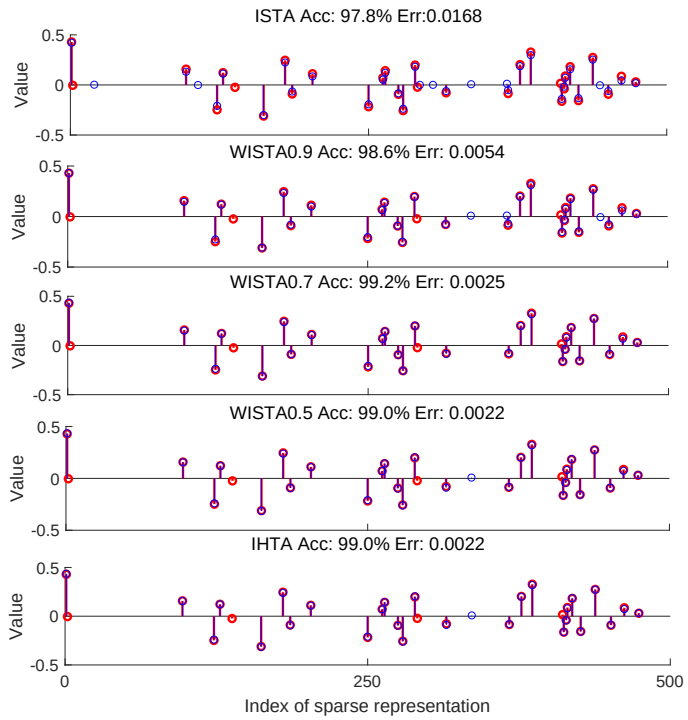


Figure 4.4: Index versus value of the recovered sparse representation (blue) compared with the original one (red) of ISTA, IHTA, WISTA0.9, WISTA0.7, and WITA0.5 (from top to bottom).

mistake in support reappears in WISTA0.5, which indicates that an appropriate lower p value can help finding sparser and more accurate results.

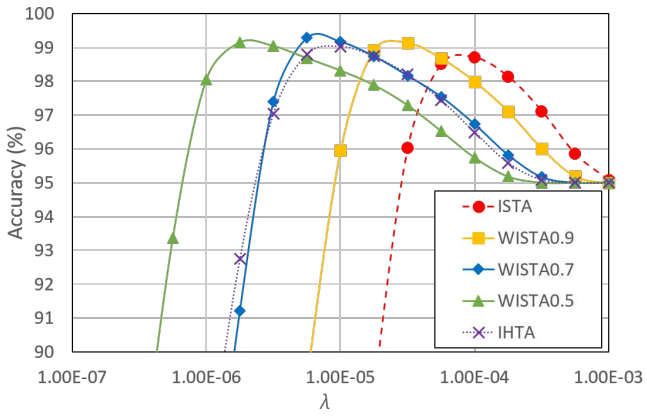
Figure 4.5 shows the performance variation in a range of varing λ values. The sparse representation accuracy is defined in equations (4.15) and (4.16), which can indicate how accurate the sparse coding algorithms can recover the positions of those non-zero values.

$$S = \text{Support}\{\mathbf{z}\} \tag{4.15}$$

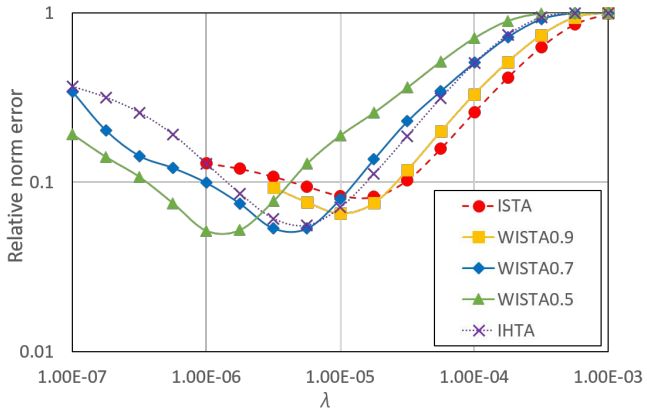
$$\text{Accuracy}(\mathbf{z}) = \frac{|S_{orig} \cap S|}{n} \times 100\% \tag{4.16}$$

The definition of support is a set containing information of zero and non-zero positions in $\mathbf{z}$. Therefore, the accuracy is the percentage of how a given $\mathbf{z}$ meets the ground truth considering whether the values are zero. In the two figures, different algorithms have different ranges of effective λ. While deciding the optimal λ regions, we find that the λ value with the smallest relative norm error is always smaller than the λ with highest accuracy for all algorithms. In detail, ISTA has the worst performance in dislocating the optimal regions. The accuracy is smaller than 90% when ISTA reaches the lowest relative norm error, which results from the

those support mistakes with small values in Figure 4.4. Again, all other algorithms tend to perform better than ISTA in finding more accurate sparse representations. WISTA surpasses IHTA in both relative norm error and accuracy when $p = 0.7$ and 0.5. With decreasing p, WISTA tends to achieve smaller relative errors, whereas $p = 0.7$ results in the highest accuracy.



(a)



(b)

Figure 4.5: Average sparse representation accuracies and relative norm errors of different sparse coding algorithms in a range of varing λ.

Performances of DNN-structured sparse coding algorithms

In the synthetic data experiment of DNN-SC algorithms, the DNN structures with various layers are used in different algorithms to show the performance of the combination of the sparse representation and DNN.

Figure 4.6 shows the performances of different original sparse coding algorithms and their DNN-structured versions when there are 15 layers. The initial letter 'L' refers to supervised learned DNN, and the initial letter 'T' refers to unsupervised learned DNN. In general, except

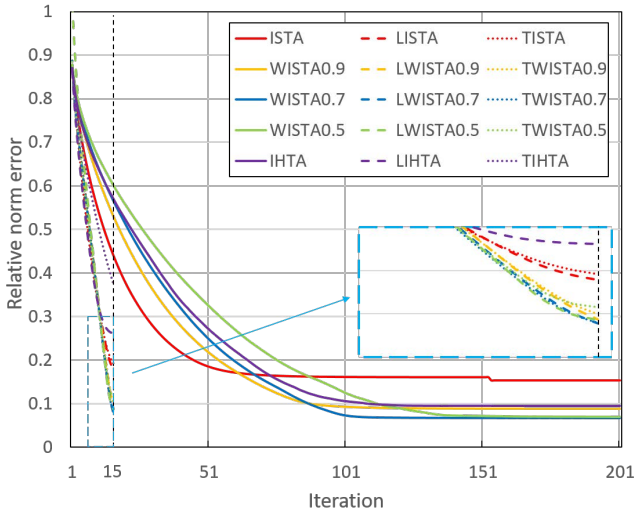
the DNN-structured IHTA, all other DNN-structured algorithms can reach similar performances to the converged results of their original sparse coding algorithms, which indicates that those learned DNN can accelerate 5-10 times in this experiment. Similar to the previous sparse coding experiment results, DNN-structured WISTA can perform better than DNN-structured ISTA when there are 15 layers.

Figure 4.7 shows the result convergence performances of different original sparse coding algorithms and their DNN-structured versions in a range of DNN layers. In general, we find four points. 1) Except DNN-structured IHTA, all DNN-structured algorithms can nearly reach converged performances of their original algorithms when there are more than 13 layers in this simulation; 2) the performances of the DNN-structured algorithms tend to improve with the increase in number of layers; 3) DNN-structured WISTA can perform better than DNN-structured ISTA; and 4) the supervised DNN tends to obtain better relative norm error than the corresponding unsupervised ones.

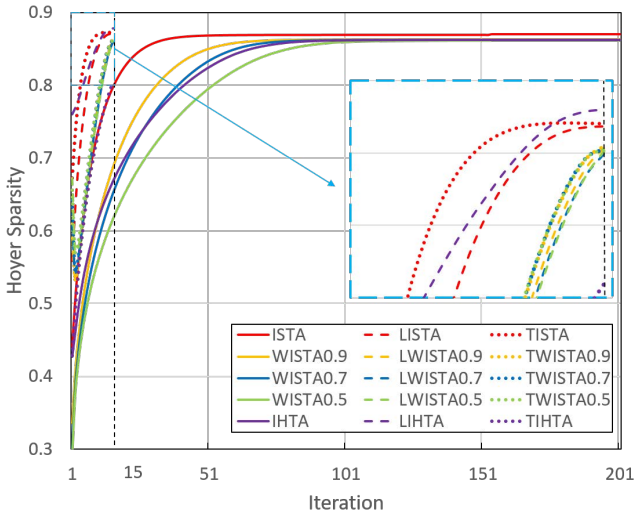
Specifically, LISTA and TISTA have the best robustness against the decrease in number of layer in this data simulation. Through the selected range of number of layer, both LISTA and TISTA can reach similar relative norm errors compared to the converged ISTA, and LISTA and TISTA have better accuracy than ISTA when there are more than 15 layers. For WISTA, decreasing p in the DNN-structured WISTA algorithms tends to increase their minimum layer requirement to have similar performances to the converged WISTA, which may indicate that a smaller p value increases the difficulty to learn an appropriate network. Although supervised WISTAs have equivalent or better relative norm error than the unsupervised ones, the unsupervised versions have obvious advantages in finding more accurate positions in sparse representations, which implies that the supervised DNN can hardly avoid small support mistakes in sparse representations, which do not greatly affect the relative norm error but reduce the position accuracy.

4.4.2 Graphic denoising experiments

In this section, we present the performance of our proposed algorithms with real-world data. There are two main parts. First, we want to show the performance of the algorithms in the image-denoising task. Then, we present the potential of applying DNN-SC algorithms in real-time video-denoising tasks.

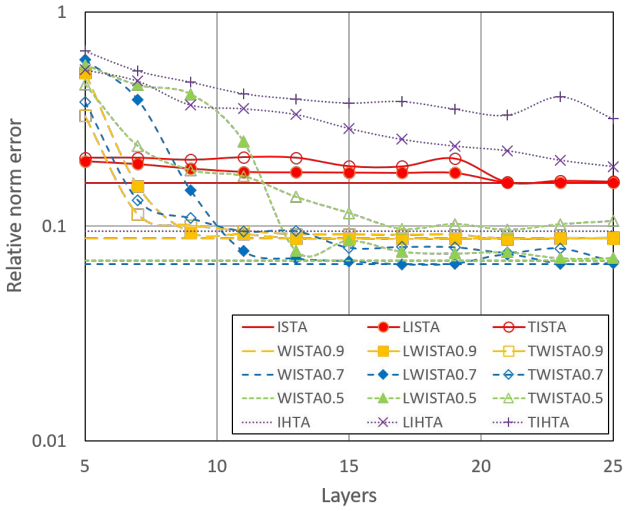


(a)

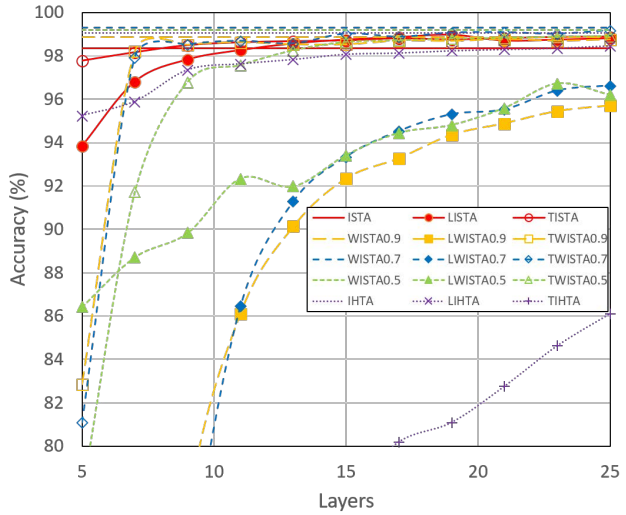


(b)

Figure 4.6: Comparison between 15-layer DNN-SC algorithms and the original sparse coding algorithms in terms of the average sparse representation relative norm errors and Hoyer sparsity.



(a)



(b)

Figure 4.7: Comparison between the DNN-SC algorithms and the converged results of their original algorithms in terms of the average sparse representation relative norm errors and accuracy in a range of number of layers.

Table 4.1: Denoising results of the sparse coding algorithms from a 20.17 dB noised image

Algorithms	PSNR (dB)	Denoising time (s)
ISTA	29.13	187.89
WISTA0.9	29.88	260.74
WISTA0.7	30.79	335.24
WISTA0.5	31.01	339.84
IHTA	31.06	445.65
OMP	30.93	77.94

Image-denoising experiments

To present the denoising performance, we applied the aforementioned algorithms in image denoising in comparison with sparse coding algorithm OMP [21, 43], which performs well in this task. The details of the image-denoising experiment are described below. We selected the image named 'Lena' as the target, which is a 512×512-pixel gray-scaled portrait photograph. Random white noise was added to the image to generate a noised image of Peak signal-to-noise ratio (PSNR) 20.17 dB. Referring to the signal model in equation (4.1), we generate a 144×256 overcomplete Discrete Cosine Transform (DCT) distributed dictionary for the denoising task. The image was separated into 12×12-pixel small patches with an interval of 2 between patches to form the input data set $\mathbf{X} \in \mathbb{R}^{64 \times 62009}$. All DNN-SC algorithms use 10-layer network structures in this experiment. The denoising results of the original sparse coding algorithms and their DNN-structured versions are presented below.

Table 4.1 and Figure 4.8 show that all sparse coding algorithms can recover the noised image from 20.17 dB to approximately 30 dB, whereas all iterative shrinkage sparse coding algorithms tend to have several times higher computation time than OMP. In detail, IHTA and WISTA0.5 obtain the highest PSNR among these algorithms with PSNR over 31 dB. In this image-denoising task, we observe an obvious increase in PSNR and denoising time when p decreases among all iterative shrinkage sparse coding algorithms, which may indicate that a smaller p value is more suitable for the overcomplete sparse denoising model. To resolve the excessive computation cost of the iterative shrinkage sparse coding algorithms, the DNN-SC algorithm can be a solution that the learned DNN has shown 5$\sim$10 times acceleration in the previous synthetic data simulation.

First, Table 4.2 and Figure 4.9 show that all DNN-SC algorithms can recover the noised image from 20.17 dB to approximately 30 dB, which is the same level of their original sparse coding algorithms. Furthermore, the learned DNNs can accelerate the denoising procedure, the sum of denoising and DNN learning time remains at least 45 times faster than the denoising

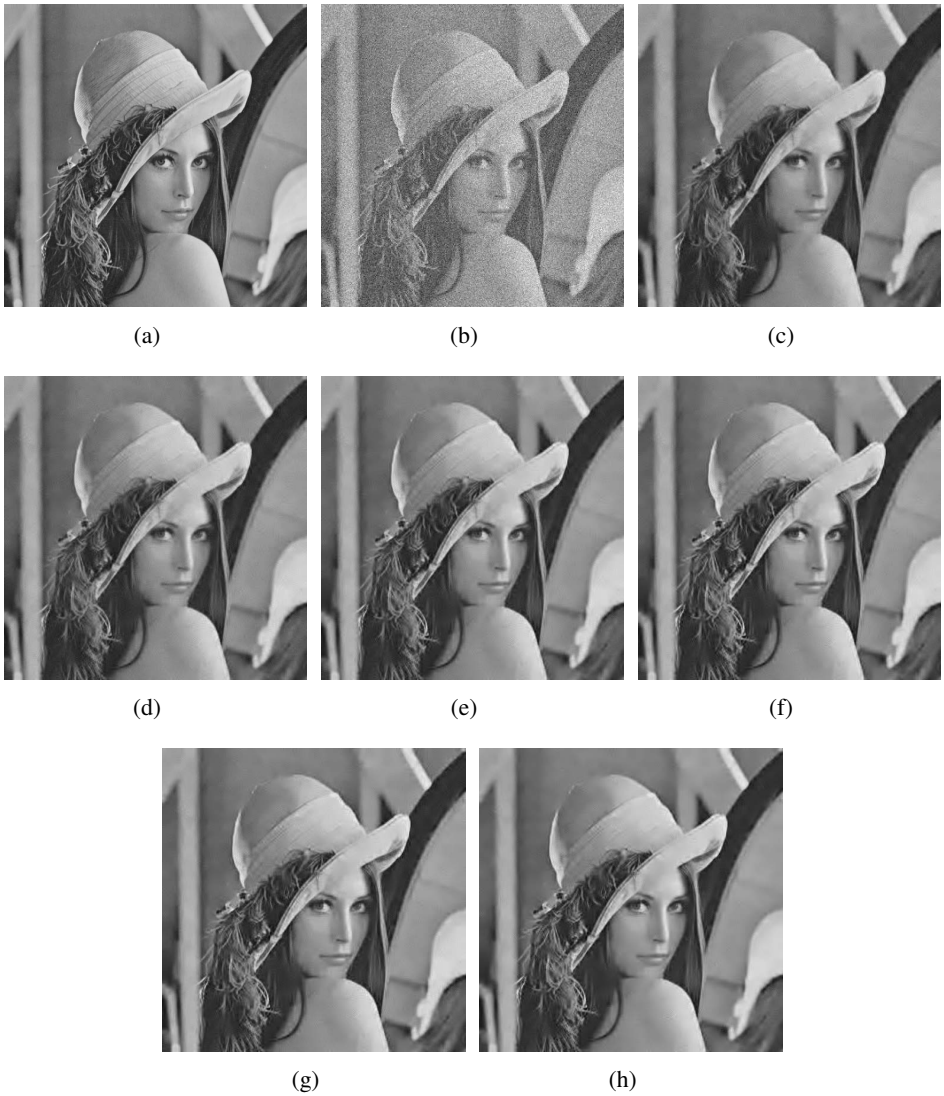
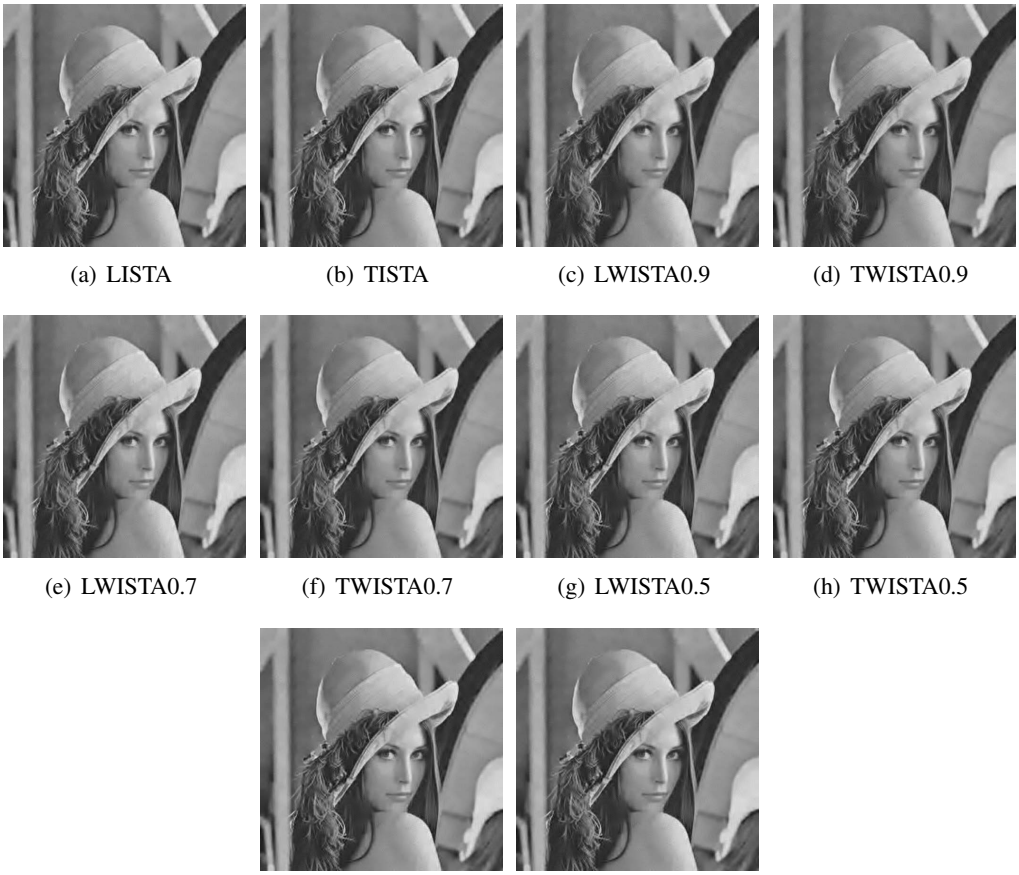


Figure 4.8: Image-denoising result: (a) Original image; (b) Noised: 20.17 dB; (c) ISTA: 29.13 dB; (d) WISTA0.9: 29.88 dB; (e) WISTA0.7: 30.79 dB; (f) WISTA0.5: 31.01 dB; (g) IHTA: 31.00 dB; and (h) OMP: 30.93 dB (from top left to bottom right)

Table 4.2: Denoising results of DNN-SC algorithms from a 20.17 dB noised image

Algorithms	PSNR (dB)	Denoising time (s)	Learning time (s)
LISTA	29.71	2.52	0.05
TISTA	29.37	2.50	0.02
LWISTA0.9	29.89	5.82	0.03
TWISTA0.9	29.69	6.01	0.03
LWISTA0.7	30.45	5.90	0.03
TWISTA0.7	30.47	5.79	0.03
LWISTA0.5	30.68	5.84	0.03
TWISTA0.5	30.76	5.88	0.03
LIHTA	30.82	9.17	0.16
TIHTA	30.80	9.26	0.07



(a) LISTA (b) TISTA (c) LWISTA0.9 (d) TWISTA0.9

(e) LWISTA0.7 (f) TWISTA0.7 (g) LWISTA0.5 (h) TWISTA0.5

(i) LIHTA (j) TIHTA

Figure 4.9: Image-denoising result with DNN-SC algorithms: (a) LISTA: 29.71 dB; (b) TISTA: 29.37 dB; (c) LWISTA0.9: 29.89 dB; (d) TWISTA0.9: 29.69 dB; (e) LWISTA0.7: 30.45 dB; (f) TWISTA0.7: 30.47 dB; (g) LWISTA0.5: 30.68 dB; (h) TWISTA0.5: 30.76 dB; (i) LIHTA: 30.82 dB; and (j) TIHTA: 30.80 dB (from top left to bottom right)

time cost of their corresponding sparse coding algorithms. In detail, the unsupervised learned DNNs have similar PSNR values to their supervised versions, which proves that unsupervised learning can function in the graphic denoising task. Among all DNNs, DNN-structured WISTA and IHTA are better than the others: the PSNRs of the denoised image using their original sparse coding algorithms are approximately 0.25 dB slightly better, but the results remain reasonably good because DNN-structured WISTA0.5 can be over 10 times faster than OMP.

Video-denoising experiments

In this section, we propose a procedure to apply the DNN-SC algorithm to real-time video denoising. The details of the video-denoising experiment are described below. We selected two videos from the dataset created by Gygli et al. [73] to test the denoising performance. The first video, which is named 'Fire Domino', is a 360×480-pixel gray-scaled video captured by a fixed camera; 'Fire Domino' is composed of 1612 frames at a rate of 25 frames/s (FPS). The second video, which is named 'Statue of Liberty', is a 360×480-pixel gray-scaled video captured by people in daily life; 'Statue of Liberty' has 1500 frames in total at a rate of 25 FPS. Random white noise was added to the two videos to generate noised videos with PSNR of approximately 20 dB. Referring to the signal model in equation (4.1), we generated a 225×256 overcomplete DCT distributed dictionary for this video-denoising task. Each video frame was separated into 15×15-pixel small patches with an interval of 10 between patches to form the input data set $\mathbf{X} \in \mathbb{R}^{225 \times 1645}$. All DNN-SC algorithms used 4-layer network structures in this experiment. Video streaming was inputed into the DNN frame by frame, which implies that the DNNs learn from one frame and finish denoising before processing the next frame. The denoising results of unsupervised DNN-SC algorithms are presented below.

Table 4.3, Table 4.4, Figure 4.10 and Figure 4.11 show that all unsupervised DNN-SC algorithms can successfully recover noised video streaming from 20 dB to approximately 30 dB with reasonable fast denoising time. More importantly, both TISTA and TWISTA can restrict the sum of denoising and DNN learning time to 0.04 s/frame, which implies that these two DNN-SC algorithms can conduct real-time video denoising for a 25-FPS 360×480-pixel gray-scaled video. In detail, we observe that TISTA continues being the fastest algorithm in processing, but its PSNR is relatively the worst. TIHTA achieves the highest PSNR among these algorithms, but its processing time is nearly twice that of TWISTA. TWISTA0.5 is the best algorithm, which restrains the processing time in the frame internal of a 25-FPS video. Furthermore, the denoising

Table 4.3: Average denoising results of the DNN-SC algorithms from the first noised video 'Fire Domino' with an initial PSNR of 20.17 ± 0.02 dB

Algorithms	PSNR(dB)	Denoising time(s)	Learning time(s)
TISTA	30.35 ± 1.06	0.015 ± 0.009	0.004 ± 0.001
TWISTA0.9	31.15 ± 1.23	0.036 ± 0.021	0.004 ± 0.001
TWISTA0.7	31.74 ± 1.25	0.036 ± 0.021	0.004 ± 0.001
TWISTA0.5	31.83 ± 1.30	0.036 ± 0.021	0.004 ± 0.001
TIHTA	31.45 ± 1.06	0.071 ± 0.043	0.014 ± 0.005

(a) Noised:20.19dB (b) TISTA:31.35dB (c) TWISTA0.9:32.00dB

(d) TWISTA0.7:32.71dB (e) TWISTA0.5:32.89dB (f) TIHTA:32.44dB

Figure 4.10: Denoising results of one frame in the video 'Fire Domino' from the initial PSNR of 20.19dB

time of the video with identical resolution may vary depending on the video; thus, the average denoising time of different algorithms for the second video is approximately 75% compared to the time cost for the first video.

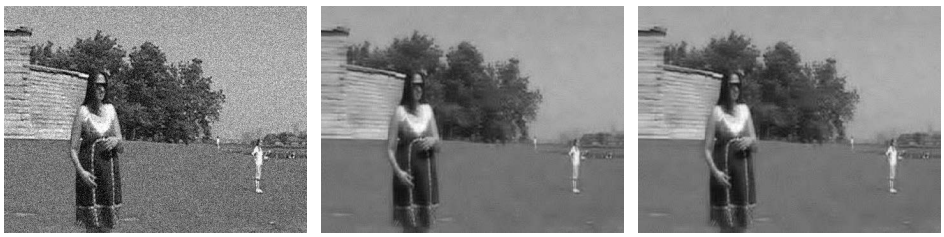
4.4.3 Discussion

Throughout the conducted experiments, there are two points for discussion.

1. DNN-IHTA, which fails in giving a close approximation of the converged IHTA in syn-

Table 4.4: Average denoising results of DNN-SC algorithms for the noised video 'Statue of Liberty' with the initial PSNR of 20.17 ± 0.01dB

Algorithms	PSNR (dB)	Denoising time (s)	Learning time (s)
TISTA	28.70 ± 1.53	0.012 ± 0.006	0.004 ± 0.001
TWISTA0.9	29.12 ± 1.58	0.029 ± 0.013	0.004 ± 0.001
TWISTA0.7	29.38 ± 1.63	0.028 ± 0.013	0.004 ± 0.001
TWISTA0.5	29.47 ± 1.71	0.028 ± 0.013	0.004 ± 0.002
TIHTA	29.81 ± 1.52	0.055 ± 0.025	0.008 ± 0.003



(a) Noised:20.17dB (b) TISTA:27.80dB (c) TWISTA0.9:28.24dB

(d) TWISTA0.7:28.48dB (e) TWISTA0.5:28.54dB (f) TIHTA:29.00dB

Figure 4.11: Denoising results of one frame in the video 'Statue of Liberty' from the initial PSNR of 20.17dB

thetic data experiments, can function well in graphic denoising experiments.

In Figure 4.7, DNN-IHTA can hardly approach the results of IHTA in synthetic data experiments for all tested DNN layers, but Table 4.2, Table 4.3 and Table 4.4 show that TIHTA can work well and achieve similar results to the converged results of IHTA. The key difference is the data set. In the synthetic data experiments, data were randomly generated and obey the normal distribution $N(0, 1)$, but in denoising experiments, the data were extracted small patches from an image or a video, i.e., the data may have certain continuity and dependence among the data columns. Figure 4.7 also shows that DNN-structured WISTA requires more DNN layers to achieve the converged results of WISTA as p decreases, whereas DNN-structured ISTA does not have this problem. One possibility is that the problem is caused by the nonconvex feature in IHTA and WISTA, which makes the back-propagation failed in finding the global optimal in random meaningless data set.

2. In both image and video denoising, the DNN learning time is small.

   Table 4.2, Table 4.3 and Table 4.4 show that all DNN-SC algorithms only require tens of millisecond to complete their learning procedure, which ensures that the DNN update is sufficiently fast for online processing. The reason is that the DNNs in these experiments only use a small number of patches from the image until the learning procedure converges,

which is not possible for the learning procedure of synthetic data experiments. In the synthetic data experiments, the DNN learning should pass several epochs of all input data to make the learned DNN converge and effective. The possible answer may again be the difference in data set. All image patches have identical standard white noise levels on a continuous and repeating graphic signal, whereas the synthetic data put white noises on random generated signals, which are also the Gaussian distribution. Thus, the denoising procedure for each patch may have certain similarities that the learned DNN from a small number of patches can stand for the entire image.

4.5 Chapter Summary

In this chapter, we have proposed two DNN-SC algorithms. The first algorithm applies deep learning approaches to WISTA, which is a modified sparse coding algorithm of ISTA proposed in this paper. WISTA considers to approximate the ℓ_p norm sparse coding problem by joining the information of the sparse representation from the previous iteration. The 'weighted' idea enables one to enjoy the advantages of the ℓ_p norm sparse constraint while maintaining the convex optimization model. The second approach combines IHTA [27] with deep learning, which is an $\ell_{0.5}$ norm sparse coding algorithm. We state the differences between two DNN learning schedules for DNN-SC algorithms: supervised and unsupervised. The benefit of unsupervised DNN learning is that it does not require input signals associated with labels, which enables one to apply DNN-SC algorithms to image denoising, video denoising and other applications that lack paired training samples.

The synthetic data experiments show that WISTA can outperform both ISTA and IHTA in terms of the relative norm error and accuracy. In addition, WISTA can retain the advantages of supervised and unsupervised DNN versions. We also find that unsupervised DNNs can achieve similar performance to their corresponding supervised ones. However, DNN-IHTA can hardly learn the appropriate parameters to yield a close approximation of the converged IHTA in synthetic data. Furthermore, it is difficult for DNN-WISTA to train the parameters when there are few layers, but all DNN-WISTAs can function well with at least 15 layers, which remains reasonable.

Then, we have applied the proposed algorithms to image and video denoising, which benefit from the unsupervised learning procedure and fast DNN learning time for graphic processing.

The denoising results show that the DNN-SC algorithms can accelerate the denoising procedures at least 45 times while maintaining reasonably good performances from 20 dB to approximately 30 dB. Then, we conducted denoising experiments on two videos. Using TISTA and TWISTA, the total processing time for each frame can be restricted to 0.04 s/frame, which indicates that TISTA and our proposed TWISTA can be applied in real-time video denoising for a 25-FPS video with good denoising results. Although we only conduct experiments for 25-FPS 360×480-pixel gray-scaled videos, the future work may extend to higher FPS, higher resolution and colored videos.

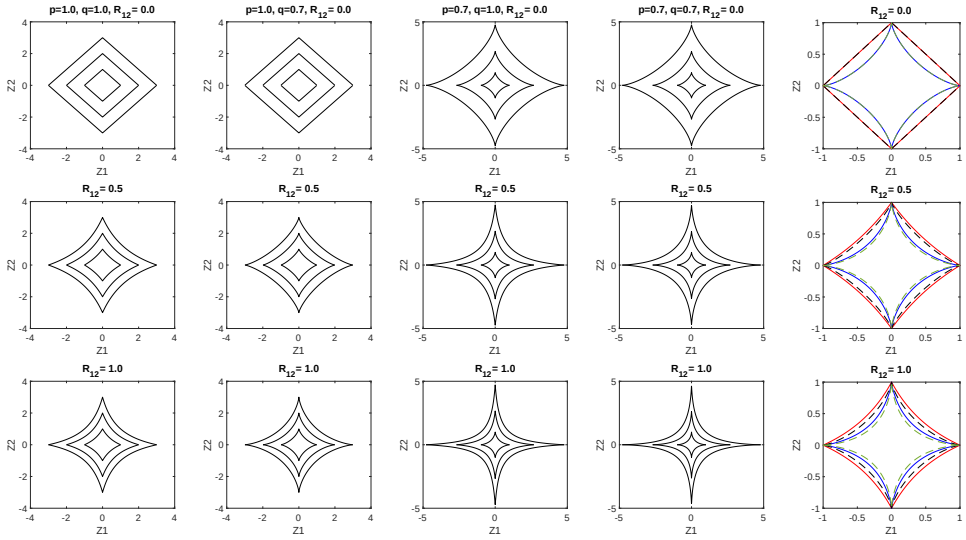
Chapter 5

ℓ_p Norm Independently Interpretable Regularization based Sparse Coding for Highly Correlated Data

5.1 Introduction

Reviewing the procedure of sparse coding, which concentrates on presenting sparse estimations from underdetermined linear measurements, based on predefined overcomplete vector sets. Utilizing sparse coding can perform parameter estimation and feature selection simultaneously, making it a powerful tool in processing high dimensional data, which is commonly appeared in biology, economy and industry [74–78]. Choosing appropriate sparse regularization to implement sparse coding is the guarantee for obtaining results efficiently and accurately.

However, commonly used sparse regularizations only consider low coherence conditions of data. When signals are highly correlated, sparse coding with normal sparse regularization cannot efficiently interpret the decomposability of model. In 2018, Takada et al. propose a new regularization, named Independently Interpretable Lasso (IILasso), which is composed with coherence between dictionary columns to enhance the ability of choosing uncorrelated variables in sparse coding [40]. IILasso has proven to be efficient in highly correlated data and provides smaller misclassification error than several other sparse coding algorithms. However, IILasso has the same problem as the other ℓ_1 norm based regularization that its result is not sparse and accurate enough. In this paper, we propose to introduce ℓ_p norm $(0 < p < 1)$ into the regularization of IILasso, which can enhance performance both in sparsity and accuracy. However,



(a) (b) (c) (d) (e)

Figure 5.1: (a)-(d) Contours of the constraint when $d_p(\mathbf{z}) + h_q(\mathbf{z}) = \|\mathbf{z}\|_p^p + (|\mathbf{z}|^q)^{\mathrm{T}}\mathbf{R}|\mathbf{z}|^q = 1, 2, 3$ with $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2]$ and $\mathbf{R} = [0, 0; 0, 0], [0, 0.5; 0.5, 0], [0, 1; 1, 0]$ from top to bottom for each row; (e) Contours comparison among the previous 4 cases when $d_p(\mathbf{z}) + h_q(\mathbf{z}) = 1$ with $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2]$ and $\mathbf{R} = [0, 0; 0, 0], [0, 0.5; 0.5, 0], [0, 1; 1, 0]$ from top to bottom; respectively red straight line for (a) $p = 1, q = 1$, black dash line for (b) $p = 1, q = 0.7$, blue straight line for (c) $p = 0.7, q = 1$ and green dash line for (d) $p = 0.7, q = 0.7$.

introducing ℓ_p norm $(0 < p < 1)$ regularization makes the regularization nonconvex and hard to solve, regarding to this, we propose to use the Coordinate Descent Algorithm (CDA) with weighted ℓ_1 norm and the Proximal Operator (PO) to solve the optimization problem with the new regularization.

The remainder of this chapter is organized as follows. In section 5.2, problem formulation and details of proposed algorithms are shown. we have shown how to introduce ℓ_p norm $(0 < p < 1)$ into the regularization part of IILasso and how it may have effect on the contours of regularization. We also show that the optimization problem with the new regularization can be solved both by CDA and PO to find a local optimum. Subsequently, we give experimental validations of proposed algorithms in section 5.3. Synthetic data experiments (section 5.3.1) present a performance comparison with different algorithms in terms of the relative norm error and support error. Real-world data experiments are shown in section 5.3.2 using highly correlated gene expression data. We present that our proposed algorithm can obtain smaller misclassification error in different gene expression datasets.

5.2 Problem Formulation

The paper considers a sparse coding problem, in which we are finding a proper approach to obtain an optimal sparse solution $\mathbf{z} \in \mathbb{R}^n$ from a given noisy data $\mathbf{x} \in \mathbb{R}^m$ based on the linear signal model described in the following equation,

$$\mathbf{x} = \mathbf{D}\mathbf{z} + \mathbf{v}, \tag{5.1}$$

where $\mathbf{D} \in \mathbb{R}^{m \times n}$ is an overcomplete dictionary matrix with $n > m$, and $\mathbf{v} \in \mathbb{R}^m$ is an additive white Gaussian distributed noise vector. The above equation (5.1) defines an underdetermined linear system. As the dictionary in the equation is supposed to be a full row-rank matrix, this model should have infinite solutions. To achieve the required sparse solution, sparse constraints are introduced. Therefore, the general minimization model with the square data fitting error and a sparse constraints are applied to solve the linear signal model,

$$\min_{\mathbf{z}} L(\mathbf{z}) = \frac{1}{2}\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda d_p(\mathbf{z}) + \gamma h_q(\mathbf{z}), \tag{5.2}$$

where $\{\lambda \geq 0, \gamma \geq 0, \lambda + \gamma \neq 0\}$ are tuning parameters to adjust the effect of the sparse constraints. The function $d_p(\mathbf{z})$ is a sparse penalty term formulated as follow,

$$d_p(\mathbf{z}) = \|\mathbf{z}\|_p^p = \sum_{k=1}^{n} |\mathbf{z}_k|^p. \tag{5.3}$$

Furthermore, $h_q(\mathbf{z})$ which introduces the coherence between dictionary columns formulated as follows,

$$h_q(\mathbf{z}) = \frac{1}{2}(|\mathbf{z}|^q)^{\mathrm{T}}\mathbf{R}|\mathbf{z}|^q = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \mathbf{R}_{jk}|\mathbf{z}_j|^q|\mathbf{z}_k|^q. \tag{5.4}$$

In the equation (5.4), $\mathbf{R} \in \mathbb{R}^{n \times n}$ is a symmetric matrix whose component $\mathbf{R}_{jk} \geq 0$ represents for the coherence between dictionary columns $\mathbf{D}_j$ and $\mathbf{D}_k$, the definition of $\mathbf{R}$ is shown as follow,

$$\mathbf{R}_{jk} = \begin{cases} |\mathbf{D}_j^{\mathrm{T}}\mathbf{D}_k| & ,j \neq k \\ 0 & ,j = k. \end{cases} \tag{5.5}$$

In the original IILasso, the regularization has $p = 1$ and $q = 1$. By reducing p and q value, contours graph of the constraint will be more concave. In Figure 5.1, we can observe the effect

of changing p and q value. In general, high value in R representing high coherence, both smaller p and smaller q value can modify contours to be more concave. In the first row where dictionary columns are orthogonal , $h_q(\mathbf{z})$ has no effect on the contours and they are the same as original ℓ_p norm contours. With coherence increasing, all settings of p and q value gradually result in more concave contours, indicating effects of $h_q(\mathbf{z})$ is strengthening. Furthermore, reducing p and q simultaneously will help changing the contours to be more concave and reducing p value is more effective than reducing q value. The last column of contours shows a more intuitive comparison between different settings and we can see the effects of changing value of p, q and $\mathbf{R}$ more clearly. These contours suggest that properly setting p and q value can help getting sparser and thus more accurate results. However, reducing p and q value also means the loss function are no longer convex and hard to solve. Therefore, we propose to use CDA with weighted ℓ_1 norm and the proximal splitting method to solve the problem in the following sections.

5.2.1 IILasso

IILasso [40] firstly introduce the coherence between dictionary columns into the sparse constraint function, and use ℓ_1 norm sparse constraint $d_1(\mathbf{z})$, and $h_1(\mathbf{z})$ in the objective function (5.2).

To solve (5.2), CDA, which was originally proposed for Lasso [23,24], is applied. CDA optimize the objective function by updating $\mathbf{z}$ element-wisely. Therefore, the optimization problem is changed to the following one,

$$\min_{\mathbf{z}_i} L(\mathbf{z}_i) = \frac{1}{2}(\hat{\mathbf{x}}_i - \mathbf{D}_{:i}\mathbf{z}_i)^{\mathrm{T}}(\hat{\mathbf{x}}_i - \mathbf{D}_{:i}\mathbf{z}_i) + \lambda|\mathbf{z}_i| + \gamma|\mathbf{z}_i|\mathbf{R}_{i:}|\mathbf{z}|, \tag{5.6}$$

where $\mathbf{D}_{:i}$ is ith column of the dictionary $\mathbf{D}$, $\mathbf{z}_i$ is ith element of the coefficient vector $\mathbf{z}$, $\hat{\mathbf{x}}_i = \mathbf{x} - \sum_{j=1,j\neq i}^{n} \mathbf{d}_j\mathbf{z}_j)$ and $\mathbf{R}_{i:}$ is ith row of $\mathbf{R}$. Therefore, the update equation can be easily derived from the differential of the reformed objective function $L(\mathbf{z}_i)$,

$$\partial_{\mathbf{z}_i} L(\mathbf{z}_i) = -\mathbf{D}_{:i}{}^{\mathrm{T}}(\hat{\mathbf{x}}_i - \mathbf{D}_{:i}\mathbf{z}_i) + (\lambda + \gamma\mathbf{R}_{i:}|\mathbf{z}|)\mathrm{sign}(\mathbf{z}_i). \tag{5.7}$$

Because of the definition of $\mathbf{R}$ that $\mathbf{R}_{ii} = 0$, $\mathbf{z}_i$ does not affect $\mathbf{R}_{i:}|\mathbf{z}|$. Eventually, we can obtain the update rule by solving $\partial_{\mathbf{z}_i} L(\mathbf{z}_i) = 0$,

$$\mathbf{z}_i \leftarrow \frac{1}{\mathbf{D}_{:i}{}^{\mathrm{T}}\mathbf{D}_{:i}}\pi_1(\mathbf{D}_{:i}{}^{\mathrm{T}}\hat{\mathbf{x}}_i, (\lambda + \gamma\mathbf{R}_{i:}|\mathbf{z}|)), \tag{5.8}$$

where $\pi_1(\mathbf{z}, \mathbf{t})$ is a soft thresholding operator defined in equation (5.9). The detail of IILasso is summarized in Algorithm 5.1.

$$[\pi_1(\mathbf{z}, \mathbf{t})]_j = \text{sgn}(\mathbf{z}_j) \max\{|\mathbf{z}_j| - t_j, 0\} \tag{5.9}$$

Algorithm 5.1: IILasso

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper positive parameters λ and γ.
Initialization: $\mathbf{z}^{(0)} = \mathbf{0}$, $k = 0$.
Main iteration: increment k by 1
for $i = 1, ..., n$
$\mathbf{z}_i^{(k)} \leftarrow \frac{1}{\mathbf{D}_{:i}{}^{\mathrm{T}}\mathbf{D}_{:i}} \pi_1(\mathbf{D}_{:i}{}^{\mathrm{T}}\hat{\mathbf{x}}_i, (\lambda + \gamma \mathbf{R}_{i:}|\mathbf{z}|))$,
end for
Stopping rule: stop if $\mathbf{z}^{(k)}$ has converged
Output: $\mathbf{z} = \mathbf{z}^{(k)}$

5.2.2 II-ISTA

Independently Interpretable Iterative Shrinkage Thresholding Algorithm (II-ISTA) has the same objective function as IILasso, namely $d_1(\mathbf{z})$ and $h_1(\mathbf{z})$. The only difference is II-ISTA use the proximal splitting method to solve the objective function.

The proximal splitting method use constant step size to pursuit the minimum of objective function, with circular iterations,

$$\mathbf{z}^{(k)} = \pi_1(\mathbf{z}^{(k-1)} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{z}^{(k-1)} - \mathbf{x}), \mathbf{t}), \tag{5.10}$$

where $\pi_1(\mathbf{z}, \mathbf{t}) = \arg\min_{\mathbf{u}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{u} - \mathbf{z}\|_2^2 + \lambda d_1(\mathbf{z}) + \gamma h_1(\mathbf{z})$ denotes to use soft thresholding operator as the proximal operator to solve the regularization. Reviewing the regularization part, since $\mathbf{R}_{ii} = 0$,

$$\lambda d_1(\mathbf{z}) + \gamma h_1(\mathbf{z}) = \sum_{i=1}^{n} |\mathbf{z}_i|(\lambda + \gamma \sum_{j=1, j\neq i}^{n} |\mathbf{z}_j|\mathbf{R}_{ij}). \tag{5.11}$$

Therefore, the regularization can regarded as a weight on ℓ_1 norm during optimization and the ith value of threshold $\mathbf{t}$ is defined in equation (5.12). The update rules of II-ISTA are summarized in Algorithm 5.2.

$$\mathbf{t}_i = \lambda + \gamma \sum_{j=1, j\neq i}^{n} |\mathbf{z}_j| \mathbf{R}_{ij}. \tag{5.12}$$

Algorithm 5.2: II-ISTA

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper positive parameters λ, γ and α.
Restriction: $\alpha >$ largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$
Initialization: $\mathbf{z}^{(0)} = \mathbf{0}$, $k = 0$.
Main iteration: increment k by 1
$\mathbf{t} = \lambda + \gamma \mathbf{R}|\mathbf{z}^{(k-1)}|$
$\mathbf{z}^{(k)} = \pi_1(\mathbf{z}^{(k-1)} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{z}^{(k-1)} - \mathbf{x}), \mathbf{t})$
Stopping rule: stop if $\mathbf{z}^{(k)}$ has converged
Output: $\mathbf{z} = \mathbf{z}^{(k)}$

5.2.3 IIWLasso

For Independently Interpretable Weighted Lasso (IIWLasso), we have $p \in (0, 1]$ and $q \in (0, 1]$ that ℓ_p norms are introduced in the regularization which makes the optimization non-convex but more sparsity-pursuing. The word 'weighted' refers to the idea to restrain the ℓ_1 sparsity constraint with information from previous iteration which can function approximately as an ℓ_p norm, namely, the regularization part is formulated as,

$$d_p(\mathbf{z}) = \|\mathbf{z}\|_p^p \approx \sum_{i}^{n} |\mathbf{z}_i^{(k-1)}|^{p-1} |\mathbf{z}_i|, \tag{5.13}$$

$$h_q(\mathbf{z}) = \frac{1}{2} \sum_{i}^{n} \sum_{j=1, j\neq i}^{n} |\mathbf{z}_i^{(k-1)}|^{q-1} |\mathbf{z}_j^{(k-1)}|^{q-1} |\mathbf{z}_i| |\mathbf{z}_j| \mathbf{R}_{ij}, \tag{5.14}$$

where k refers to iteration number. Since we can consider the component-wise weighted part from the previous iteration as constant during iterations, the sparse constraint is still an ℓ_1 norm structure in each iteration and we can still use CDA to obtain the update rule by solving $\partial_{\mathbf{z}_i} L(\mathbf{z}_i) = 0$ in equation (5.15) and (5.16). The update rules of IIWLasso can be summarized as Algorithm 5.3.

$$\begin{aligned} \partial_{\mathbf{z}_i} L(\mathbf{z}_i) = &- \mathbf{D}_{:i}{}^{\mathrm{T}}(\hat{\mathbf{x}}_i - \mathbf{D}_{:i}\mathbf{z}_i) \\ &+ (\lambda |\mathbf{z}_i^{(k-1)}|^{p-1} + \gamma |\mathbf{z}_i^{(k-1)}|^{q-1} \mathbf{R}_{i:} |\mathbf{z}|^q) \mathrm{sign}(\mathbf{z}_i). \end{aligned} \tag{5.15}$$

$$\mathbf{z}_i \leftarrow \frac{1}{\mathbf{D}_{:i}{}^{\mathrm{T}}\mathbf{D}_{:i}} \pi_1(\mathbf{D}_{:i}{}^{\mathrm{T}}\hat{\mathbf{x}}_i, \lambda |\mathbf{z}_i^{(k-1)}|^{p-1} + \gamma |\mathbf{z}_i^{(k-1)}|^{q-1} \mathbf{R}_{i:} |\mathbf{z}|^q), \tag{5.16}$$

Algorithm 5.3: IIWLasso

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper positive parameters λ and γ.
Initialization: $\mathbf{z}^{(0)} = \mathbf{0}$, $k = 0$.
Main iteration: increment k by 1
for $i = 1, ..., n$
$\mathbf{z}_i^{(k)} \leftarrow \frac{1}{\mathbf{D}_{:i}{}^{\mathrm{T}}\mathbf{D}_{:i}} \pi_1(\mathbf{D}_{:i}{}^{\mathrm{T}}\hat{\mathbf{x}}_i, \lambda|\mathbf{z}_i^{(k-1)}|^{p-1} + \gamma|\mathbf{z}_i^{(k-1)}|^{q-1}\mathbf{R}_{i:}|\mathbf{z}|^q)$,
end for
Stopping rule: stop if $\mathbf{z}^{(k)}$ has converged
Output: $\mathbf{z} = \mathbf{z}^{(k)}$

5.2.4 Independently Interpretable Proximal Operator (IIPO)

In this section, we intend to solve a condition that $p = q = \frac{2}{3}$ using the proximal operator inspired from the closed-form thresholding formulas $\ell_{\frac{2}{3}}$ regularization [28,29]. By applying the proximal splitting method, we pursuit the minimum of the objective function by repeating

$$\mathbf{z}^{(k)} = \pi_p(\mathbf{z}^{(k-1)} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{z}^{(k-1)} - \mathbf{x}), \mathbf{t}), \tag{5.17}$$

where $\pi_p(\mathbf{z}, \mathbf{t}) = \arg\min_{\mathbf{u}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{u} - \mathbf{z}\|_2^2 + \lambda d_p(\mathbf{z}) + \gamma h_p(\mathbf{z})$. Following the same derivation process in II-ISTA, since we known $\mathbf{R}_{ii} = 0$,

$$\lambda d_p(\mathbf{z}) + \gamma h_p(\mathbf{z}, \mathbf{D}) = \sum_{i=1}^{n} |\mathbf{z}_i|^p(\lambda + \gamma \sum_{j=1, j\neq i}^{n} |\mathbf{z}_j|^p \mathbf{R}_{ij}). \tag{5.18}$$

Therefore, the regularization can regarded as a weight on ℓ_p norm during optimization and the threshold $\mathbf{t}$ is defined as follow,

$$\mathbf{t}_i = \lambda + \gamma \sum_{j=1, j\neq i}^{n} |\mathbf{z}_j|^p \mathbf{R}_{ij}, i = 1, ..., n. \tag{5.19}$$

The reason of choosing $p = q = \frac{2}{3}$ is that the proximal operator π_p for $p = \frac{l}{s}$ is obtained by solving the $2s - l$ degree polynomial equation [28],

$$x^{2s-l} - zx^{s-l} + \frac{l}{s}u = 0, x > 0. \tag{5.20}$$

Since then, other from $p \in \{\frac{1}{2}, \frac{2}{3}\}$, the degree of the polynomial is higher than 4 and it is hard to have a closed-form formula, however, we can still obtain approximate solution of

equation (5.20) by using the Newton method [28]. Because the $\ell_{\frac{2}{3}}$ norm regularization performs better than the $\ell_{\frac{1}{2}}$ norm regularization, we only present $p = \frac{2}{3}$ in this paper. $\ell_{\frac{2}{3}}$ norm is a good regularization for achieving a sparser representation more efficiently, meanwhile its converged result can be obtained when λ is small enough and dictionary D satisfies a certain concentration assumption [29, 71]. The closed-form formulas for the proximal operators $\pi_{\frac{2}{3}}(\mathbf{z}, \mathbf{t})$ are given in equation (5.21) and (5.22). The update rules of II2/3PO with $\ell_{\frac{2}{3}}$ norm based regularization are summarized in Algorithm 5.4.

$$\begin{aligned}\left[\pi_{\frac{2}{3}}(\mathbf{z}, \mathbf{t})\right]_j =& \frac{1}{8}(\sqrt{2\mathbf{v}_j} + \sqrt{\frac{2|\mathbf{z}_j|}{\sqrt{2\mathbf{v}_j}} - 2\mathbf{v}_j})^3 \\ & \operatorname{sign}(\mathbf{z}_j \max\{|\mathbf{z}_j| - 2(\frac{2}{3}\mathbf{t}_j)^{\frac{3}{4}}, 0\}),\end{aligned} \tag{5.21}$$

where

$$\mathbf{v}_j = (\frac{\mathbf{z}_j^2}{16} + \sqrt{\frac{\mathbf{z}_j^4}{256} - \frac{8\mathbf{t}_j^3}{729}})^{\frac{1}{3}} + (\frac{\mathbf{z}_j^2}{16} - \sqrt{\frac{\mathbf{z}_j^4}{256} - \frac{8\mathbf{t}_j^3}{729}})^{\frac{1}{3}}. \tag{5.22}$$

Algorithm 5.4: II2/3PO

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper positive parameters λ, γ and α.
Restriction: $\alpha >$ largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$
Initialization: $\mathbf{z}^{(0)} = \mathbf{0}$, $k = 0$.
Main iteration: increment k by 1
$\mathbf{t} = \lambda + \gamma\mathbf{R}|\mathbf{z}^{(k-1)}|^p$
$\mathbf{z}^{(k)} = \pi_{\frac{2}{3}}(\mathbf{z}^{(k-1)} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{z}^{(k-1)} - \mathbf{x}), \mathbf{t})$
Stopping rule: stop if $\mathbf{z}^{(k)}$ has converged
Output: $\mathbf{z} = \mathbf{z}^{(k)}$

5.3 Experiments

5.3.1 Synthetic data experiments

In this section, we present our experiments results of proposed algorithms in finding an optimal solution to the ground truth of sparse representations. Synthetically generated data were used throughout experiments, which was built by using ground true dictionaries generated with Gaussian random values. All experiments were performed via Matlab R2018b, and programs were run on a PC with a 2.7 GHz Intel core and 12GB RAM.

In the synthetic data experiments, to valid the effects of regularization in highly correlated Data, we formed three dictionaries $\mathbf{D}_{\mathrm{c}}$ with different coherence, where the subscript c stands for mean coherence used in experiments. All dictionaries were sized as 30×50 matrices and column

normalized after generating value. $\mathbf{D}_{0.15}$ was generated by drawing value randomly from the normal distribution $N(0,1)$. The Gaussian random dictionary usually have a mean coherence around $(0.10, 0.20)$, and in this experiments its mean coherence is 0.15. The coherence of two vectors with the same scale is defined as follow,

$$\text{coherence}(\mathbf{D}_j, \mathbf{D}_k) = |\frac{\mathbf{D}_j^{\mathrm{T}}\mathbf{D}_k}{\|\mathbf{D}_j\|\|\mathbf{D}_k\|}|. \tag{5.23}$$

The other two high coherence dictionaries were generated by adding a small normal distributed vector to a baseline vector generated from the normal distribution $N(0,1)$, namely $\mathbf{D}_{0.50}$ with mean coherence 0.50 and $\mathbf{D}_{0.80}$ with mean coherence 0.80. The coherence distribution of the three dictionary is shown in Figure 5.2. The diagram counts numbers of coherence between every two columns in the dictionary, so it has a total number $(n-1)n/2$. The quantity of ground true sparse representation vectors $\mathbf{Z}_{\text{orig}} = (\mathbf{z}_{\text{orig}1}, ..., \mathbf{z}_{\text{orig}100})$ was 100 in experiments. Correspondingly, data set $\mathbf{X}_{\text{orig}}$ had 100 sample size which was generated by $\mathbf{D}_{\text{c}}$ and $\mathbf{Z}_{\text{orig}}$ based on the equation (5.1), The noise vectors $\mathbf{v}$ were added based on Gaussian random entries with 20dB SNR.

Sparsity measurement used Hoyer sparsity measure [66] based on the relationship between the ℓ_1-norm and the ℓ_2-norm, which can give a well defined sparsity. The Hoyer sparsity measure is formulated as following,

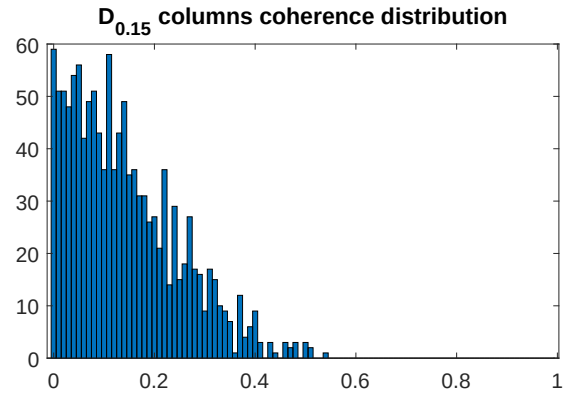
$$\text{Hoyer sparsity}(\mathbf{z}) = \frac{\sqrt{n} - (\sum |\mathbf{z}_i|)/\sqrt{\sum {\mathbf{z}_i}^2}}{\sqrt{n}-1}, \tag{5.24}$$

where n represents the dimension of $\mathbf{z}$, and when the value of the equation is closer to 1, the $\mathbf{z}$ vector is approaching to a sparser vector.

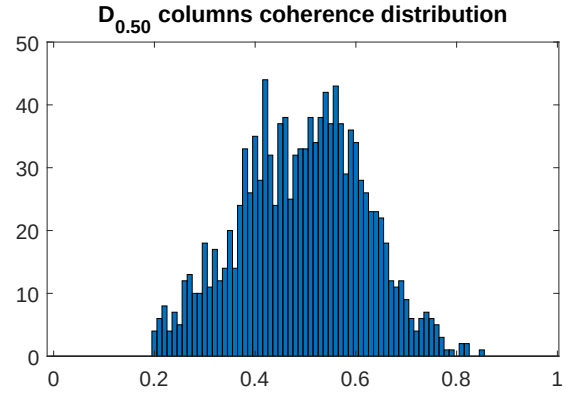
Fig.5.3 shows the ability of recovering accurate sparse representations using three sparse coding algorithms with dictionary $\mathbf{D}_{0.15}$. Sparse representation error use the relative norm error compared to $\mathbf{Z}_{\text{orig}}$, defined in equation (5.25).

$$\text{Relative norm error}(\mathbf{z}) = \frac{\|\mathbf{z}_{\text{orig}} - \mathbf{z}\|_2^2}{\|\mathbf{z}_{\text{orig}}\|_2^2}. \tag{5.25}$$

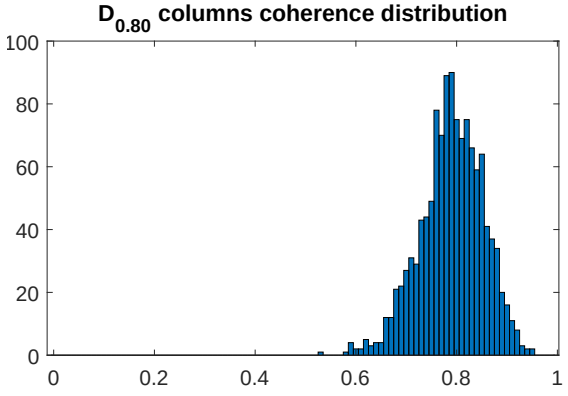
Fig.5.4 shows the performance variation in a range of λ values with dictionary $\mathbf{D}_{0.15}$. Sparse representation accuracy is compared using support error, which is defined in equations (5.26)



(a)



(b)



(c)

Figure 5.2: Coherence distribution of synthetic generated dictionary: (a) $\mathbf{D}_{0.15}$, (b) $\mathbf{D}_{0.50}$ and (c) $\mathbf{D}_{0.80}$.

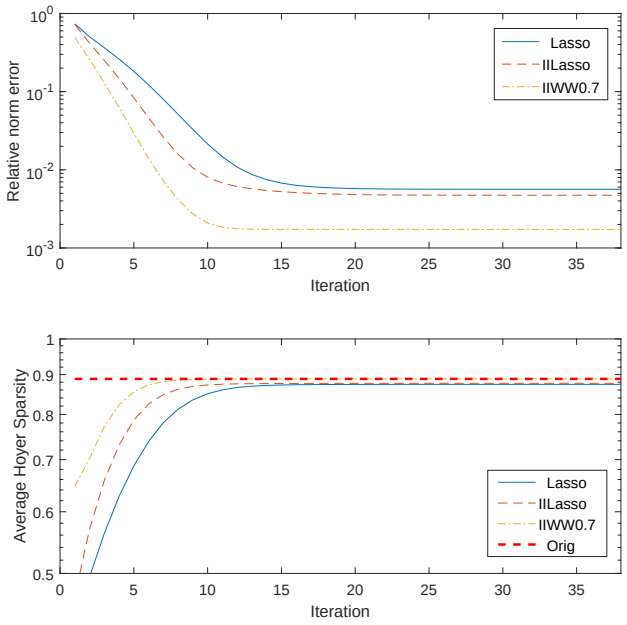


Figure 5.3: Average sparse representation errors and Hoyer sparsity convergence graph of different algorithms with Gaussian random dictionary $\mathbf{D}_{0.15}$.

and (5.27),

$$S = Support\{\mathbf{z}\}, \tag{5.26}$$

$$\text{Support error}(\mathbf{z}) = \frac{\max\{|S_{orig}|, |S|\} - |S_{orig} \cap S|}{\max\{|S_{orig}|, |S|\}}. \tag{5.27}$$

The definition of support is a set containing information of non-zero positions in $\mathbf{z}$. Therefore, support error can show how given $\mathbf{z}$ meets the ground true one considering both non-zero positions and zero positions and it is more sensitive to the accuracy of finding non-zero positions.

In the rest of this section, we intend to compare algorithms proposed in this paper, namely II-ISTA, II2/3PO and IIWLasso, to their original algorithms, namely ISTA, 2/3PO, Lasso and IILasso. In detail, we subdivided IILasso into three special cases, namely IIWL0.7 with $p = 0.7$ and $q = 1$, IIWR0.7 with $p = 1$ and $q = 0.7$, IIWW0.7 with $p = 0.7$ and $q = 0.7$. The reason of only choosing $p = 0.7$ is that weighted ℓ_1 norm tends to obtain its best performance in experiments when $p = 0.7$, which is also validated in our previous paper [34].

Performance in Gaussian random dictionary $\mathbf{D}_{0.15}$

From Figure 5.3, we can see all three algorithms can achieve convergent results with small error and high sparsity closed to the ground truth in small numbers of iteration with Gaussian

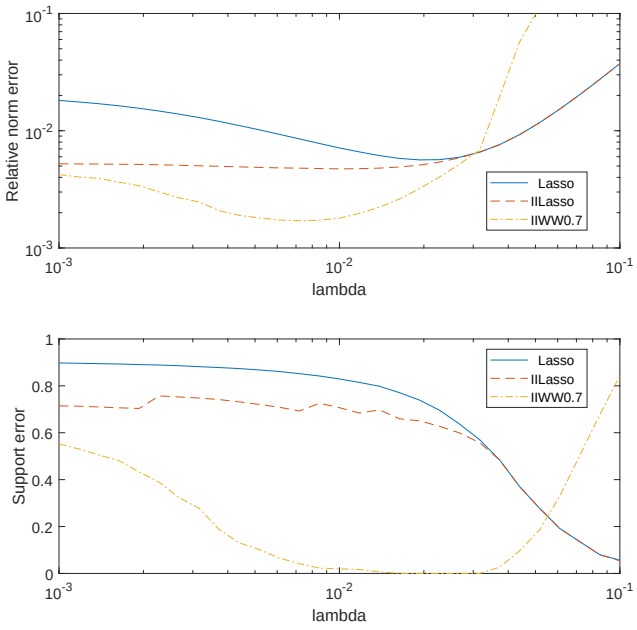


Figure 5.4: Average relative norm errors and support error of different algorithms in a range of λ with optimized γ and Gaussian random dictionary $\mathbf{D}_{0.15}$.

random dictionary $\mathbf{D}_{0.15}$. In detail, we can find IIWW0.7 can reach smallest error with smallest iterations, and its converged result can best fit the original sparsity. From Figure 5.4, we can see when ℓ_1 norm based Lasso and IILasso reach their smallest relative norm error, their support errors are usually not good which indicate that their results contain many small values in wrong positions. On the contrary, curve trends of relative norm error and support error are more similar in ℓ_p norm $(0 < p < 1)$ based IIWW0.7, indicating ℓ_p norm $(0 < p < 1)$ can obtain smaller relative norm error and support error simultaneously. Furthermore, we can notice the curve of IILasso in relative norm error tends to be straight when lambda is small, which means the coherence based regularization itself can also obtain sparse and accurate result in certain degree.

Performances of the other algorithms with $\mathbf{D}_{0.15}$ are summarized in Table 5.1. IIWR0.7 and IIWL0.7 refer to different p and q choices in Figure 5.1. IIWR0.7 represents for $p = 1$ and $q = 0.7$ and IIWL0.7 represents for $p = 0.7$ and $q = 1$. In general, we can see that all independently interpretable algorithms, namely those which add coherence related regularization $h_q(\mathbf{z})$, can obtain smaller relative norm error than their original algorithms. Except II2/3PO, the other independently interpretable algorithms also obtain smaller support error than their original algorithms. Meanwhile, we can figure out ℓ_p norm $(0 < p < 1)$ based algorithms can obtain better results both in relative norm error and support error. In detail, IIWW0.7 presents the smallest relative norm error with relatively small support error.

Figure 5.5 shows that all three algorithms can generally find accurate and sparse representa-

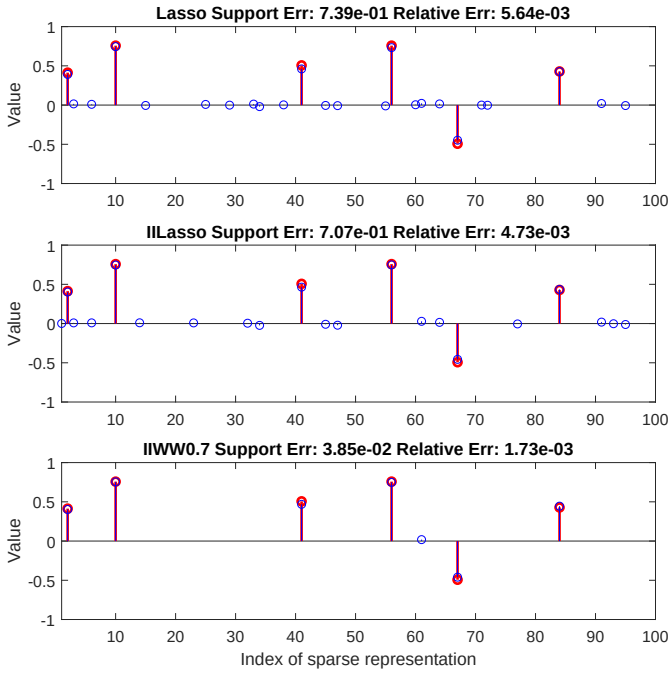


Figure 5.5: Well matched recovered sparse representation (blue) compared with original one (red) of algorithm Lasso, IILasso, IIWW0.7 in $\mathbf{D}_{0.15}$ (from up to down).

tions but having several small values in wrong positions compared to the ground true representation (red). Lasso has the largest amount of small value mistakes, so it has a larger relative norm error than the other algorithms. IILasso performs slightly better than Lasso in both relative norm error and support error, which indicates its regularization may help enhancing performance but still can hardly remove small values in wrong positions. For IIWW0.7, the number of support mistakes is only one in these two cases which results in small support error, indicating ℓ_p can help increasing accuracy of sparse representation significantly.

Figure 5.6 shows time cost of different algorithms with Gaussian random dictionary $\mathbf{D}_{0.15}$. Although IILasso and II-ISTA converge at similar relative norm error, II-ISTA converges fastest in computation time because of that PO update coefficient as a whole while CDA update element by element. We can also notice that PO is not always converge faster than CDA comparing II2/3PO to the others which is caused by complexity of the closed-form formulas for the proximal operators $\pi_{\frac{2}{3}}(\mathbf{z}, \mathbf{t})$.

Performance in relatively highly correlated dictionary $\mathbf{D}_{0.50}$

From Figure 5.7, we can see all three algorithms can achieve convergent results with small error and high sparsity in relatively small numbers of iteration with relative highly correlated

Table 5.1: Average relative norm errors and corresponded support error of different algorithms with optimized parameters and Gaussian random dictionary $\mathbf{D}_{0.15}$.

Algorithm	Relative Norm Error	Support Error
Lasso	5.64×10^{-3}	7.39×10^{-1}
IILasso	4.73×10^{-3}	7.07×10^{-1}
IIWR0.7	2.35×10^{-3}	2.68×10^{-1}
IIWL0.7	1.84×10^{-3}	3.85×10^{-2}
IIWW0.7	$\mathbf{1.65 \times 10^{-3}}$	4.15×10^{-2}
ISTA	5.71×10^{-3}	7.41×10^{-1}
II-ISTA	4.79×10^{-3}	7.02×10^{-1}
2/3PO	1.79×10^{-3}	$\mathbf{3.23 \times 10^{-2}}$
II2/3PO	1.69×10^{-3}	4.15×10^{-2}

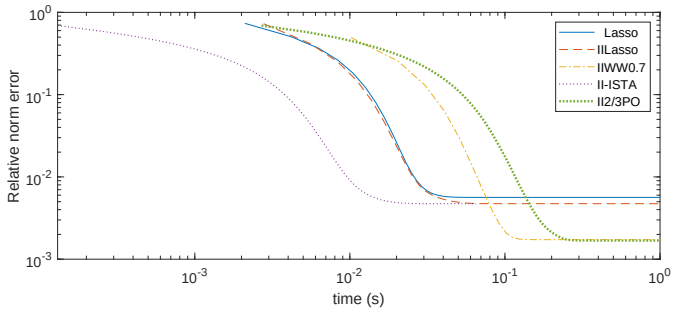


Figure 5.6: Time cost of different algorithms with Gaussian random dictionary $\mathbf{D}_{0.15}$.

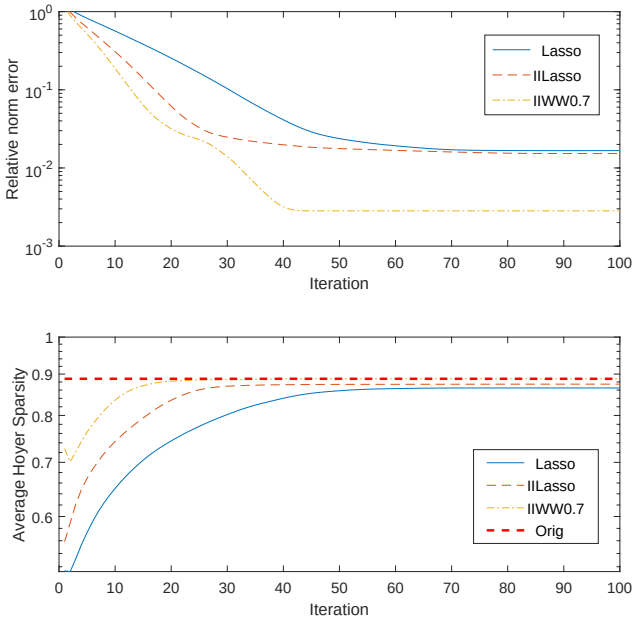


Figure 5.7: Average sparse representation errors and Hoyer sparsity convergence graph of different algorithms with highly correlated dictionary $\mathbf{D}_{0.50}$.

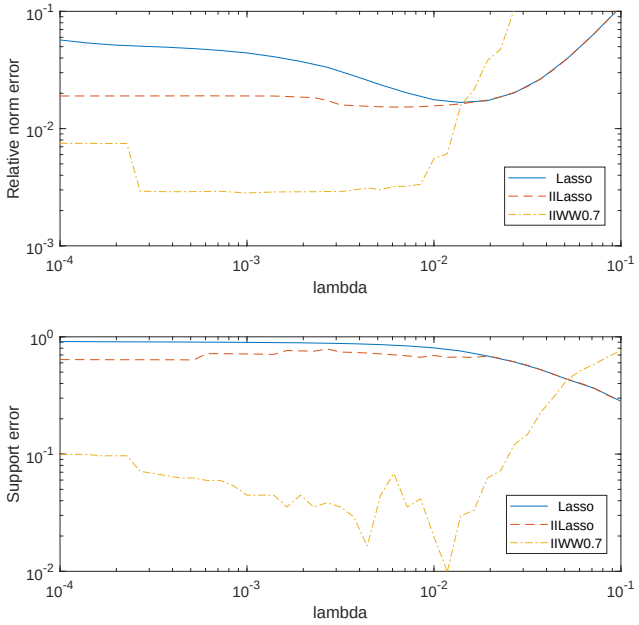


Figure 5.8: Average relative norm errors and support error of different algorithms in a range of λ with optimized γ and relatively highly correlated dictionary $\mathbf{D}_{0.50}$.

Table 5.2: Average relative norm errors and corresponded support error of different algorithms with optimized parameters and relatively highly correlated dictionary $\mathbf{D}_{0.50}$

Algorithm	Relative Norm Error	Support Error
Lasso	1.67×10^{-2}	7.57×10^{-1}
IILasso	1.53×10^{-2}	7.01×10^{-1}
IIWR0.7	3.22×10^{-3}	7.69×10^{-1}
IIWL0.7	3.30×10^{-3}	7.41×10^{-2}
IIWW0.7	$\mathbf{2.83 \times 10^{-3}}$	$\mathbf{4.46 \times 10^{-2}}$
ISTA	2.47×10^{-2}	7.08×10^{-1}
II-ISTA	2.46×10^{-2}	6.81×10^{-1}
2/3PO	1.33×10^{-2}	1.54×10^{-1}
II2/3PO	1.32×10^{-2}	1.42×10^{-1}

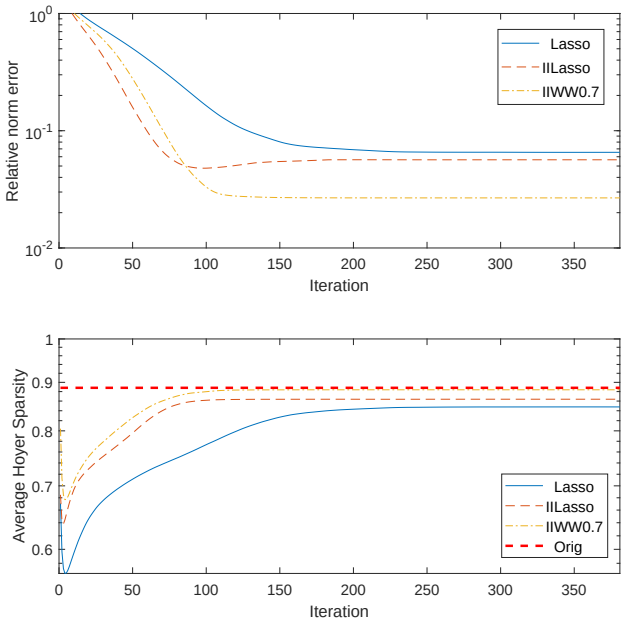


Figure 5.9: Average sparse representation errors and Hoyer sparsity convergence graph of different algorithms with highly correlated dictionary $\mathbf{D}_{0.80}$.

dictionary $\mathbf{D}_{0.50}$. Iteration numbers for reaching convergence of all algorithms have increased compared to $\mathbf{D}_{0.15}$. In detail, we can find that IIWW0.7 can still reach smallest relative norm error, and its converged result can also fit the original sparsity best. From Figure 5.8, we can see the similar trend of Lasso and IILasso about the dislocation between relative norm error and support error. Furthermore, we can observe a larger optimal lambda range for IIWW0.7 in $\mathbf{D}_{50}$ than that in $\mathbf{D}_{15}$, which may be caused that effect of coherence based regularization $h_q(\mathbf{z})$ is strengthening with increase of coherence.

Performances of the other algorithms are summarized with $\mathbf{D}_{0.50}$ in Table 5.2. In this case, we can still observe the trend that all independently interpretable algorithms can obtain smaller relative norm error and support error than their original algorithms, while IIWW0.7 presents both the smallest relative norm error and support error.

Performance in highly correlated dictionary $\mathbf{D}_{0.80}$

From Figure 5.9, we can also see that all three algorithms can achieve convergent results with small error and high sparsity in reasonable numbers of iteration with highly correlated dictionary $\mathbf{D}_{0.80}$. Iteration numbers for reaching convergence of all algorithms is nearly 10 times compared to $\mathbf{D}_{0.15}$. In detail, we can find that IIWW0.7 can still reach smallest relative norm error, and its converged result also have the relatively best performance to fit the original sparsity. Furthermore, although IILasso can obtain smaller relative norm error compared to

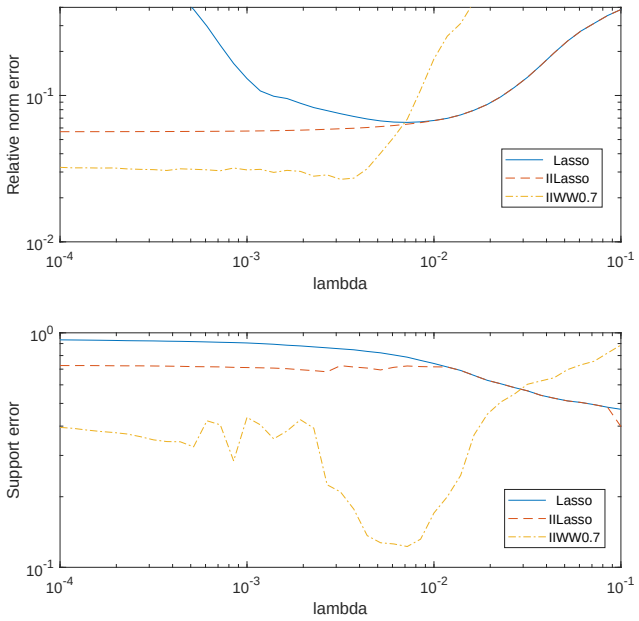


Figure 5.10: Average relative norm errors and support error of different algorithms in a range of λ with optimized γ and highly correlated dictionary $\mathbf{D}_{0.80}$.

Table 5.3: Average relative norm errors and corresponded support error of different algorithms with optimized parameters and highly correlated dictionary $\mathbf{D}_{0.80}$

Algorithm	Relative Norm Error	Support Error
Lasso	6.54×10^{-2}	7.87×10^{-1}
IILasso	5.65×10^{-2}	7.26×10^{-1}
IIWR0.7	2.79×10^{-2}	2.20×10^{-1}
IIWL0.7	2.70×10^{-2}	3.30×10^{-1}
IIWW0.7	$\mathbf{2.68 \times 10^{-2}}$	$\mathbf{2.05 \times 10^{-1}}$
ISTA	1.38×10^{-1}	7.50×10^{-1}
II-ISTA	1.12×10^{-1}	6.42×10^{-1}
2/3PO	1.40×10^{-1}	6.62×10^{-1}
II2/3PO	1.31×10^{-1}	4.23×10^{-1}

Table 5.4: abstract of Gene Expression Datasets

dataset name	sample number	dimension	label number	mean coherence
alon	62	2000	2	0.80
shipp	77	7129	2	0.88
gravier	168	2905	2	0.41
christensen	217	1413	3	0.95

Lasso, its convergence curve has a strange rebound in relative norm error after 100 iterations which may indicate that in highly correlated data, the strategy of choosing uncorrelated data may also result in certain inappropriate updating consequence.

From Figure 5.10, we can see the similar trend of Lasso and IILasso about the dislocation between relative norm error and support error. Furthermore, we can observe the relative norm error of IILasso is getting smaller gradually with smaller lambda and we found that when lambda is 0, IILasso may obtain the smallest relative norm error in this case, which may indicate that only $h_1(\mathbf{z})$ itself can function as a good sparse regularization in highly correlated data and $d_1(\mathbf{z})$ may result in obstruction.

Performances of the other algorithms are summarized with $\mathbf{D}_{0.80}$ in Table 5.3. In this case, we can still observe the trend that all independently interpretable algorithms can obtain smaller relative norm error and support error than their original algorithms, and again IIWW0.7 presents both the smallest relative norm error and support error.

Performance comparison among different coherence dictionaries

A more intuitive comparison of different independently interpretable algorithms in different coherence conditions is shown in Figure 5.11. In general, we can figure that all algorithms experience increase in relative norm error with increase in coherence of dictionary. Moreover, all cases of IIWLasso, namely IIWR0.7, IIWL0.7 and IIWW0.7, can obtain better performances both in relative norm error and support error compared with IILasso. In detail, IIWW0.7 can obtain smallest relative norm error in all coherence conditions, and IIWW0.7 also obtain smallest support error among all algorithms with coherence increase. Furthermore, we notice performance of II-ISTA and II2/3PO falls quickly with the increase in coherence which indicates PO may not be suitable for high coherence condition.

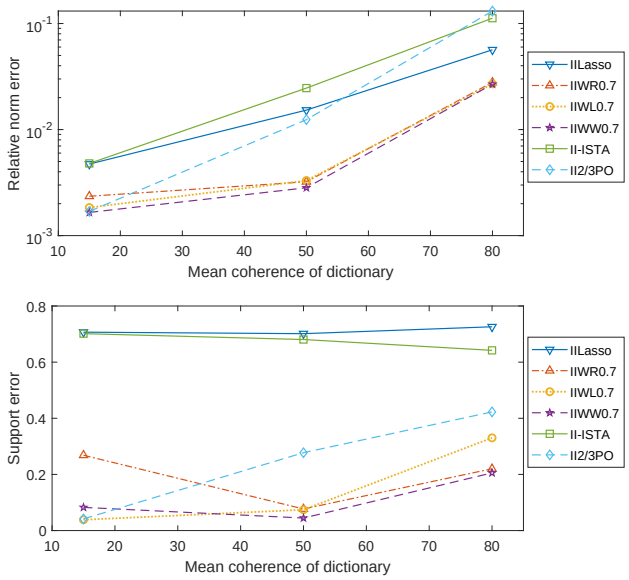


Figure 5.11: Average relative norm errors and support error of different algorithms with optimized λ and γ in differently correlated dictionaries.

5.3.2 Gene Expression Data experiments

In this subsection, we intend to present the performance of our proposed algorithms in real-world data. We applied out methods in various gene expression datasets, which are highly 'alon' [74] (colon cancer), 'shipp' [75] (lymphoma), 'gravier' [76] (breast cancer) and 'christensen' [77] (Tissue-Specific DNA Methylation). All these datasets are provided by R package datamicroarray. Detail of these datasets are shown in Table 5.4. All datasets are small-sample high-dimensional and highly correlated DNA microarray data. We use 10-fold cross-validation and softmax classifier in this experiment to evaluate misclassification errors of different algorithms.

Gene expression data is usually obtained from oligonucleotide arrays which consist of collections of microscopic single-stranded DNA sequences. An oligonucleotide array can measure the expression levels of large numbers of genes simultaneously. Properly processed gene expression data can stand for effects of certain treatments, diseases, and developmental stages on gene expression. However, gene expression data is usually highly correlated and hard to be interpreted. Obtaining small misclassification errors in this experiment can indicate that corresponded algorithms can interpret the "decomposability" of datasets and thus give suggestions about new gene expression data on treatments, diseases, and developmental stages on gene expression with appropriate database.

From Figure 5.12, we can see that tendencies of misclassification errors between different

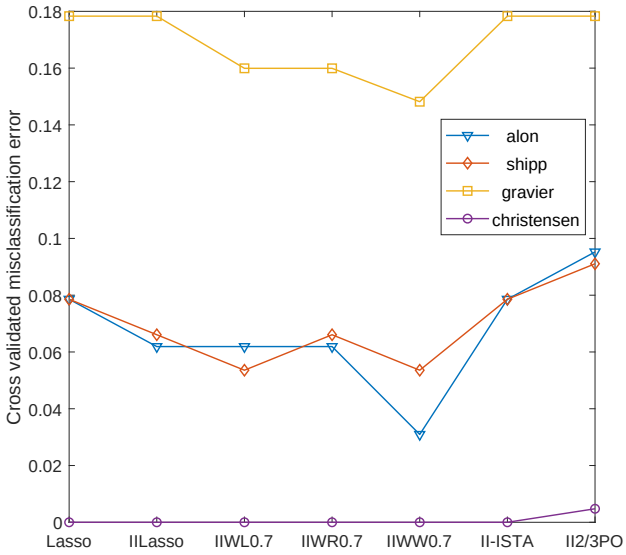


Figure 5.12: 10-fold cross validated Misclassification error of different algorithms in different gene expression datasets.

datasets are similar. IIWW0.7 always obtain the smallest or at least the same misclassification error compared with the other algorithms. IIWL0.7 and IIWR0.7 tend to have similar performance in these experiments and perform better than or equivalent to IILasso. This experiments also indicate that PO is not quite suitable for high coherence condition. IIWW0.7 performs well in the dataset ”shipp”, ”christensen” and ”alon” for classification in cross-validation with misclassification error smaller than 0.06, whereas slightly weakens in ”gravier”, which indicates that IIWW0.7 can present reasonable suggestions for certain diseases and developmental stages on gene expression.

5.3.3 Discussion

Throughout the conducted experiments, there are two points for discussion.

1. *Field of using independently interpretable regularization is not restricted in highly correlated data*

   Throughout synthetic experiments in dictionaries with different coherence conditions, we can notice algorithms with independently interpretable regularization all perform better than their original algorithms, not restricted in highly correlated data. These results indicate that the strategy of selecting uncorrelated variables is effective in enhancing the performance of algorithms in different coherence conditions. Further researches may

consider using independently interpretable regularization not only in highly correlated data.

2. *Advantage of II-ISTA*

   In both highly correlated synthetic data and gene expression data, we can notice PO with independently interpretable regularization can not function well, but II-ISTA still has the advantage in smaller computation time while obtaining similar relative norm error compared with IILasso in Gaussian random synthetic dictionary. We may suggest to use II-ISTA where time cost is more essential.

5.4 Chapter Summary

In this chapter, we have proposed an approach that introduces ℓ_p norm $(0 < p < 1)$ into the regularization of IILasso to form a new regularization for sparse coding with highly correlated data. The original regularization of IILasso which introduces the coherence term has shown good performance in highly correlated data but it has the same problem as the other ℓ_1 norm based regularization that its result is not sparse and accurate enough. Through introducing ℓ_p norm $(0 < p < 1)$, the regularization can be more sparsity-pursuing while fitting highly correlated data. We use CDA and PO to solve the optimization problem with the new regularization. In CDA, we use weighted ℓ_1 norm to approximate ℓ_p norm sparse coding problem by using information of sparse representation from the previous iteration. The 'weighted' idea makes it possible to enjoy advantages of ℓ_p norm sparse constraint while keeping the convex property. In PO, we applied the PO of $\ell_{\frac{2}{3}}$ norm regularization in the new regularization.

In synthetic data experiments, three dictionaries were generated with mean coherence 0.15, 0.50 and 0.80. We have shown that IIWW0.7 can outperform the other algorithms both in relative norm error and support error in different coherence condition. Furthermore, how different coherence condition may have effect on the results of different algorithms is shown in synthetic experiments.

We then applied proposed algorithms in gene expression datasets, which are small-sample high-dimensional and highly correlated. 10-fold cross-validation results have shown that IIWW0.7 can obtain the relatively best performance for misclassification errors among all algorithms in different datasets, which suggests that IIWW0.7 can present reasonable suggestions for diseases and developmental stages on gene expression.

Chapter 6

Deep Neural Network Structured Sparse Coding for Highly Correlated Data

6.1 Introduction

In this chapter, we still concentrate on processing highly correlated data with the strategy of selecting uncorrelated variables. We show that it is possible to construct DNN-SC versions of proposed algorithms in the previous chapter to further enhance efficiency of those algorithms. Following similar methodology in Chapter 4, we can build and train an effective encoder for highly correlated data from independently interpretable algorithms in Chapter 5.

The outline of this chapter is as follows. We state the problem formulation and present how to build encoders and train parameters in section 6.2, We then validate performance of proposed algorithms in section 6.3. Finally, chapter summary are drawn in Section 6.4.

6.2 Problem Formulation

This chapter considers a sparse coding problem, in which we are finding a proper approach to obtain an optimal sparse solution $\mathbf{z} \in \mathbb{R}^n$ from a given noisy data $\mathbf{x} \in \mathbb{R}^m$ based on the linear signal model described in the following equation,

$$\mathbf{x} = \mathbf{D}\mathbf{z} + \mathbf{v}, \tag{6.1}$$

where $\mathbf{D} \in \mathbb{R}^{m \times n}$ is an overcomplete dictionary matrix with $n > m$, and $\mathbf{v} \in \mathbb{R}^m$ is an additive white Gaussian distributed noise vector. The above equation (6.1) defines an underdetermined linear system. As the dictionary in the equation is supposed to be a full row-rank matrix, this model should have infinite solutions. To achieve the required sparse solution, sparse constraints are introduced. Therefore, the general minimization model with the square data fitting error and a sparse constraints are applied to solve the linear signal model,

$$\min_{\mathbf{z}} L(\mathbf{z}) = \frac{1}{2}\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda d_p(\mathbf{z}) + \gamma h_q(\mathbf{z}), \tag{6.2}$$

where $\{\lambda \geq 0, \gamma \geq 0, \lambda + \gamma \neq 0\}$ are tuning parameters to adjust the effect of the sparse constraints. The function $d_p(\mathbf{z})$ is a sparse penalty term formulated as follow,

$$d_p(\mathbf{z}) = \|\mathbf{z}\|_p^p = \sum_{k=1}^{n} |\mathbf{z}_k|^p. \tag{6.3}$$

Furthermore, $h_q(\mathbf{z})$ which introduce the coherence between dictionary columns formulated as follow,

$$h_q(\mathbf{z}) = \frac{1}{2}(|\mathbf{z}|^q)^{\mathrm{T}}\mathbf{R}|\mathbf{z}|^q = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\mathbf{R}_{jk}|\mathbf{z}_j|^q|\mathbf{z}_k|^q. \tag{6.4}$$

In the equation (5.4), $\mathbf{R} \in \mathbb{R}^{n \times n}$ is a symmetric matrix whose component $\mathbf{R}_{jk} \geq 0$ represents for the coherence between dictionary columns $\mathbf{D}_j$ and $\mathbf{D}_k$, the definition of $\mathbf{R}$ is shown as follow,

$$\mathbf{R}_{jk} = \begin{cases} |\mathbf{D}_j^{\mathrm{T}}\mathbf{D}_k| & ,j \neq k \\ 0 & ,j = k. \end{cases} \tag{6.5}$$

The key element for building a DNN-SC algorithm from a sparse coding algorithm is that the encoders are continuous and overall differentiable throughout its whole process. In the following sections, we would like to propose three DNN-SCs relatively based on IILasso, II-ISTA and IIWLasso. The origin for training their network parameters is the same as discussed in Chapter 3, namely supervised learning using equation (4.10) and unsupervised learning using equation (4.12).

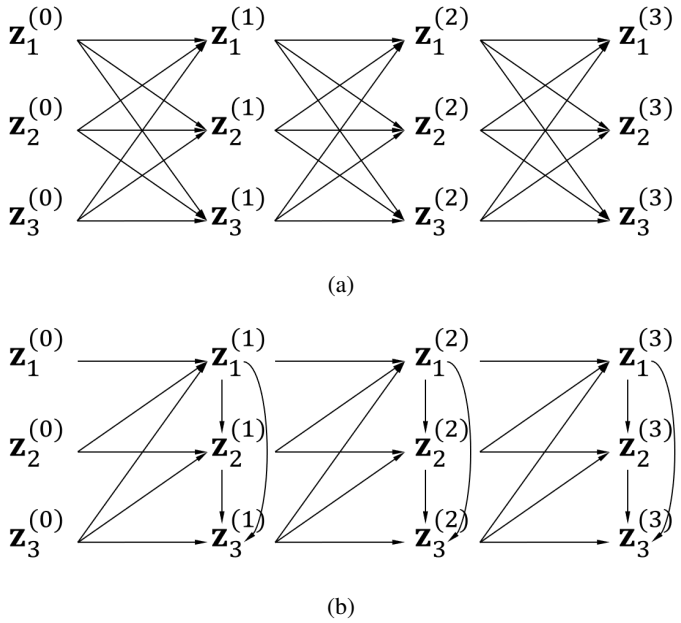


Figure 6.1: Neurons update diagram comparison between (a) PO, e.g. ISTA, IHTA, II-ISTA, etc.; and (b) CDA, e.g. Lasso, IILasso, IIWLasso, etc..

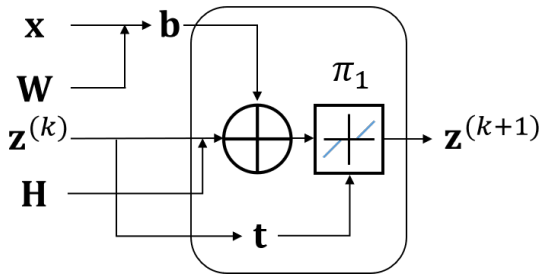
6.2.1 DNN-structured IILasso

By unfolding the iterations of IILasso from Algorithm 5.1, we can construct neural network structures. The algorithm can be rewritten as a network structure as follows, its neuron update diagram is shown in Figure 6.1 (b) and the unfolded network structure is shown in Figure 6.2 (b). Since IILasso use CDA to update elementwisely, we can figure that the network of IILasso is also element-wise which is key difference between DNN-IILasso and those DNN-SC algorithms using PO.

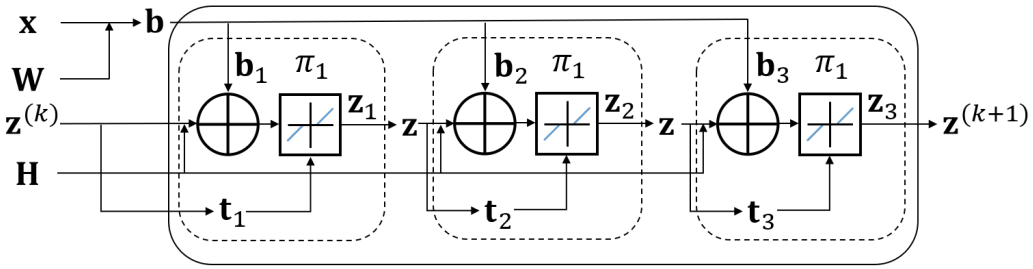
Algorithm 6.1: DNN-IILasso Forward propagation

Input: data $\mathbf{x}$, column normalized dictionary $\mathbf{D}$, proper parameters λ and γ, network layer T.
Initialization: $\mathbf{z}^{(0)} = \mathbf{0}$, $\mathbf{H} = \mathbf{I} - \mathbf{D}^{\mathrm{T}}\mathbf{D}$, $\mathbf{W} = \mathbf{D}^{\mathrm{T}}$, $\mathbf{b} = \mathbf{W}\mathbf{x}$.
For $k = 1$ to T
  for $i = 1$ to n
    $\mathbf{c}_i^{(k-1)} = \mathbf{b}_i + \mathbf{H}\mathbf{z}^{(k-1)}$
    $\mathbf{t}_i^{(k-1)} = \lambda + \gamma \mathbf{R}_{i:}|\mathbf{z}|$
    $\mathbf{z}_i^{(k)} = \pi_1(\mathbf{c}_i^{(k-1)}, \mathbf{t}_i^{(k-1)})$
  end for
End For
Output: $\mathbf{C} = \{\mathbf{c}^{(0)}, ..., \mathbf{c}^{(T-1)}\}$, $\mathbf{Z} = \{\mathbf{z}^{(0)}, ..., \mathbf{z}^{(T)}\}$

In the network, $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{t}$ are still parameters to train. Comparing forward structure with the previous DNN-SCs in Chapter 4, we can figure that the process is very similar that the se-



(a)



Note: For IILasso, $\mathbf{t}_i = \lambda + \gamma \mathbf{R}_{i:}|\mathbf{z}|$; for IIWLasso, $\mathbf{t}_i = \left|\mathbf{z}_i^{(k)}\right|^{p-1} (\lambda + \gamma \mathbf{R}_{i:}|\mathbf{z}|^q)$.

(b)

Figure 6.2: Illustration comparison of unfolded k-th iteration (part surrounded by black frame) for $\mathbf{z} = [\mathbf{z}_1; \mathbf{z}_2; \mathbf{z}_3]$ between (a) II-ISTA, where the optimal sparse representation can be recursively obtained in two steps: $\mathbf{z}^{(k+1)} = \pi_1(\mathbf{b} + \mathbf{H}\mathbf{z}^{(k)}, \mathbf{t})$, $\mathbf{t} = \lambda + \gamma \mathbf{R}|\mathbf{z}|^{(k)}$; and (b) IILasso and IIWLasso, where the optimal sparse representation can be recursively implementing an iteration with $n = 3$ steps, and every step contains two sub-steps: $\mathbf{t}_i = \lambda + \gamma \mathbf{R}_{i:}|\mathbf{z}|$ for IILasso and $\mathbf{t}_i = |\mathbf{z}_i^{(k)}|^{p-1}(\lambda + \gamma \mathbf{R}_{i:}|\mathbf{z}|^q)$ for IIWLasso, $\mathbf{z}_i = \pi_1(\mathbf{b}_i + \mathbf{H}_{i:}\mathbf{z}, \mathbf{t}_i)$. $\mathbf{x}$ is the input signal, π_1 is the soft thresholding operator with a changing threshold $\mathbf{t}$ during the iterations, $\mathbf{W} = \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}$, $\mathbf{b} = \mathbf{W}\mathbf{x}$, $\mathbf{H} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}\mathbf{D}$, $\alpha >$ largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$ for II-ISTA and $\alpha = 1$ for IILasso and IIWLasso.

Algorithm 6.2: DNN-IILasso Back propagation

Input: $\mathbf{x}$, $\mathbf{D}$, $\delta\mathbf{z}^{(T)}$, $\mathbf{Z}$, $\mathbf{C}$, $\mathbf{b}$, $\mathbf{H}$, $\mathbf{t}$, λ and γ.
Initialization: $\delta\mathbf{t}^{(T)} = \mathbf{0}$, $\delta\mathbf{b}^{(T)} = \mathbf{0}$, $\delta\mathbf{H}^{(T)} = \mathbf{0}$.
For $k = T - 1$ down to 0
  for $i = n$ down to 1
    $\delta\mathbf{t}_i^{(k)} = \delta\mathbf{t}_i^{(k+1)} + \partial_{\mathbf{t}}\pi_1(\mathbf{c}_i^{(k)}, \mathbf{t}_i^{(k)})\delta\mathbf{z}_i^{(k+1)}$
    $\delta\mathbf{c}_i^{(k)} = \partial_{\mathbf{c}}\pi_1(\mathbf{c}_i^{(k)}, \mathbf{t}_i^{(k)})\delta\mathbf{z}_i^{(k+1)}$
    $\delta\mathbf{b}_i^{(k)} = \delta\mathbf{b}_i^{(k+1)} + \delta\mathbf{c}_i^{(k)}$
    $\delta\mathbf{H}_{i:}^{(k)} = \delta\mathbf{H}_{i:}^{(k+1)} + \delta\mathbf{c}_{i:}^{(k)}\mathbf{z}^{\mathrm{T}}$
    $\delta\mathbf{z}_i^{(k)} = \mathbf{H}^{\mathrm{T}}\delta\mathbf{c}_i^{(k)} + \partial_{\mathbf{z}}\mathbf{t}_i^{(k)}\delta\mathbf{t}_i^{(k)}$
  end for
End For
$\delta\mathbf{W} = \delta\mathbf{b}^{(0)}\mathbf{x}^{\mathrm{T}}$
Output: $\delta\mathbf{W}$, $\delta\mathbf{H}^{(0)}$, $\delta\mathbf{t}^{(0)}$

quence is constructed by connecting linear transformations and non-linear transformations. The difference is that there is only one pair of linear transformation and non-linear transformation

in one layer of previous DNN-SCs, while there are n pairs in DNN-IILasso. After the forward propagation with determined network layer T, $\mathbf{C}$ and $\mathbf{Z}$ of each layer in the IILasso network are saved for the back-propagation parameter training. The gradient-based parameter learning schedules of the network are shown in Algorithm 6.2. The learning procedure can be either adapted to supervised or unsupervised learning by respectively choosing $\delta\mathbf{z}^{(T)}$ from equation (4.10) or equation (4.12).

6.2.2 DNN-structured II-ISTA

By unfolding the iterations of IILasso from Algorithm 5.2, we can construct neural network structures for II-ISTA. The algorithm can be rewritten as a network structure in Algorithm 6.3, its neuron update diagram is shown in Figure 6.1 (a) and the unfolded network structure is shown in Figure 6.2 (a). II-ISTA has similar structure as WISTA that their thresholds have a weight part compared to original ISTA, the difference is that weight fro II-ISTA based on coherence of dictionary.

Algorithm 6.3: DNN-II-ISTA Forward propagation

Input: data $\mathbf{x}$, dictionary $\mathbf{D}$, proper parameters λ and α, network layer T.
Restriction: $\alpha >$ largest eigenvalue of $\mathbf{D}^{\mathrm{T}}\mathbf{D}$
Initialization: $\mathbf{z}^{(0)} = \mathbf{0}$, $\mathbf{H} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}\mathbf{D}$, $\mathbf{W} = \frac{1}{\alpha}\mathbf{D}^{\mathrm{T}}$, $\mathbf{b} = \mathbf{W}\mathbf{x}$.
For $k = 1$ to T
  $\mathbf{c}^{(k-1)} = \mathbf{b} + \mathbf{H}\mathbf{z}^{(k-1)}$
  $\mathbf{t}^{(k-1)} = \lambda + \gamma\mathbf{R}|\mathbf{z}|^{(k-1)}$
  $\mathbf{z}^{(k)} = \pi_1(\mathbf{c}^{(k-1)}, \mathbf{t}^{(k-1)})$
End For
Output: $\mathbf{C} = \{\mathbf{c}^{(0)}, ..., \mathbf{c}^{(T-1)}\}$, $\mathbf{Z} = \{\mathbf{z}^{(0)}, ..., \mathbf{z}^{(T)}\}$

Algorithm 6.4: DNN-II-ISTA Back propagation

Input: $\mathbf{x}$, $\mathbf{D}$, $\delta\mathbf{z}^{(T)}$, $\mathbf{Z}$, $\mathbf{C}$, $\mathbf{b}$, $\mathbf{H}$, $\mathbf{t}$, λ and α.
Initialization: $\delta\mathbf{t}^{(T)} = \mathbf{0}$, $\delta\mathbf{b}^{(T)} = \mathbf{0}$, $\delta\mathbf{H}^{(T)} = \mathbf{0}$.
For $k = T - 1$ down to 0
  $\delta\mathbf{t}^{(k)} = \delta\mathbf{t}^{(k+1)} + \frac{\partial\pi_1(\mathbf{c}^{(k)},\mathbf{t}^{(k)})}{\partial\mathbf{t}}\delta\mathbf{z}^{(k+1)}$
  $\delta\mathbf{c}^{(k)} = \frac{\partial\pi_1(\mathbf{c}^{(k)},\mathbf{t}^{(k)})}{\partial\mathbf{c}}\delta\mathbf{z}^{(k+1)}$
  $\delta\mathbf{b}^{(k)} = \delta\mathbf{b}^{(k+1)} + \delta\mathbf{c}^{(k)}$
  $\delta\mathbf{H}^{(k)} = \delta\mathbf{H}^{(k+1)} + \delta\mathbf{c}^{(k)}\mathbf{z}^{(k)^{\mathrm{T}}}$
  $\delta\mathbf{z}^{(k)} = \mathbf{H}^{\mathrm{T}}\delta\mathbf{c}^{(k)}$
End For
$\delta\mathbf{W} = \delta\mathbf{b}^{(0)}\mathbf{x}^{\mathrm{T}}$
Output: $\delta\mathbf{W}$, $\delta\mathbf{H}^{(0)}$, $\delta\mathbf{t}^{(0)}$

In the network, $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{t}$ are still parameters to train. After the forward propagation

with determined network layer T, $\mathbf{C}$ and $\mathbf{Z}$ of each layer in the II-ISTA network are saved for the back-propagation parameter training. The gradient-based parameter learning schedules of the network are shown in Algorithm 6.4. The learning procedure can be either adapted to supervised or unsupervised learning by respectively choosing $\delta\mathbf{z}^{(T)}$ from equation (4.10) or equation (4.12).

6.2.3 DNN-structured IIWLasso

By unfolding the iterations of IIWLasso from Algorithm 5.3, we can construct neural network structures. The algorithm can be rewritten as a network structure as follows, its neuron update diagram is shown in Figure 6.1 (b) and the unfolded network structure is shown in Figure 6.2 (b). IIWLasso also update elementwisely using CDA, so it has similar structure as IILasso.

Algorithm 6.5: DNN-IIWLasso Forward propagation

Input: data $\mathbf{x}$, column normalized dictionary $\mathbf{D}$, proper parameters λ and γ, network layer T.
Initialization: $\mathbf{z}^{(0)} = \mathbf{0}$, $\mathbf{H} = \mathbf{I} - \mathbf{D}^{\mathrm{T}}\mathbf{D}$, $\mathbf{W} = \mathbf{D}^{\mathrm{T}}$, $\mathbf{b} = \mathbf{W}\mathbf{x}$.
For $k = 1$ to T
  for $i = 1$ to n
    $\mathbf{c}_i^{(k-1)} = \mathbf{b}_i + \mathbf{H}\mathbf{z}^{(k-1)}$
    $\mathbf{t}_i^{(k-1)} = |\mathbf{z}_i^{(k-1)}|^{p-1}(\lambda + \gamma\mathbf{R}_{i:}|\mathbf{z}|^q)$
    $\mathbf{z}_i^{(k)} = \pi_1(\mathbf{c}_i^{(k-1)}, \mathbf{t}_i^{(k-1)})$
  end for
End For
Output: $\mathbf{C} = \{\mathbf{c}^{(0)}, ..., \mathbf{c}^{(T-1)}\}$, $\mathbf{Z} = \{\mathbf{z}^{(0)}, ..., \mathbf{z}^{(T)}\}$

Algorithm 6.6: DNN-IIWLasso Back propagation

Input: $\mathbf{x}$, $\mathbf{D}$, $\delta\mathbf{z}^{(T)}$, $\mathbf{Z}$, $\mathbf{C}$, $\mathbf{b}$, $\mathbf{H}$, $\mathbf{t}$, λ and γ.
Initialization: $\delta\mathbf{t}^{(T)} = \mathbf{0}$, $\delta\mathbf{b}^{(T)} = \mathbf{0}$, $\delta\mathbf{H}^{(T)} = \mathbf{0}$.
For $k = T - 1$ down to 0
  for $i = n$ down to 1
    $\delta\mathbf{t}_i^{(k)} = \delta\mathbf{t}_i^{(k+1)} + \partial_{\mathbf{t}}\pi_1(\mathbf{c}_i^{(k)}, \mathbf{t}_i^{(k)})\delta\mathbf{z}_i^{(k+1)}$
    $\delta\mathbf{c}_i^{(k)} = \partial_{\mathbf{c}}\pi_1(\mathbf{c}_i^{(k)}, \mathbf{t}_i^{(k)})\delta\mathbf{z}_i^{(k+1)}$
    $\delta\mathbf{b}_i^{(k)} = \delta\mathbf{b}_i^{(k+1)} + \delta\mathbf{c}_i^{(k)}$
    $\delta\mathbf{H}_{i:}^{(k)} = \delta\mathbf{H}_{i:}^{(k+1)} + \delta\mathbf{c}_{i:}^{(k)}\mathbf{z}^{\mathrm{T}}$
    $\delta\mathbf{z}_i^{(k)} = \mathbf{H}^{\mathrm{T}}\delta\mathbf{c}_i^{(k)} + \partial_{\mathbf{z}}\mathbf{t}_i^{(k)}\delta\mathbf{t}_i^{(k)}$
  end for
End For
$\delta\mathbf{W} = \delta\mathbf{b}^{(0)}\mathbf{x}^{\mathrm{T}}$
Output: $\delta\mathbf{W}$, $\delta\mathbf{H}^{(0)}$, $\delta\mathbf{t}^{(0)}$

In the network, $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{t}$ are still parameters to train. Comparing with DNN-IILasso, the difference is the existence of an additional weight part to approximate ℓ_p norm ($0 < p < 1$).

After the forward propagation with determined network layer T, $\mathbf{C}$ and $\mathbf{Z}$ of each layer in the IIWLasso network are saved for the back-propagation parameter training. The gradient-based parameter learning schedules of the network are shown in Algorithm 6.6. The learning procedure can be either adapted to supervised or unsupervised learning by respectively choosing $\delta\mathbf{z}^{(T)}$ from equation (4.10) or equation (4.12).

6.3 Experiments

6.3.1 Synthetic data experiments

Experiments settings in the synthetic data experiments are the same as those in Chapter 5, namely, dictionaries $\mathbf{D}_{\text{c}} \in \mathbb{R}^{30\times 50}$ are generated from gaussian distributed values. $\mathbf{D}_{0.15}$ was generated by drawing value randomly from the normal distribution $N(0,1)$. $\mathbf{D}_{0.50}$ was generated by adding a small normal distributed vector to a baseline vector generated from the normal distribution $N(0,1)$. Coherence distribution of the two dictionary have been shown in Figure 5.2 in Chapter 5. The quantity of ground true sparse representation vectors $\mathbf{Z}_{\text{orig}} = (\mathbf{z}_{\text{orig}1}, ..., \mathbf{z}_{\text{orig}100})$ was 100 in experiments. Correspondingly, ground true data set $\mathbf{X}_{\text{orig}}$ had 100 sample size which was generated by $\mathbf{D}_{\text{c}}$ and $\mathbf{Z}_{\text{orig}}$ based on the equation (6.1), The noise vectors $\mathbf{v}$ were added based on Gaussian random entries with 20dB SNR. Thus we have the signal set $\mathbf{X} \in \mathbb{R}^{30\times 100}$ to train DNN.

Experimental results are compared in terms of relative norm error, support error and Hoyer sparsity with the same definition in Chapter 5. We present supervised learnt DNN in this chapter. Both IIWW and DNN-IIWW were set as $p = 0.7$ and $q = 0.7$.

Figure 6.3 shows relative norm errors and support error of different independently interpretable DNN-SC (II-DNN-SC) algorithms in a range of layers with $\mathbf{D}_{0.15}$. In general, we can see that with appropriate layer numbers for different II-DNN-SC algorithms, all these II-DNN-SC algorithms can obtain equivalent or smaller relative norm errors and support error compared to converged results of their original sparse coding algorithm. Except DNN-IILasso, increase of layers can result in obvious decrease in relative norm errors and support error of II-DNN-SC. Relative norm error and support error of DNN-IILasso rebound to converged results of IILasso after 2 layers, indicating that it might not be good to construct a too deep neural network for supervised trained DNN-IILasso which may learn too much from the original sparse coding structure. Moreover, we can see that DNN-II-ISTA can obtain both smaller relative norm error

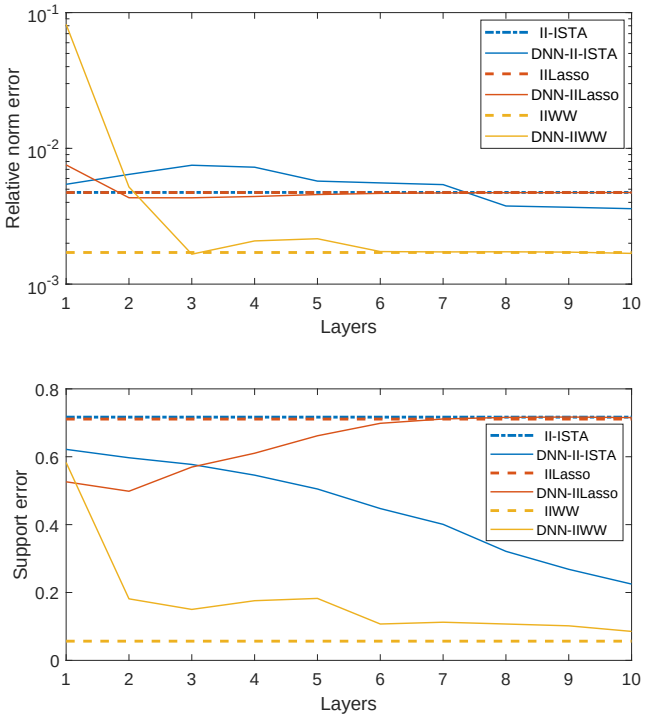


Figure 6.3: Relative norm errors and support error comparison between converged results of different independently interpretable algorithms and their DNN-SC versions in a range of layers with $\mathbf{D}_{0.15}$.

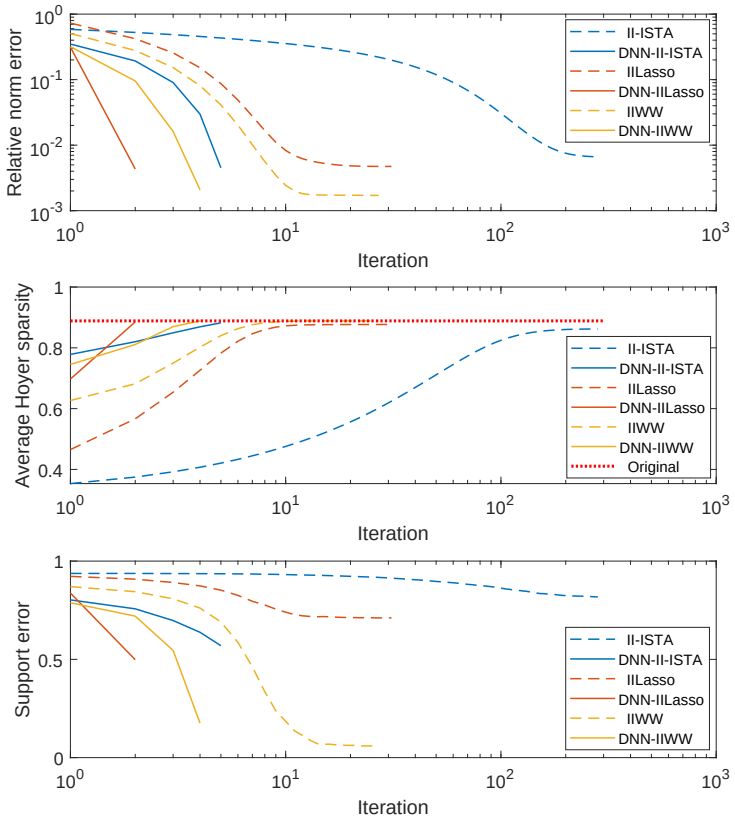


Figure 6.4: Average relative norm errors, average Hoyer sparsity and support error comparison between different independently interpretable algorithms and their DNN-SC versions with $\mathbf{D}_{0.15}$.

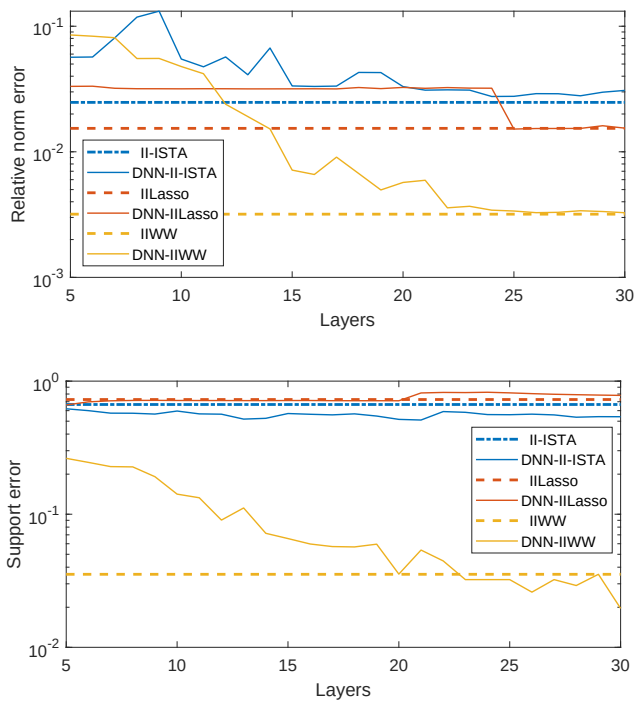


Figure 6.5: Relative norm errors and support error comparison between converged results of different independently interpretable algorithms and their DNN-SC versions in a range of layers with $\mathbf{D}_{0.50}$.

and support error when layer is over 8. Furthermore, DNN-IIWW can obtain the best performance in all II-DNN-SC algorithm benefited from good performance of its original algorithm.

We then present performances of 5 layer DNN-II-ISTA, 2 layer DNN-IILasso and 4 layer DNN-IIWW, compared with the convergence graph of their original sparse coding algorithms with $\mathbf{D}_{0.15}$ in Figure 6.4. We can see that well trained II-DNN-SC algorithms can accelerate about 10 times compared their original sparse coding algorithms with equivalent relative norm errors, Hoyer sparsity and support error.

Figure 6.5 shows relative norm errors and support error of different independently interpretable DNN-SC (II-DNN-SC) algorithms in a range of layers with $\mathbf{D}_{0.50}$. In general, we can see that with appropriate layer numbers for different II-DNN-SC algorithms, all these II-DNN-SC algorithms can obtain equivalent or smaller relative norm errors and support error compared to converged results of their original sparse coding algorithm. Increase of layers can result in obvious decrease in relative norm errors for all II-DNN-SC algorithms. Moreover, we can observe obvious increase in layers for obtaining good performances in all II-DNN-SC algorithms with $\mathbf{D}_{0.50}$, which is caused by increase in convergence iteration number of their original sparse coding algorithms which was shown in Chapter 5. Furthermore, we can see that DNN-IIWW can obtain smallest relative norm error among all II-DNN-SC algorithms and smallest support

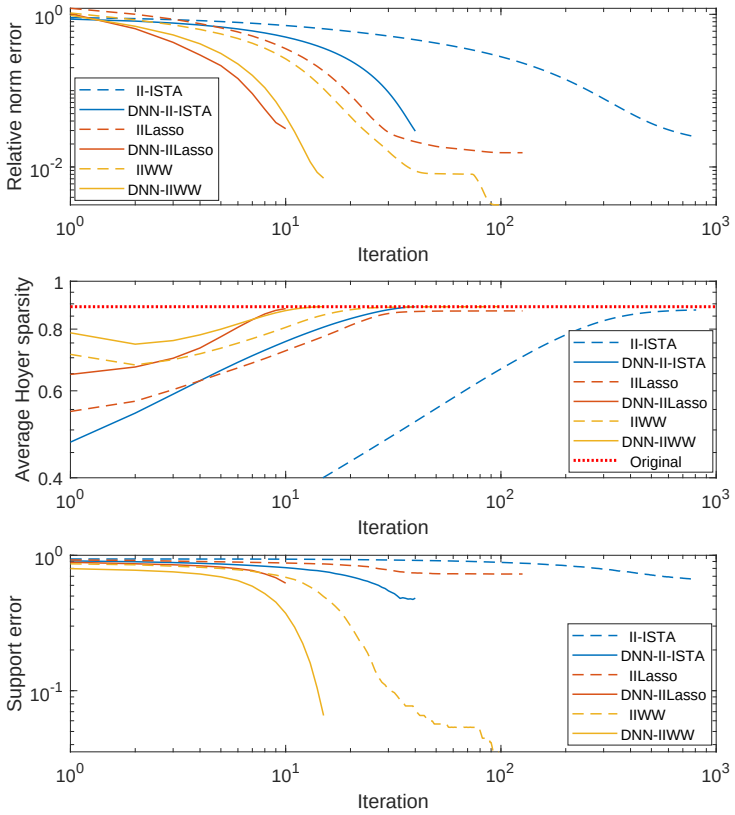


Figure 6.6: Average relative norm errors, average Hoyer sparsity and support error comparison between different independently interpretable algorithms and their DNN-SC versions with $\mathbf{D}_{0.50}$.

error in all sparse coding algorithms when layer is larger than 23.

We then present performances of 40 layer DNN-II-ISTA, 10 layer DNN-IILasso and 15 layer DNN-IIWW, compared with the convergence graph of their original sparse coding algorithms with $\mathbf{D}_{0.50}$ in Figure 6.6. We can see that well trained II-DNN-SC algorithms can accelerate about 10 times compared their original sparse coding algorithms with equivalent relative norm errors, Hoyer sparsity and support error.

6.4 Chapter Summary

In this chapter, we propose to build DNN-SC algorithms for highly correlated data by unfolding independently interpretable algorithms and training approximate encoders for them. We form three new II-DNN-SC algorithms from IILasso, II-ISTA and IIWLasso. Synthetic data experiments have been conducted to validate performances of proposed algorithms. We show that proposed II-DNN-SC algorithms can obtain equivalent or smaller relative norm error and support error in different coherence condition, while trained encoder can be about 10 times faster

than their original sparse coding algorithms.

Chapter 7

Conclusions

In this chapter, we conclude the dissertation by summarizing our contributions and presenting directions for further research.

7.1 Contributions

This thesis makes contributions in the area of sparse representation of signals. We now highlight the main contributions of the dissertation of each chapter.

In Chapter 3, we presented a dictionary learning algorithm to train a data-adaptive dictionary with ℓ_p norm ($0 < p < 1$) regularization. The algorithm utilizes two approximations in the regularization, namely weighted ℓ_1 norm for ℓ_p norm and a smoothed approximation for the absolute value, to make it possible to use gradient descent method to update both sparse coefficient set and dictionary. We validated our algorithm in synthetic data experiments with different noise level. A remarkable advantage of the proposed dictionary learning algorithm HDLWL is its robustness to noise that it can recover synthetic dictionary to nearly 100% when SNR level is higher than 10 dB where classical dictionary learning algorithms may fail.

In Chapter 4, we proposed to implement parameter training methods of RNN in ℓ_p norm ($0 < p < 1$) based iterative shrinkage algorithms. We show how we can formulate ℓ_p norm ($0 < p < 1$) based DNN-SCs by unfolding truncated iterations of their original algorithms. Moreover, we present how we can train DNN-SCs supervisedly or unsupervisedly. In the synthetic data experiments, we show our proposed algorithms can effectively enhance performance in efficiently finding sparse and accurate representations. In image denoising experiments, we show that all DNN-SCs can accelerate the denoising procedures at least 45 times while obtaining

equivalent performances compared to their original algorithms. Among all algorithms, DNN-WISTA0.5 and DNN-IHTA can obtain the best results from 20.0 dB to approximately 30.8 dB. Finally, in video-denoising experiments, we use unsupervised learnt DNN-SCs to realize the goal of video online denoising. We show acceleration brought by DNN-SC can help reducing the processing time for denoising to the interval between frames of 25-FPS 360×480-pixel gray-scaled videos using only CPU, indicating that TISTA and our proposed TWISTA can be applied in real-time video denoising for a 25-FPS video with good denoising results.

In Chapter 5, we introduce ℓ_p norm ($0 < p < 1$) to the coherence related regularization of IILasso for processing highly correlated data. To efficiently solve the non-convex problem brought by ℓ_p norm ($0 < p < 1$), we use CDA with weighted ℓ_1 norm and the Proximal Operator (PO) to solve the optimization problem with the new regularization. In synthetic data experiments, we show our proposed algorithms can obtain smaller relative norm error and support error in various coherence condition. We then validate performance of our algorithms in highly correlated gene expression datasets by conducting 10-fold cross-validation. Our proposed algorithm IIWW0.7 can obtain the best misclassification errors in various gene expression datasets indicating the algorithm can interpret the "decomposability" of datasets and thus present reasonable suggestions for diseases and developmental stages on gene expression.

In Chapter 6, we further enhance efficiency of proposed algorithms in Chapter 5 to form their DNN-SC versions. Details of how to train parameters of these DNN-SCs are shown. In synthetic data experiments, we show these independently interpretable DNN-SCs can obtain equivalent performance in relative norm error and support error while obviously reducing computation time.

7.2 Future Works

Choice of the regularization parameter

A very important issue in the methods presented in above chapters is the choice of the regularization parameters. λ in regularization part of all algorithm helps balancing effects of norm error to signals with sparse constraint. c in Chapter 3 helps approximate the absolute function. Although larger c makes the approximation closer to the absolute function, it is not a good choice to use too large c in HDLWL. In Chapter 5, λ and γ help balancing effects of classical sparse constraint $\|\mathbf{z}\|_p^p$ with the coherence-induced independently interpretable regularization. We can

notice that best λ and γ change with increasing of coherence. In the future, we require to develop an automatic method for the choice of the regularization parameters as in [79] in order to obtain the optimal solution and reduce iterations as far as possible.

Applications

In Chapter 3 and 6, we validate performance of our algorithms in synthetic data experiments, more experiments for real world data remain to be done. Moreover, DNN-SC has proven its ability in performing online denoising, more online processing applications can be considered for DNN-SC algorithms. The same situation are independently interpretable algorithms, which is not restricted in gene expression datasets. In the future, we will further explore potentials of our proposed algorithms in different suitable areas.

References

[1] R. Baraniuk, V. Cevher, and M. B. Wakin, "Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective," *Proceedings of the IEEE*, vol. 98, no. ARTICLE, pp. 959–971, 2010.

[2] M. Elad, *Sparse and Redundant Representations From Theory to Applications in Signal and Image Processing*. NewYork: Springer, 2010.

[3] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM journal on computing*, vol. 24, no. 2, p. 227C234, 1995.

[4] M. Huang, W. Yang, J. Jiang, Y. Wu, Y. Zhang, W. Chen, Q. Feng, A. D. N. Initiative *et al.*, "Brain extraction based on locally linear representation-based classification," *Neuroimage*, vol. 92, pp. 322–339, 2014.

[5] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[6] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM review*, vol. 51, no. 1, pp. 34–81, 2009.

[7] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *Multiscale Modeling & Simulation*, vol. 7, no. 1, pp. 214–241, 2008.

[8] D. L. Donoho *et al.*, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[9] R. G. Baraniuk, "Compressive sensing," *IEEE signal processing magazine*, vol. 24, no. 4, 2007.

[10] Y. Wang, J. Zeng, Z. Peng, X. Chang, and Z. Xu, "Linear convergence of adaptively iterative thresholding algorithms for compressed sensing," *IEEE Transactions on Signal Processing*, vol. 63, no. 11, pp. 2957–2971, 2015.

[11] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling [a sensing/sampling paradigm that goes against the common knowledge in data acquisition]," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[12] Y. Tsaig and D. L. Donoho, "Extensions of compressed sensing," *Signal processing*, vol. 86, no. 3, pp. 549–571, 2006.

[13] E. Candes and J. Romberg, "Sparsity and incoherence in compressive sampling," *Inverse problems*, vol. 23, no. 3, p. 969, 2007.

[14] M. Elad, M. A. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 972–982, 2010.

[15] M. G. Jafari and M. D. Plumbley, "Fast dictionary learning for sparse representations of speech signals," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 1025–1031, 2011.

[16] M.-J. Fadili, J.-L. Starck, and F. Murtagh, "Inpainting and zooming using sparse representations," *The Computer Journal*, vol. 52, no. 1, pp. 64–79, 2009.

[17] A. Adler, V. Emiya, M. G. Jafari, M. Elad, R. Gribonval, and M. D. Plumbley, "Audio inpainting," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 922–932, 2011.

[18] C. F. Caiafa and A. Cichocki, "Computing sparse representations of multidimensional signals using kronecker bases," *Neural computation*, vol. 25, no. 1, pp. 186–220, 2013.

[19] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[20] W. Dong, L. Zhang, and G. Shi, "Centralized sparse representation for image restoration," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1259–1266.

[21] J. A. Tropp, "Greed is good: algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, Oct 2004.

[22] I. Daubechies, M. Defrise, and C. D. Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[23] J. Friedman, T. Hastie, H. Höfling, R. Tibshirani *et al.*, "Pathwise coordinate optimization," *The annals of applied statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[24] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, vol. 33, no. 1, p. 1, 2010.

[25] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001. [Online]. Available: https://doi.org/10.1137/S003614450037906X

[26] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural computation*, vol. 15, no. 2, pp. 349–396, 2003.

[27] Z. Xu, X. Chang, F. Xu, and H. Zhang, "$l_{\frac{1}{2}}$ regularization: A thresholding representation theory and a fast solver," *IEEE Transactions on Neural Networks and Learning Systems,*, vol. 23, no. 7, pp. 1013–1027, 2012.

[28] F. Chen, "Composite minimization: Proximity algorithms and their applications," *Dissertations - ALL*, 2015.

[29] W. Cao, J. Sun, and Z. Xu, "Fast image deconvolution using closed-form thresholding formulas of lq (q= 12, 23) regularization," *Journal of Visual Communication and Image Representation*, vol. 24, no. 1, pp. 31–41, 2013.

[30] M. A. T. Figueiredo and R. D. Nowak, "A bound optimization approach to wavelet-based image deconvolution," in *IEEE International Conference on Image Processing 2005*, vol. 2, Sept 2005, pp. II–782.

[31] M. A. T. Figueiredo, J. M. Bioucas-Dias, and R. D. Nowak, "Majorization-minimization algorithms for wavelet-based image restoration," *IEEE Transactions on Image Processing*, vol. 16, no. 12, pp. 2980–2991, Dec 2007.

[32] E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted l_1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 877–905, Dec 2008. [Online]. Available: https://doi.org/10.1007/s00041-008-9045-x

[33] H. Zhao, S. Ding, Y. Li, Z. Li, X. Li, and B. Tan, "Dictionary learning for sparse representation using weighted l_1-norm," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec 2016, pp. 292–296.

[34] H. Zhao, S. Ding, X. Li, and H. Huang, "Deep neural network structured sparse coding for online processing," *IEEE Access*, vol. 6, pp. 74 778–74 791, 2018.

[35] D. Malioutov and A. Aravkin, "Iterative log thresholding," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 7198–7202.

[36] Z. Li, S. Ding, T. Hayashi, and Y. Li, "Incoherent dictionary learning with log-regularizer based on proximal operators," *Digital Signal Processing*, vol. 63, pp. 86–99, 2017.

[37] P. Bühlmann and S. Van De Geer, *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011.

[38] P. J. Bickel, Y. Ritov, A. B. Tsybakov *et al.*, "Simultaneous analysis of lasso and dantzig selector," *The Annals of Statistics*, vol. 37, no. 4, pp. 1705–1732, 2009.

[39] Z. C. Lipton, "The mythos of model interpretability," *arXiv preprint arXiv:1606.03490*, 2016.

[40] M. Takada, T. Suzuki, and H. Fujisawa, "Independently interpretable lasso: A new regularizer for sparse regression with uncorrelated variables," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Storkey and F. Perez-Cruz, Eds., vol. 84. Playa Blanca, Lanzarote, Canary Islands: PMLR, 09–11 Apr 2018, pp. 454–463. [Online]. Available: http://proceedings.mlr.press/v84/takada18a.html

[41] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury, "The dual-tree complex wavelet transform," *IEEE signal processing magazine*, vol. 22, no. 6, pp. 123–151, 2005.

[42] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, 1st ed. Springer-Verlag New York, 2010.

[43] M. Aharon, M. Elad, A. Bruckstein *et al.*, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, p. 4311, 2006.

[44] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, Dec 2006.

[45] E. K., A. S. O., and H. J. H., "Method of optimal directions for frame design," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1999, pp. 2443–2446.

[46] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 399–406. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104374

[47] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Learning efficient sparse and low rank models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1821–1833, 2015.

[48] Z. Wang, Q. Ling, and T. S. Huang, "Learning deep ℓ_0 encoders," *CoRR*, vol. abs/1509.00153, 2015. [Online]. Available: http://arxiv.org/abs/1509.00153

[49] M. Borgerding and P. Schniter, "Onsager-corrected deep networks for sparse linear inverse problems," *CoRR*, vol. abs/1612.01183, 2016. [Online]. Available: http://arxiv.org/abs/1612.01183

[50] R. R., P. T., and E. M., "Analysis k-svd: A dictionary-learning algorithm for the analysis sparse model," *IEEE Trans. on Signal Processing*, vol. 61, no. 3, pp. 661–677, 2013.

[51] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar conference on signals, systems and computers*. IEEE, 1993, pp. 40–44.

[52] S.-B. Chen, C. Ding, B. Luo, and Y. Xie, "Uncorrelated lasso," in *Twenty-seventh AAAI conference on artificial intelligence*, 2013.

[53] D. Kong, R. Fujimaki, J. Liu, F. Nie, and C. Ding, "Exclusive feature learning on arbitrary structures via $l_{1,2}$-norm," in *Advances in Neural Information Processing Systems*, 2014, pp. 1655–1663.

[54] J. Domke, "Parameter learning with truncated message-passing," in *CVPR 2011*, June 2011, pp. 2937–2943.

[55] ——, "Learning graphical model parameters with approximate marginal inference," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 10, pp. 2454–2467, 2013.

[56] U. Kamilov and H. Mansour, "Learning optimal nonlinearities for iterative thresholding algorithms," *IEEE Signal Process Letters*, vol. 23, no. 5, pp. 747–751, 2016.

[57] D. Mahapatra, S. Mukherjee, and C. S. Seelamantula, "Deep sparse coding using optimized linear expansion of thresholds," *CoRR*, vol. abs/1705.07290, 2017. [Online]. Available: http://arxiv.org/abs/1705.07290

[58] T. Moreau and J. Bruna, "Understanding Trainable Sparse Coding via Matrix Factorization," *ArXiv e-prints*, Sep. 2016.

[59] L. Guo and C. Guo, "A deep sparse coding method for fine-grained visual categorization," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 632–639.

[60] S. Zhang, J. Wang, X. Tao, Y. Gong, and N. Zheng, "Constructing deep sparse coding network for image classification," *Pattern Recognition*, vol. 64, pp. 130 – 140, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320316303466

[61] X. Sun, N. M. Nasrabadi, and T. D. Tran, "Supervised multilayer sparse coding networks for image classification," *CoRR*, vol. abs/1701.08349, 2017. [Online]. Available: http://arxiv.org/abs/1701.08349

[62] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," *CoRR*, vol. abs/1704.03264, 2017. [Online]. Available: http://arxiv.org/abs/1704.03264

[63] J. Zhang and B. Ghanem, "Ista-net: Iterative shrinkage-thresholding algorithm inspired deep network for image compressive sensing," *CoRR*, vol. abs/1706.07929, 2017. [Online]. Available: http://arxiv.org/abs/1706.07929

[64] J. R. Chang, C. Li, B. Póczos, B. V. K. V. Kumar, and A. C. Sankaranarayanan, "One network to solve them all - solving linear inverse problems using deep projection models," *CoRR*, vol. abs/1703.09912, 2017. [Online]. Available: http://arxiv.org/abs/1703.09912

[65] N. Gillis and F. Glineur, "Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization," *Neural computation*, vol. 24, no. 4, pp. 1085–1105, 2012.

[66] P. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of machine learning research,*, vol. 5, no. 11, pp. 1457–1469, 2004.

[67] X. Hu, F. Heide, Q. Dai, and G. Wetzstein, "Convolutional sparse coding for rgb+nir imaging," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1611–1625, April 2018.

[68] M. Hanif, A. Tonazzini, P. Savino, and E. Salerno, "Sparse representation based inpainting for the restoration of document images affected by bleed-through," *Proceedings*, vol. 2, no. 2, 2018. [Online]. Available: http://www.mdpi.com/2504-3900/2/2/93

[69] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, Nov 2010.

[70] J. Jiang, J. Ma, C. Chen, X. Jiang, and Z. Wang, "Noise robust face image super-resolution through smooth sparse representation," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3991–4002, Nov 2017.

[71] J. Zeng, S. Lin, Y. Wang, and Z. Xu, "$l_{\frac{1}{2}}$ regularization: Convergence of iterative half thresholding algorithm," *IEEE Transactions on Signal Processing*, vol. 62, no. 9, pp. 2317–2328, 2014.

[72] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[73] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating summaries from user videos," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 505–520.

[74] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proceedings of the National Academy of Sciences*, vol. 96, no. 12, pp. 6745–6750, 1999.

[75] M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J. L. Kutok, R. C. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus *et al.*, "Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning," *Nature medicine*, vol. 8, no. 1, p. 68, 2002.

[76] E. Gravier, G. Pierron, A. Vincent-Salomon, N. Gruel, V. Raynal, A. Savignoni, Y. De Rycke, J.-Y. Pierga, C. Lucchesi, F. Reyal *et al.*, "A prognostic dna signature for t1t2 node-negative breast cancer patients," *Genes, chromosomes and cancer*, vol. 49, no. 12, pp. 1125–1134, 2010.

[77] B. C. Christensen, E. A. Houseman, C. J. Marsit, S. Zheng, M. R. Wrensch, J. L. Wiemels, H. H. Nelson, M. R. Karagas, J. F. Padbury, R. Bueno *et al.*, "Aging and environmental exposures alter tissue-specific dna methylation dependent upon cpg island context," *PLoS genetics*, vol. 5, no. 8, p. e1000602, 2009.

[78] H. Huang, H. Zhao, X. Li, S. Ding, L. Zhao, and Z. Li, "An accurate and efficient device-free localization approach based on sparse coding in subspace," *IEEE Access*, vol. 6, pp. 61 782–61 799, 2018.

[79] K. S.J., K. K., L. M., B. Stephen, and G. Dimitry, "An interior-point method for large-scale ℓ^1-regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007.