New Jersey Institute of Technology
Digital Commons @ NJIT

Dissertations | Electronic Theses and Dissertations

5-31-2022

One-stage blind source separation via a sparse autoencoder framework

Jason Anthony Dabin
New Jersey Institute of Technology

Follow this and additional works at: https://digitalcommons.njit.edu/dissertations

 Part of the Artificial Intelligence and Robotics Commons, Electrical and Electronics Commons, and the Neuroscience and Neurobiology Commons

Recommended Citation

Dabin, Jason Anthony, "One-stage blind source separation via a sparse autoencoder framework" (2022). *Dissertations*. 1600.
https://digitalcommons.njit.edu/dissertations/1600

This Dissertation is brought to you for free and open access by the Electronic Theses and Dissertations at Digital Commons @ NJIT. It has been accepted for inclusion in Dissertations by an authorized administrator of Digital Commons @ NJIT. For more information, please contact digitalcommons@njit.edu.

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

NJIT New Jersey's Science & Technology University

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

ABSTRACT

ONE-STAGE BLIND SOURCE SEPARATION VIA A SPARSE AUTOENCODER FRAMEWORK

By
Jason Anthony Dabin

Blind source separation (BSS) is the process of recovering individual source transmissions from a received mixture of co-channel signals without a priori knowledge of the channel mixing matrix or transmitted source signals. The received co-channel composite signal is considered to be captured across an antenna array or sensor network and is assumed to contain sparse transmissions, as users are active and inactive aperiodically over time. An unsupervised machine learning approach using an artificial feedforward neural network sparse autoencoder with one hidden layer is formulated for blindly recovering the channel matrix and source activity of co-channel transmissions. The BSS sparse autoencoder provides one-stage learning using the receive signal data only, which solves for the channel matrix and signal sources simultaneously.

The recovered co-channel source signals are produced at the encoded output of the sparse autoencoder hidden layer. A complex-valued soft-threshold operator is used as the activation function at the hidden layer to preserve the ordered pairs of real and imaginary components. Once the weights of the sparse autoencoder are learned, the latent signals are recovered at the hidden layer without requiring any additional optimization steps. The generalization performance on future received data demonstrates the ability to recover signal transmissions on untrained data and outperform the two-stage BSS process.

ONE-STAGE BLIND SOURCE SEPARATION VIA A SPARSE AUTOENCODER FRAMEWORK

by
Jason Anthony Dabin

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Helen and John C. Hartmann Department of
Electrical and Computer Engineering

May 2022

Copyright © 2022 by Jason Anthony Dabin

ALL RIGHTS RESERVED

APPROVAL PAGE

ONE-STAGE BLIND SOURCE SEPARATION VIA A SPARSE AUTOENCODER FRAMEWORK

Jason Anthony Dabin

Dr. Alexander M. Haimovich, Dissertation Advisor — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Osvaldo Simeone, Committee Member — Date
Professor of Information Engineering, King's College London. London, U.K.

Dr. Joerg Kliewer, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Ali Abdi, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Hongya Ge, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Jason Anthony Dabin

Degree: Doctor of Philosophy

Date: May 2022

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2022

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2004

- Bachelor of Science in Electrical Engineering Technology
  New Jersey Institute of Technology, Newark, NJ, 2001

Major: Electrical Engineering

Presentations and Publications:

J. A. Dabin, A. M. Haimovich, J. Mauger, and J. Kliewer, "One-Stage Blind Co-Channel Source Separation via a Complex-Valued Sparse Autoencoder," *IEEE Transactions on Cognitive Communications and Networking*, submitted, 2022.

J. A. Dabin, A. M. Haimovich, J. Mauger and A. Dong, "Blind Source Separation with L1 Regularized Sparse Autoencoder," *29th Wireless and Optical Communications Conference (WOCC)*, 2020, pp. 1-5.

In dedication to my mother and father for their inspiration, my brother and sister for all their encouragement, and to my wife and children for all their patience and support during my doctoral academic journey.

ACKNOWLEDGMENT

I am sincerely grateful to have had Dr. Alexander M. Haimovich as my Ph.D. advisor and wish to express my deepest appreciation to Dr. Haimovich for all his unwavering support and profound guidance he has provided to me as a doctoral graduate student. Dr. Haimovich instilled key research attributes in me during my dissertation research pursuit that I will carry forward with me throughout my career. Dr. Haimovich has not only positively impacted me as a graduate student, but his guidance has had a significant impact on my professional career for which I am extremely thankful.

I would like to express my sincere gratitude to each of my Ph.D. dissertation committee members including Dr. Osvaldo Simeone, Dr. Joerg Kliewer, Dr. Ali Abdi, and Dr. Hongya Ge for their contributions, time, and participation in my dissertation proposal defense and dissertation defense. I am extremely appreciative of the comments, feedback, and suggestions provided by each of my Ph.D. committee members, which helped expand my research direction and scope.

I wish to thank the U.S. Department of Defense Science, Mathematics, and Research for Transformation (SMART) Scholarship-for-Service Program for sponsoring my Ph.D. degree as a retention scholar and my sponsoring facility Naval Information Warfare Center (NIWC) Pacific for their support as well. I am greatly appreciative of the SMART scholarship I received, which enabled me to complete my Ph.D. degree while being employed by NIWC Pacific.

I would like to extend my gratitude to my NIWC Pacific Information Operations Division Head Gregory Settelmayer who was extremely supportive of my doctoral degree and provided a work environment that enabled me to pursue my academic studies. I

sincerely appreciate the in-depth mathematical discussions and research guidance provided to me by my NIWC Pacific colleague Dr. Justin Mauger.

Special thanks to NJIT Electrical and Computer Engineering (ECE) administrative staff member Monteria Bass for all her support with my Ph.D. academic administrative matters. I wish to express my thanks to Dr. Annan Dong for the technical research discussions we had on two-stage blind source separation and dictionary learning techniques while he was a Ph.D student with the NJIT ECE Center for Wireless Information Processing.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Machine Learning for Blind Source Separation

Blind source separation (BSS) is the process of recovering individual source transmissions from a received mixture of co-channel signals without a-priori knowledge of the channel mixing matrix or transmitted source signals. There are various applications for blind source separation including radio frequency (RF) co-channel signal separation [1]-[4], spectrum sensing for cognitive radio [5]-[7], self-interference cancellation for co-time co-frequency full-duplex systems [8]-[10], speech signal separation also known as the cocktail party problem [11]-[13], musical instrument sound signal separation [14], and in the medical field for separation of electroencephalography (EEG) data that measures electrical neural signal activity in the brain [15], [16].

The BSS problem for co-channel source separation can benefit from machine learning given it is useful for problems that pose a *model deficit* [17]. Blind source separation presents a *model deficit* in that we do not know the wireless channel or transmitted signals. Machine learning provides the ability to use data in the form of training examples to learn a prediction model for regression scenarios in the continuous variable case or a classifier for the discrete output decision space [17], [19]. A model or hypothesis class is proposed for the machine learning problem and the parameters are optimized to fit the example data in such a way that future prediction can be performed on new unseen future data samples [18], [35]. The ability to perform prediction on data outside the training set is known as *generalization* [19], [35]. The training sample set of examples can include

labeled input and output data pairs or include unlabeled inputs only, which is referred to as supervised learning and unsupervised learning, respectively [20].

The BSS problem is well-suited for unsupervised machine learning given the nature of having unlabeled receive data only and the need to learn the transmitted source sequences without knowing the channel or transmit signal waveforms. Given the nature of the problem is known to some degree it is best to use the context of the problem when positing a hypothesis class, which results in an *inductive bias* for the learning algorithm [17], [18]. Utilizing hypothesis classes that exploit sparsity of transmitted sources is an *inductive bias* for solving the blind source separation problem. Without such *inductive bias* machine learning is ill-posed and it is not sufficient to find a solution without some assumptions [18], [20]. This is known as the *no free lunch theorem* where the learning algorithm performs well within our *inductive bias* and not necessarily outside the scope of the problem [17], [20].

There are various known methods that can be utilized to blindly recover latent signals including independent component analysis (ICA) [18]-[23], Least Absolute Shrinkage and Selection Operator (LASSO) [24], and exploitation of cyclostationarity of signal sources [25], [26], that can solve for the latent signals under different assumptions. The ICA approach hinges on the summation of a number of independent signals being Gaussian distributed based on the central limit theorem. ICA then finds an unmixing matrix that maximizes the non-Gaussianity of the projections of the received signal data for source separation. ICA assumes a full rank channel matrix or basis whereby the number of transmitted source signals equals the number of receive channels. Therefore, ICA cannot be applied to an underdetermined system for solving for the latent variables. On the other

hand, LASSO assumes an overcomplete dictionary or channel, and imposes a sparsity constraint on the latent signal coefficients, which ultimately makes it possible to find unique solutions amongst an infinite solution space [24], [27].

LASSO solves half of the blind source recovery problem within a sparse signal representation framework given by $\boldsymbol{\alpha} = \boldsymbol{\Psi\beta}$, where $\boldsymbol{\beta}$ is the sparsest signal vector of non-zero coefficients solved by LASSO, $\boldsymbol{\Psi}$ is the overcomplete dictionary or wireless channel matrix, and $\boldsymbol{\alpha}$ denotes the response vector or received signal vector taken as a snapshot across an antenna array. The received signal data vector denoted by $\boldsymbol{\alpha}$ can be considered either real-valued or complex-valued although in-phase and quadrature components of complex-valued data should be treated as a group as discussed in Section 1.4. The other half of the blind source recovery problem requires a second optimization stage for learning the overcomplete dictionary $\boldsymbol{\Psi}$ using the Method of Optimal Directions (MOD) [28], K-Singular Value Decomposition (K-SVD) [29], Multiple Dictionary Update (MDU) [30], or online block-coordinate descent [31].

The alternating optimization approach between learning the sparse coefficients $\boldsymbol{\beta}$ and dictionary channel matrix $\boldsymbol{\Psi}$ is referred to as a two-stage optimization process [30]. This dissertation derives a one-stage learning approach for blind source separation in bursty or sparse signal RF environments when the transmitted signals and wireless multiple-input multiple-output (MIMO) channel are both unknown at the receiver. The one-stage learning approach solves for the sparse signal coefficients and dictionary in one optimization stage and does not require alternate optimization as performed in the two-stage process. This is accomplished by exploiting the universal function approximation property of neural networks [38], [39].

1.2 Two-Stage Sparse Signal Recovery

Given that the channel and latent sources are both assumed unknown, the blind source separation problem can be formulated as an iterative two-stage convex optimization process for finding the sparse latent signal coefficients and dictionary atoms or channel columns associated with each of the active sources [29]-[32], [35]. Jointly optimizing over the sparse latent signals and dictionary is a non-convex problem, so the two-stage iterative process alternates between solving for the sparse latent signals also known as sparse coding while holding the dictionary fixed and then vice versa for updating the dictionary. Sparsity of signal transmissions can be exploited for detecting intermittent source activity when the system is in fact underdetermined whereby there are actually more sources present than receive sensor elements [32]-[34]. This is possible assuming only a few sources are actually active at any given instance of time over the duration of all intermittent source activity.

A sparse signal representation given by $\boldsymbol{x} = \mathbf{H}\boldsymbol{s}$ implies that an M-dimensional signal $\boldsymbol{x}$ can be modeled as a linear combination of a relatively few number of columns or atoms $\{\mathbf{h}_i\}_{i=1}^{N}$ from an overcomplete dictionary $\mathbf{H}$ of size $M \times N$ where $M < N$ [24], [35], [37]. The sparse coefficient vector $\boldsymbol{s}$ is N-dimensional and contains a relatively few number of non-zero coefficients or k transmit signals defined by $k = \|\boldsymbol{s}\|_0$ where $k << M < N$. The $\ell_0 - \text{norm}$ does not satisfy all axiomatic properties of a norm, but nonetheless provides a count for the number of non-zero coefficients [49]. The goal is to find a unique sparse solution given there are an infinite number of solutions $\boldsymbol{s}$ that solve for $\boldsymbol{x}$ given $\mathbf{H}$. More formally, a sparse coefficient vector $\boldsymbol{s}$ with k non-zero components is considered unique for $k < spark(\mathbf{H})/2$. The $spark$ is defined as the smallest number of linearly dependent columns or atoms from a given matrix [37], [49]. The least upper bound or

supremum given by $spark(\mathbf{H})/2$ is derived via the triangle inequality which holds for complex variables as well [43]. Considering $\mathbf{H}$ to be full row rank, $spark(\mathbf{H}) = M + 1$ and the number of non-zero components of $\boldsymbol{s}$ should satisfy $\|\boldsymbol{s}\|_0 < (\mathrm{M} + 1)/2$.

The ideal constrained optimization problem for solving for the dictionary $\mathbf{H}$ and sparse coefficient vectors $\{\boldsymbol{s}_i\}_{i=1}^{\mathrm{D}}$ for D received signal examples $\{\boldsymbol{x}_i\}_{i=1}^{\mathrm{D}}$ is given in Equation (1.1).

$$\begin{aligned} &\arg\min_{\mathbf{H},\mathbf{S}} \ \|\mathbf{X} - \mathbf{H}\mathbf{S}\|_F^2 \\ &\text{subject to} \quad \|\boldsymbol{s}_i\|_0 \leq k, \quad \forall\, 1 \leq i \leq \mathrm{D} \end{aligned} \tag{1.1}$$

This is known to be a computational intractable problem requiring an exhaustive search over subsets of the dictionary $\mathbf{H}$ and selecting the solution $\boldsymbol{s}_i$ with the smallest number of non-zeros from the set $\{\boldsymbol{s}_i : \boldsymbol{x}_i = \mathbf{H}\boldsymbol{s}_i\}_{i=1}^{\mathrm{D}}$ [27], [37], [49], [52]. The constrained optimization problem in Equation (1.1) is non-convex given it is structured as a joint optimization over the dictionary and sparse representation coefficients. This can be ameliorated by splitting the joint optimization problem into a two-stage optimization process whereby the dictionary is held fixed during optimization over the sparse representation coefficients and vice versa [29]-[32], [35]. To resolve the combinatorial exhaustive search due to the ℓ_0 norm on the sparse representation coefficients an ℓ_1 norm penalty has been proposed instead [24], [27], [35], [37], [49], [52]. The ℓ_1 norm penalty provides convexification of the problem in Equation (1.1) assuming $\mathbf{H}$ is held fixed and enforces the sparse aspect of the solution space [24], [37]. The ℓ_1 norm penalty applies a

constraint on the ordinary least squares estimates, which results in shrinkage of coefficients and zeros out coefficients less than a given threshold also known as soft-thresholding [24], [35], [51], [52]. The ℓ_1 regularized least squares optimization problem or LASSO assuming $\mathbf{H}$ is fixed is given in Equation (1.2).

$$\begin{aligned} &\arg\min_{\mathbf{S}} \ \|\mathbf{X}-\mathbf{H}\mathbf{S}\|_F^2 \\ &\text{subject to} \quad \|\boldsymbol{s}_i\|_1 \le t, \quad \forall\, 1 \le i \le \mathrm{D} \end{aligned} \tag{1.2}$$

The tuning parameter t is a budget on the sum of the absolute values of the coefficients and imposes sparsity on the solution space by shrinking coefficients towards 0 for $t < \|\boldsymbol{s}_{ls}\|_1$ where $\boldsymbol{s}_{ls}$ denotes the ordinary least-squares estimate [24]. The Lagrangian formulation of the LASSO is given in Equation (1.3).

$$\arg\min_{\boldsymbol{s}_i} \frac{1}{\mathrm{M}}\|\boldsymbol{x}_i-\mathbf{H}\boldsymbol{s}_i\|_2^2+\lambda\|\boldsymbol{s}_i\|_1, \quad \forall\, 1 \le i \le \mathrm{D}, \quad \lambda \ge 0 \tag{1.3}$$

The parameter λ controls the level of sparsity or number of non-zero coefficients of the latent signal space $\{\boldsymbol{s}_i\}_{i=1}^{\mathrm{D}}$ [52]. As λ increases there is greater shrinkage imposed on the coefficients $\{\boldsymbol{s}_i\}_{i=1}^{\mathrm{D}}$ and a majority are set to zero based on the uniqueness property via $\|\boldsymbol{s}\|_0 < spark(\mathbf{H})/2 \ll M < N$. For $\lambda = 0$ the resultant estimate of $\{\boldsymbol{s}_i\}_{i=1}^{\mathrm{D}}$ is the minimum ℓ_2 norm solution, and doesn't provide sparse solutions as needed for the blind

source separation problem. The i^{th} sparse coefficient vector solution to the LASSO bound problem in Equation (1.2) is equivalent to the Lagrangian formulation in Equation (1.3) for $t = \|\hat{\boldsymbol{s}}_i(\lambda)\|_1$. Cyclic coordinate descent can be utilized for solving for the sparse coefficient vectors $\{\boldsymbol{s}_i\}_{i=1}^{\mathrm{D}}$ in Equation (1.3) [52]. A more robust and more computationally efficient Lagrangian based approach for sparse coding known as the *alternating direction method of multipliers* (ADMM) is derived in Chapter 3.

After updating the sparse coefficient matrix $\mathbf{S} = \{\boldsymbol{s}_i\}_{i=1}^{\mathrm{D}}$ in Equation (1.3) an optimal solution for the dictionary $\mathbf{H}$ is found by minimizing the residual sum of squares in Equation (1.4). All sparse signal vectors $\{\boldsymbol{s}_i\}_{i=1}^{\mathrm{D}}$ are updated first before proceeding with the next dictionary update stage.

$$\arg\min_{\mathbf{H}\in\mathcal{C}} \|\mathbf{X} - \mathbf{H}\mathbf{S}\|_F^2 \qquad (1.4)$$

$$\text{where} \quad \mathcal{C} = \{\mathbf{H} \in \mathbb{C}^{\mathrm{M}\times\mathrm{N}} : \|\mathbf{h}_i\|_2 = 1, \forall\, 1 \le i \le N\}$$

Many different approaches for dictionary learning have been proposed for optimizing $\mathbf{H}$ while holding the sparse coefficients $\mathbf{S}$ fixed within the two-stage alternating optimization process [28]-[32]. The MOD [28] provides a least-squares update of the channel $\widehat{\mathbf{H}}$ given by $\widehat{\mathbf{H}} = \mathbf{X}\mathbf{S}^{\boldsymbol{H}}(\mathbf{S}\mathbf{S}^{\boldsymbol{H}})^{-\mathbf{1}}$ and is considered to be fairly robust [49]. Other dictionary learning approaches including K-SVD [29] and MDU [30] seek to improve upon the MOD with only moderate improvement upon convergence. The columns or atoms of the dictionary $\mathbf{H}$ are each constrained to be unit-norm to resolve the scaling ambiguity during alternate optimization between the dictionary and sparse coefficients [30], [31],

[53]. The two-stage alternating optimization process between estimating the signal matrix and dictionary matrix is repeated until convergence.

1.3 Undercomplete and Overcomplete Autoencoders

An autoencoder is an unsupervised feedforward neural network with an input layer, hidden layer, and output layer as shown in Figure 1.1 [35], [40], [42], [83]. The autoencoder has an encoder function that converts the input signal into a new representation $\boldsymbol{\xi} = \boldsymbol{f}(\boldsymbol{x})$ and a decoder function that approximately maps the new representation back to the original input signal $\hat{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{\xi})$. The only computational layers are the hidden layer and output layer since the input layer represents the set of examples being fed to the autoencoder. The output layer response is intended to reproduce an estimate of the input signal to the neural network, while performing representation learning at the hidden layer [40], [42].

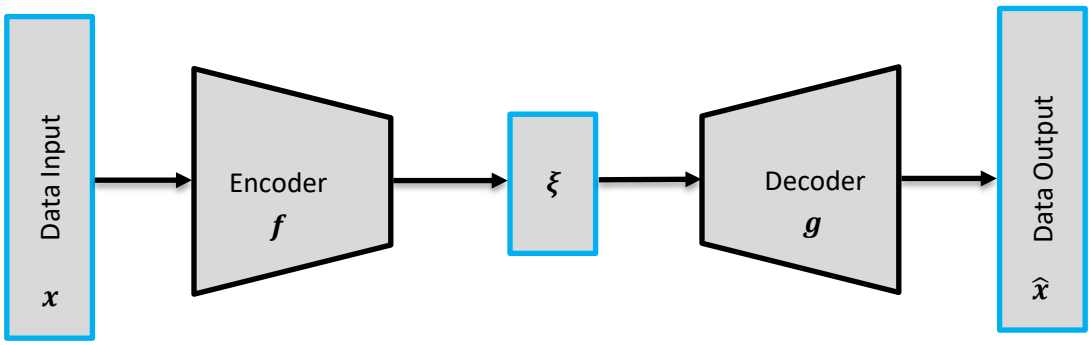

Figure 1.1 Conventional undercomplete autoencoder functional architecture with input $\boldsymbol{x}$, hidden layer encoded output $\boldsymbol{\xi} = \boldsymbol{f}(\boldsymbol{x})$, and decoded output $\hat{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{\xi})$.

The undercomplete autoencoder constrains the hidden layer by having less nodes than the input for dimensionality reduction, which results in a compressed encoded signal representation for learning key features of the input data [35], [40], [42], [83]. An

undercomplete autoencoder is similar to Principle Component Analysis (PCA) in the sense it learns a reduced representation of the data, but autoencoders have the ability to learn a non-linear mapping, which is more powerful than the linear transformation of PCA [42]. On the other hand, an overcomplete autoencoder has a larger hidden layer width than the input layer and output layer for learning a sparse representation of the input data [40], [42]. The autoencoder feedforward neural network has been used for various RF applications including anomaly detection [43], modulation recognition [44], signal classification [45], and even learning a channel encoder and decoder function that matches the same block error rate performance as a communication system with binary phase-shift keying (BPSK) and a Hamming (7,4) code [46].

It is known that feedforward neural networks can provide universal function approximation with at least one hidden layer in a neural network [38], [39]. The expressive power of neural networks is exploited in this dissertation for providing a one-stage learning solution for blind source separation within a sparse autoencoder framework without requiring separate alternate optimization between the sparse coefficients and dictionary channel matrix [40]. Furthermore, the sparse autoencoder is able to generalize efficiently to data outside the training set and produce sparse code representations at the output of the encoder without requiring additional optimization steps as is the case for the two-stage sparse coding process as described in Section 1.2 [81], [82], [92].

1.4 Sparse Coding of Complex-Valued Data

Sparse coding of complex-valued data can be formulated in two different ways. Either directly in the complex domain or in the real domain via a mapping from complex-valued

data to real-data [55]. That is, complex-valued signals can be mapped from $\mathbb{C}^N \rightarrow \mathbb{R}^{2N}$ and sparse coding can be applied to the real and imaginary parts of the complex numbers separately [46], [56], [61], [62]. Therefore, an *N*-dimensional complex space is transformed into a 2*N*-dimensional real space. This is accomplished by reformulating the sparse coding and dictionary learning problem $\boldsymbol{x_c} = \mathbf{H_c}\boldsymbol{s_c}$ as defined in Equation (1.5), where the subscript c denotes complex-valued data.

$$\begin{bmatrix} \mathrm{Re}(\boldsymbol{x_c}) \\ \mathrm{Im}(\boldsymbol{x_c}) \end{bmatrix} = \begin{bmatrix} \mathrm{Re}(\mathbf{H_c}) & -\mathrm{Im}(\mathbf{H_c}) \\ \mathrm{Im}(\mathbf{H_c}) & \mathrm{Re}(\mathbf{H_c}) \end{bmatrix} \begin{bmatrix} \mathrm{Re}(\boldsymbol{s_c}) \\ \mathrm{Im}(\boldsymbol{s_c}) \end{bmatrix} \tag{1.5}$$

Processing complex-valued data as shown in Equation (1.5) is done quite often due to the lack of available software packages that support complex-valued neural network algorithms and activation functions [47], [59]. On the other hand, it is particularly important to maintain the in-phase and quadrature pair groupings [54] during soft-thresholding as shown for the complex LASSO approach in [55] and not apply ℓ_1 regularized least squares for real and imaginary components separately. Applying sparse coding to data that has been transformed to the real space as defined in Equation (1.5) results in independent soft-thresholding of the real and imaginary components of the complex numbers. For sparse coding applications it is imperative that complex-valued data be processed fully in the complex domain to avoid losing phase information. Processing complex-valued data after mapping to real data as in Equation (1.5) will result in independent shrinkage of the ordered pairs of complex numbers, which should be set to zero or non-zero simultaneously. A fully complex sparse autoencoder is derived in Chapter

5 that performs sparse coding at the hidden layer while maintaining the complex-valued in-phase and quadrature data without performing separate processing on the real and imaginary parts of the complex-valued data as in [46], [57] [61]-[63].

1.5 Optimization of Complex-Valued Data

In order to fully process complex-valued data through the complex-valued sparse autoencoder without mapping complex numbers to real and imaginary parts as defined in Equation (1.5), the hidden layer function $\boldsymbol{\xi} = \boldsymbol{f}(\boldsymbol{x})$ must support complex numbers and optimization of the encoder and decoder weights must be performed in the complex domain [58], [60]. The optimization of complex-valued weights is carried out using Wirtinger Calculus [64], [65], which provides a complex-valued differentiable operator that satisfies the partial derivatives of the real and imaginary parts of a complex number as defined in Equations (1.7) and (1.8).

In general, let $f(z)$ be a complex-valued function of a complex variable z given by $f(z)$: $\mathbb{C} \rightarrow \mathbb{C}$, where $z = a + ib$ and $a, b \in \mathbb{R}$. $f(z)$ can be further defined in terms of it's real and imaginary parts as $f(z) = u(a,b) + iv(a,b)$, where $u(a,b)$ is the real part and $v(a,b)$ is the imaginary part of $f(z)$. By defining a and b as a function of z as given in Equation (1.6), $f(z)$ can be rewritten as $f(a(z), b(z))$. z^* in Equation (1.6) is the complex conjugate of the complex variable z.

$$a = \frac{z + z^*}{2} \qquad\qquad b = \frac{z - z^*}{2i} \tag{1.6}$$

By applying the chain rule to $f(a(z), b(z))$ and differentiating with respect to z and z^* results in the two expressions in Equations (1.7) and (1.8), respectively [66].

$$\frac{\partial f}{\partial \mathrm{z}} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial \mathrm{z}} + \frac{\partial f}{\partial b}\frac{\partial b}{\partial \mathrm{z}} = \frac{1}{2}\left(\frac{\partial f}{\partial a} - i\frac{\partial f}{\partial b}\right) \tag{1.7}$$

$$\frac{\partial f}{\partial \mathrm{z}^*} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial \mathrm{z}^*} + \frac{\partial f}{\partial b}\frac{\partial b}{\partial \mathrm{z}^*} = \frac{1}{2}\left(\frac{\partial f}{\partial a} + i\frac{\partial f}{\partial b}\right) \tag{1.8}$$

The differential operator in Equation (1.7) with respect to z^* is utilized for complex optimization of the weights of the sparse autoencoder in Chapter 5, where a cost function or loss $\mathcal{L}(z)$ is defined as a real-valued function over a domain of complex-valued variables given in general by the mapping $\mathcal{L}(z)$: $\mathbb{C} \to \mathbb{R}$.

1.6 Organization of this Dissertation

The blind source separation system model is described in Chapter 2. Chapter 3 derives the two-stage blind source separation approach based on the ADMM algorithm for real-valued and complex-valued data. Chapter 4 and Chapter 5 describe the $\ell_1$norm regularized sparse autoencoder for blind source separation for real-valued data and complex-valued data, respectively. Chapter 6 describes generalization for regression problems and a data-dependent generalization bound based on the Rademacher complexity for the one-stage learning blind source separation problem. Finally, the conclusion is provided in Chapter 7.

CHAPTER 2

SYSTEM MODEL

2.1 Multiple-Input Multiple Output System

The blind source separation system model formulation assumes there are N independent transmitter sources and M received signals across an antenna array or distribution of M sensor elements. A linear time-invariant flat fading channel is assumed whereby the channel gains are represented as complex Gaussian random variables with zero-mean [67], [69]. This represents a rich scattering environment also commonly referred to as a Rayleigh fading channel based on the magnitude of the complex Gaussian random variables being Rayleigh distributed [68]. The flat fading channel assumption refers to the transmitted signal bandwidth being within the coherence bandwidth or inverse delay spread of the channel and thus, the received signal does not experience frequency selective fading or intersymbol interference (ISI) [70].

Figure 2.1 depicts a multiple-input multiple output (MIMO) blind source separation scenario where there are only a sparse number of active sources at any given time that are received over the MIMO channel. All sources are considered to be transmitting at the same frequency creating a co-channel mixture of sources at the receive array. The received signal snapshot across the antenna array denoted by $\boldsymbol{x}(n)$ is defined in Equation (2.1).

$$\boldsymbol{x}(n) = \sqrt{\gamma}\mathbf{H}\boldsymbol{s}(n) + \mathbf{v}(n) \qquad n = 1,\ldots,\mathrm{D} \qquad (2.1)$$

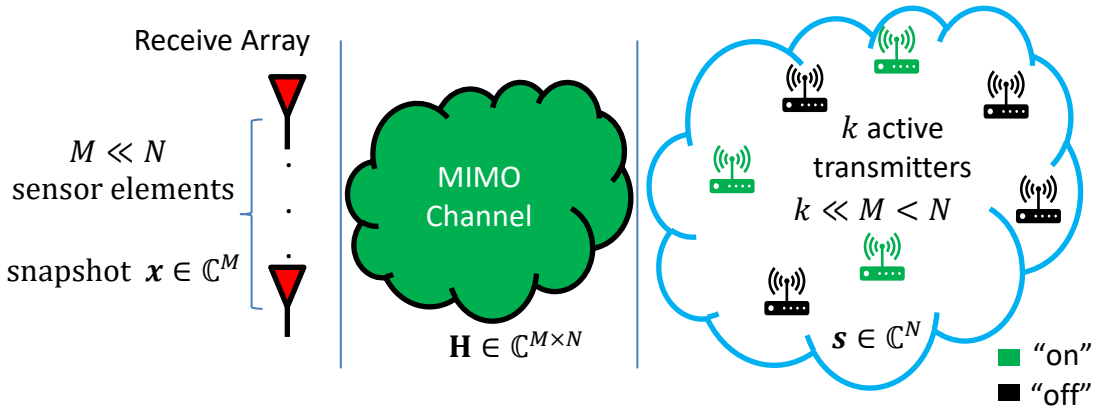

Figure 2.1 Illustration of a blind source separation (BSS) scenario with a sparse number of active sources k received over a MIMO channel with M receive antenna elements where $k << M < N$ and N represents the total number of potential independent co-channel transmitters.

It is assumed that snapshots of data $\boldsymbol{x}(n)$ are taken across a synchronized antenna array or distribution of M sensors where n denotes a particular snapshot over time for $n = 1, \ldots, \mathrm{D}$. $\boldsymbol{s}(n)$ denotes the sparse signal vector of active sources, $\mathbf{H}$ is the wireless MIMO channel, and γ is the signal-to-noise ratio (SNR). For the n^{th} snapshot the number of signal sources active within the signal vector $\boldsymbol{s}(n)$ is given by $k = \|\mathbf{s}(n)\|_0$ where $k << M < N$. Hence, the signal activity is considered sparse relative to the dictionary or channel matrix $\mathbf{H} \in \mathbb{C}^{M \times N}$, and $\boldsymbol{x}(n)$ is a sparse representation or sparse linear combination of $\mathbf{H}$. $\boldsymbol{\nu}(n)$ is complex Gaussian noise with zero-mean and unit variance denoted by $\boldsymbol{\nu}(n) \sim \boldsymbol{\mathcal{N}}_{\mathbb{C}}(\mathbf{0}, \boldsymbol{I})$, where $\boldsymbol{I}$ is the identity covariance matrix.

The received signal model over D snapshots is given in matrix form in Equation (2.2).

$$\mathbf{X} = \sqrt{\gamma}\mathbf{H}\mathbf{S} + \mathbf{N} \tag{2.2}$$

The receive signal matrix is defined as $\mathbf{X} \in \mathbb{C}^{M \times \mathrm{D}}$, $\mathbf{H}$ is an $M \times N$ matrix of complex Gaussian elements with zero-mean and unit variance, $\mathbf{S}$ is a sparse matrix of source transmissions, and $\mathbf{N} \in \mathbb{C}^{M \times \mathrm{D}}$ is a matrix of zero-mean unit variance complex Gaussian noise elements $\nu_{ij} \sim \mathcal{N}_{\mathbb{C}}(0,1)$. The SNR term γ is considered to be scaled by the number of active sources $\|\boldsymbol{s}(n)\|_0$, so that γ defines the true SNR of the received signal as k fluctuates over D snapshots for $n = 1, \ldots, \mathrm{D}$ [71].

2.2. Source Activity Model

The signal source transmit data denoted by $\mathbf{S}$ with rows representing different sources is considered intermittent over time and is common with short bursty communication activity due to intermittent speech activity [74], mobile communications with discontinuous transmission to preserve mobile handset power [75], or the Internet of Things for smart city communications [76], [77]. Modeling time series data or intermittent source activity can be synthesized using a hidden Markov model (HMM) [74], [78]. A first-order hidden Markov model is implemented per transmission source and is shown in Figure (2.2).

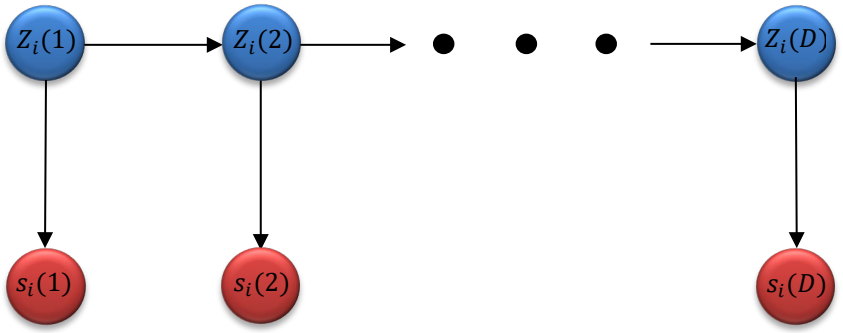

Figure 2.2 First-order HMM for i^{th} transmitter source activity. The hidden states $Z_i(n)$ represent the n^{th} state of the i^{th} transmitter and $s_i(n)$ denotes the observation output.

The hidden states $Z_i(n)$ represent the state of the i^{th} source being either ON or OFF for the nth snapshot. Whether or not the i^{th} source is actively transmitting at any given time is based on a two-state Markov chain with state space $\mathcal{S} = \{\text{OFF}, \text{ON}\}$. The state transition probability graph for the hidden states of each i^{th} source is shown in Figure 2.3 [79].

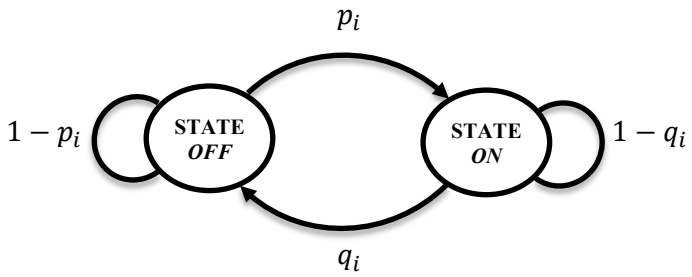

Figure 2.3 Transition probability graph of a two-state Markov chain for the i^{th} source activity.

Each state is modeled as a Bernoulli process where a state change from OFF to ON or ON to OFF takes place upon a success conditioned on the current state. The state-transition probability from the i^{th} source activity off-state to on-state is denoted by $p_i =$

$Pr(Z_i(\mathrm{n}) = ON \mid Z_i(\mathrm{n}-1) = OFF)$ and from the on-state to off-state is given by $q_i = Pr(Z_i(\mathrm{n}) = OFF \mid Z_i(\mathrm{n}-1) = ON)$ [79]. The steady-state probability of a source being in the ON state or OFF state is denoted by π_{ON} and π_{OFF}, respectively. The steady-state probabilities π_{ON} and π_{OFF} are defined in Equation (2.3) [79].

$$\pi_{ON} = \frac{p_i}{p_i + q_i} \qquad \pi_{OFF} = \frac{q_i}{p_i + q_i} \tag{2.3}$$

The average number of active sources ζ is based on the total number of active sources over D snapshots and the steady-state probability of each source being in an active state, and is given by $\zeta = N * \pi_{ON}$, where N denotes the maximum number of sources. The average transmission duration of the i^{th} source is based on the inverse of the mean of a geometric random variable and is given by ${q_i}^{-1}$ [79].

CHAPTER 3

TWO-STAGE LEARNING FOR BLIND SOURCE SEPARATION

3.1 ADMM LASSO Sparse Coding for Blind Source Separation

The two-stage blind source separation approach alternates between a sparse coding stage based on a fixed channel estimate and a channel estimation or dictionary learning stage while holding the sparse code estimates fixed as described in general in Section 1.2 [32]. The co-channel signal sources defined in Chapter 2 are separated using the sparse coding alternating direction method of multipliers (ADMM) optimization algorithm. The ADMM LASSO algorithm for BSS is based on minimizing the augmented Lagrangian in Equation (3.1) [52].

$$L_\rho(\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\mu}) = \frac{1}{M}\|\boldsymbol{x} - \mathbf{H}\boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{w}\|_1 + (\boldsymbol{z} - \boldsymbol{w})^H\boldsymbol{\mu} + \rho\|\boldsymbol{z} - \boldsymbol{w}\|_2^2 \tag{3.1}$$

Optimization of the augmented Lagrangian cost function in Equation (3.1) with respect to $(\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\mu})$ is performed using the ADMM algorithm by successively minimizing $\boldsymbol{z}$ and $\boldsymbol{w}$ followed by a dual ascent update of the Lagrange multiplier vector $\boldsymbol{\mu}$ [52]. The ADMM updates for iterations $t = 0, 1, 2, \ldots$ are defined in Equations (3.2a)-(3.2c) for real-valued data.

$$\boldsymbol{z}^{t+1} = \underset{\boldsymbol{z}\in\mathbb{R}^N}{\arg\min}\, L_\rho(\boldsymbol{z}, \boldsymbol{w}^t, \boldsymbol{\mu}^t) \tag{3.2a}$$

$$\boldsymbol{w}^{t+1} = \underset{\boldsymbol{w}\in\mathbb{R}^N}{\arg\min}\, L_\rho(\boldsymbol{z}^{t+1}, \boldsymbol{w}, \boldsymbol{\mu}^t) \tag{3.2b}$$

$$\boldsymbol{\mu}^{t+1} = \boldsymbol{\mu}^t + \rho(\boldsymbol{z}^{t+1} - \boldsymbol{w}^{t+1}) \tag{3.2c}$$

The ADMM iterative updates defined in Equations (3.2a)-(3.2c) are derived in Equations (3.3a)-(3.3c). Minimization of $L_\rho(\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\mu})$ with respect to $\boldsymbol{z}$ provides a ridge regression update for $\boldsymbol{z}$ in Equation (3.3a) and minimization with respect to $\boldsymbol{w}$ involves a soft-threshold update in Equation (3.3b). The sparse signal source vectors $\{\boldsymbol{s}(n)\}_{n=1}^{\mathrm{D}}$ in Equation (2.1) are estimated by $\{\boldsymbol{w}(n)\}_{n=1}^{\mathrm{D}}$ in Equation (3.3b). The Lagrange multiplier vector update $\boldsymbol{\mu}^{t+1}$ is updated based on the new updated iterations of vectors $\boldsymbol{z}^{t+1}$ and $\boldsymbol{w}^{t+1}$. ρ is considered a fixed parameter where $\rho > 0$ and the quadratic augmented Lagrangian term involving ρ penalizes solutions that violate the constraint outside the feasible region [85].

$$\boldsymbol{z}^{t+1} = \left(\frac{1}{M}\mathbf{H}^{\mathrm{T}}\mathbf{H} + \rho\mathrm{I}\right)^{-1}\left(\frac{1}{M}\mathbf{H}^{\mathrm{T}}\boldsymbol{x} + \rho\boldsymbol{w}^t - \boldsymbol{\mu}^t\right) \tag{3.3a}$$

$$w_i^{t+1} = sign\left(z_i^{t+1} + \frac{\mu_i^t}{\rho}\right)\left(\left|z_i^{t+1} + \frac{\mu_i^t}{\rho}\right| - \frac{\lambda}{\rho}\right)_+ \quad for\ i = 1,\ \ldots,\ N \tag{3.3b}$$

$$\boldsymbol{\mu}^{t+1} = \boldsymbol{\mu}^t + \rho(\boldsymbol{z}^{t+1} - \boldsymbol{w}^{t+1}) \tag{3.3c}$$

The soft-threshold operator in Equation (3.3b) shrinks the absolute value term by λ/ρ and the operator $(\psi)_+$ is set to ψ for $\psi > 0$ and equals zero for $\psi \leq 0$. Hence, a change in ρ impacts the optimal value of the hyperparameter λ given the ratio λ/ρ has an effect on the shrinkage and ultimately the sparse coding solution. The hyperparameter λ is optimized using cross-validation, which is described in Chapter 4.

Optimization of the ADMM LASSO augmented Lagrangian in Equation (3.1) for complex variables is performed using Wirtinger calculus as defined in Section 1.5. The ADMM LASSO updates for complex variables is defined in Equations (3.4a) and (3.4b). The Lagrange multiplier vector update $\boldsymbol{\mu}^{t+1}$ is as defined in Equation (3.3c).

$$\boldsymbol{z}^{t+1} = \underset{\boldsymbol{z}^* \in \mathbb{C}^N}{\arg\min}\, L_\rho(\boldsymbol{z}, \boldsymbol{w}^t, \boldsymbol{\mu}^t) \tag{3.4a}$$

$$\boldsymbol{w}^{t+1} = \underset{\boldsymbol{w}^* \in \mathbb{C}^N}{\arg\min}\, L_\rho(\boldsymbol{z}^{t+1}, \boldsymbol{w}, \boldsymbol{\mu}^t) \tag{3.4b}$$

Minimization of Equations (3.4a) and (3.4b) with respect to the complex conjugates $\boldsymbol{z}^*$ and $\boldsymbol{w}^*$ results in the updates $\boldsymbol{z}^{t+1}$ and $\boldsymbol{w}^{t+1}$ in Equations (3.5a) and (3.5b), respectively. Equation (3.3a) for the real-valued case differs from the complex-valued case in Equation (3.5a) by taking the adjoint or Hermitian of $\mathbf{H}$ in Equation (3.5a) and not just the transpose of $\mathbf{H}$ as in Equation (3.3a).

$$\boldsymbol{z}^{t+1} = \left(\frac{1}{\mathrm{M}}\mathbf{H}^H\mathbf{H} + \rho\mathrm{I}\right)^{-1}\left(\frac{1}{\mathrm{M}}\mathbf{H}^H\mathbf{x} + \rho\boldsymbol{w}^t - \boldsymbol{\mu}^t\right) \tag{3.5a}$$

$$w_i^{t+1} = \frac{z_i^{t+1} + \frac{\mu_i^t}{\rho}}{\left|z_i^{t+1} + \frac{\mu_i^t}{\rho}\right|}\left(\left|z_i^{t+1} + \frac{\mu_i^t}{\rho}\right| - \frac{\lambda}{\rho}\right)_+ \quad for\ i = 1,\ \ldots,\ N \tag{3.5b}$$

3.2 Dictionary Learning for Channel Estimation

The two-stage learning algorithm performs dictionary learning or channel estimation after all sparse code vectors in matrix $\mathbf{S}$ are updated for iteration $t+1$. There are various dictionary learning algorithms that build off of the method of optimal directions, which provides a least-squares update of the channel as described in Section 1.2 [28], [29]. The MOD algorithm has been shown to be a robust tradeoff for dictionary learning in comparison to other methods such as the K-SVD algorithm with less computational complexity [98]. The MOD algorithm is defined in Equation (3.6) and provides a least-

squares update $\widehat{\mathbf{H}}$ of the channel $\mathbf{H}$ in Equation (2.1). The Hermitian of $\mathbf{S}$ for complex-valued data defaults to the transpose when applied to real-valued data in Equation (3.6).

$$\widehat{\mathbf{H}} = \underset{\mathbf{H}\in\mathbb{C}^{M\times N}}{\arg\min} \|\mathbf{X} - \mathbf{HS}\|_F^2 = \mathbf{XS}^{\boldsymbol{H}}(\mathbf{SS}^{\boldsymbol{H}})^{-\mathbf{1}}, \quad where\ \|\mathbf{h}_i\|_2 = 1, \forall\ 1 \leq i \leq N \tag{3.6}$$

Signal recovery performance results for the ADMM LASSO are compared against the sparse autoencoder for real-valued data and complex-valued data as described in Chapters 4 and 5, respectively.

CHAPTER 4

SPARSE AUTOENCODER FOR REAL-VALUED SIGNALS

4.1 Real-Valued Sparse Autoencoder for Source Separation

Autoencoders are known to provide feature extraction at the hidden layer of an artificial neural network that provides a code representation of the input data while reconstructing the input data at the decoder output of the autoencoder [42]. Sparse autoencoders or regularized overcomplete autoencoders have a hidden layer width greater than the input layer and impose regularization on the weights or hidden layer output to enable sparse coding with a small number of non-zeros at the encoder hidden layer output [40], [81], [92], [93]. A sparse autoencoder with an ℓ_1 norm penalty on the encoder weights and a sparsifying logistic sigmoid activation function for the hidden layer was utilized for feature extraction of unique representations of handwritten numerals and natural image patches, which were then used for supervised training of a neural network classifier [92]. A rectified linear unit (ReLU) activation function rectifier$(x) = max(0, x)$ [40] was used within a sparse autoencoder framework with an ℓ_1 norm penalty applied to the hidden layer output for image and text classification and shown to produce actual zeros at the hidden layer output [81]. Another method for promoting sparsity within an autoencoder framework is to utilize a Kullback-Leibler (KL) divergence penalty term between a small target activation percentage for a hidden layer neuron and the mean activation over training examples, which was demonstrated for image classification in [94].

An ℓ_1 regularized sparse autoencoder feedforward neural network is applied to the blind source separation problem for recovering real-valued independent co-channel

sources. In addition to an ℓ_1 norm penalty of the hidden layer output added to the mean-squared loss function between the input data and reconstructed output, a soft-threshold operator activation function is utilized within the hidden layer to promote sparse coding [82]. The complex-valued sparse autoencoder learning model is addressed in Chapter 5, which is a non-trivial extension of the real-valued case due to the need to maintain in-phase and quadrature pair groupings of complex-valued signals throughout the neural network for sparse coding as described in Section 1.4. The blind source separation sparse autoencoder architecture is shown in Figure 4.1.

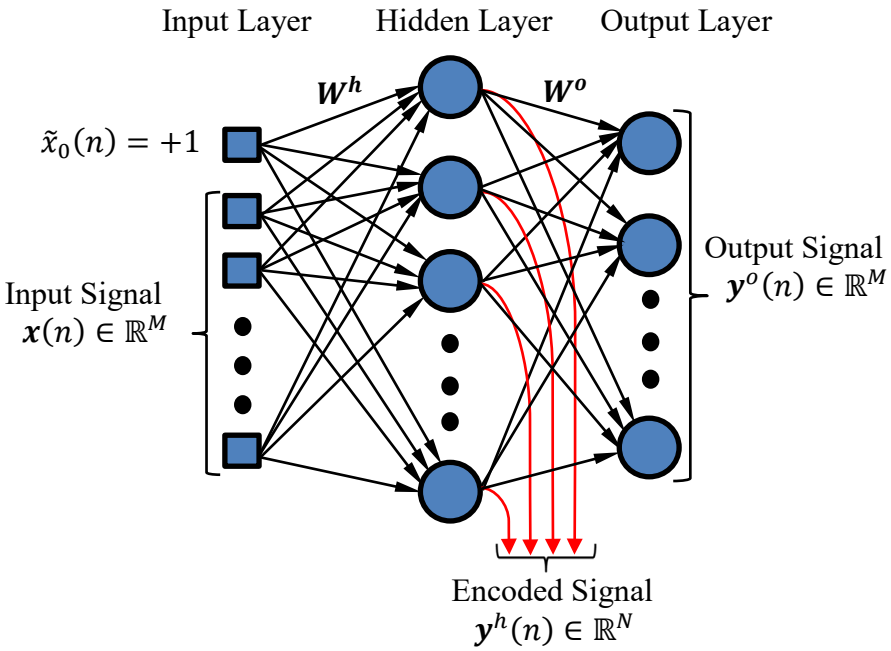

Figure 4.1 Blind sparse autoencoder feedforward neural network architecture with $N > M$ hidden layer nodes. The encoded signal output at the hidden layer provides a sparse representation of the transmitted sources.

The input signal to the sparse autoencoder in Figure 4.1 is denoted by $\boldsymbol{x}(n) \in \mathbb{R}^M$, which represents the n^{th} received signal snapshot across an antenna array with M elements as defined in Chapter 2, but for real-values only in this chapter. The real-valued scenario is considered valid for real-valued modulation types such as binary phase-shift keying [80]. The received signal snapshot model for $\boldsymbol{x}(n)$ is defined in Equation (4.1) for real-valued data, where the channel $\mathbf{H}$ is an $M \times N$ matrix with zero-mean unit-variance Gaussian random variables, $\boldsymbol{v}(n) \in \mathbb{R}^M$ is a noise vector of zero-mean unit-variance Gaussian random variables, and $\boldsymbol{s}(n)$ is the signal vector of sparse source transmissions or baseband symbols. γ is defined as the SNR.

$$\boldsymbol{x}(n) = \sqrt{\gamma}\mathbf{H}\boldsymbol{s}(n) + \boldsymbol{v}(n) \qquad n = 1,\ldots,\mathrm{D} \qquad (4.1)$$

The encoded signal output at the hidden layer of Figure 4.1 denoted by $\boldsymbol{y}^h(n)$ produces the sparse latent signal vectors $\boldsymbol{s}(n)$ up to a permutation and sign ambiguity [53]. Finally, the output signal $\boldsymbol{y}^o(n)$ of the output layer of the sparse autoencoder in Figure 4.1 provides an estimate of the input to the autoencoder $\boldsymbol{x}(n)$. The sparse autoencoder acts as a replicator network, while learning a representation at the hidden layer that explains the unique features of the data [42], [81].

The *inductive bias* of the learning model or assumptions being made for selecting a sparse learning algorithm is that transmitted sources experience intermittent activity and sparsity can be exploited for solving for the latent signal sources [74]-[77]. The BSS sparse autoencoder in Figure 4.1 enforces sparsity on the latent signal space in three ways. First

the cost function in Equation (4.2) imposes sparsity on the encoded output of the hidden layer during training via ℓ_1 regularization of the hidden layer outputs, second the hidden layer is constructed to be wider than the input and output layers forcing the autoencoder to learn a sparse representation given the overcomplete structure of the output layer weight matrix, and third the activation function at the hidden layer inherently performs soft-thresholding resulting in hidden layer nodes with zero output based on the shrinkage operator [82].

The cost function or loss function for optimizing the weights of the sparse autoencoder in Figure 4.1 is given in Equation (4.2).

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2|B|}\sum_{n\in B}\|\boldsymbol{d}(n) - \boldsymbol{y}^o(n)\|_2^2 + \lambda\|\boldsymbol{y}^h(n)\|_1 \tag{4.2}$$

In Equation (4.2) $\boldsymbol{d}(n)$ is the desired response, which is equal to the input snapshot $\boldsymbol{x}(n) \in \mathbb{R}^M$. $\boldsymbol{y}^o(n) \in \mathbb{R}^M$ is the output signal at the output layer of the autoencoder feedforward neural network, $\boldsymbol{y}^h(n) \in \mathbb{R}^N$ is the encoded sparse representation output of the hidden layer, λ is the sparsity penalty parameter, and n denotes the iteration or time-step. The summation in Equation (4.2) is taken over a batch of examples of size $|B|$, where $|B|$ denotes the cardinality of set B. The loss function $\mathcal{L}(\boldsymbol{W})$ is a function of all weights in the neural network denoted by $\boldsymbol{W}$, which represents the hidden layer and output layer weights given by $\boldsymbol{W}^h$ and $\boldsymbol{W}^o$, respectively. The sum of the squared errors is minimized with respect to the synaptic weights $\boldsymbol{W}$ of the feedforward neural network while imposing an ℓ_1

norm penalty on the hidden layer output. The weights of the entire neural network are learned using the received signal snapshots $\{\boldsymbol{x}(n)\}_{n=1}^{\mathrm{D}}$ as unlabeled input examples only.

The output layer signal vector $\boldsymbol{y}^o(n)$ contains M neural network node outputs and is given in Equation (4.3) as a function of the output layer activation function $\varphi_o(\cdot)$. The output layer activation function $\varphi_o(\cdot)$ is a linear identity function with no non-linearity. $\boldsymbol{v}^o(n)$ is the activation potential vector for the output layer and is defined in Equation (4.4).

$$\boldsymbol{y}^o(n) = \varphi_o(\boldsymbol{v}^o(n)) \tag{4.3}$$

$$\boldsymbol{v}^o(n) = \boldsymbol{W}^o(n)\boldsymbol{y}^h(n) \tag{4.4}$$

The hidden layer output vector $\boldsymbol{y}^h(n)$ contains N nodes and is defined in Equation (4.5) as a function of the hidden layer activation function $\varphi_h(\cdot)$. $\varphi_h(\cdot)$ is a soft-threshold operator or shrinkage function and is defined in Equation (4.6). The operator $(\psi)_+$ in Equation (4.6) is defined as $(\psi)_+ = \max(\psi, 0)$, which is equal to ψ for $\psi > 0$. The hidden layer activation potential $\boldsymbol{v}^h(n)$ is defined in Equation (4.7).

$$\boldsymbol{y}^h(n) = \varphi_h\left(\boldsymbol{v}^h(n)\right) \tag{4.5}$$

$$\varphi_h\left(v_i^h(n)\right) = sign(v_i^h(n))(\left|v_i^h(n)\right| - \lambda)_+ \tag{4.6}$$

$$\boldsymbol{v}^h(n) = \boldsymbol{W}^h(n)\tilde{\boldsymbol{x}}(n) \tag{4.7}$$

$\boldsymbol{W}^o \in \mathbb{R}^{M\times N}$ and $\boldsymbol{W}^h \in \mathbb{R}^{N\times M+1}$ denote the weight matrices for the output layer and hidden layer, respectively. Note that the first column of $\boldsymbol{W}^h$ represents the bias terms for each neuron in the hidden layer. This effectively results in an affine transformation of the linear combined output between the input signal and weights per neuron. $\tilde{\boldsymbol{x}}(n) \in \mathbb{R}^{M+1}$ is the input vector example to the neural network for the n^{th} snapshot across a spatial array of M antenna elements as defined in Equation (4.1) with the first element $\tilde{x}_0(n) = +1$ to account for the bias weight term per hidden neuron.

The weights of the ℓ^{th} layer of the neural network are updated using mini-batch stochastic gradient descent with adaptive moments (ADAM) [84]. The gradient of the loss function with respect to the output and hidden layer weights is derived via the backpropagation algorithm [83]. The gradient of the loss function with respect to the output weights in Equation (4.2) is defined in Equation (4.8).

$$\frac{\partial \mathcal{L}(W)}{\partial w_{kj}^{(o)}(n)} = \frac{1}{|B|}\sum_{n\in B} \delta_k^o(n)\, y_j^h(n) \tag{4.8}$$

$\delta_k^o(n)$ in Equation (4.8) is considered the local gradient of the loss function in Equation (4.2) with respect to the activation potential $v_k^o(n)$ of the k^{th} neuron in the output layer of the neural network. $\delta_k^o(n)$ is defined in Equation (4.9)

$$\delta_k^o(n) = -[d_k(n) - y_k^o(n)]\varphi_o'(v_k^o(n)) \tag{4.9}$$

The gradient of the loss function in Equation (4.2) with respect to the hidden layer weights is defined in Equation (4.10).

$$\frac{\partial \mathcal{L}(W)}{\partial w_{ji}^{(h)}(n)} = \frac{1}{|B|}\sum_{n\in B}\delta_j^h(n)\tilde{\mathrm{x}}_i\,(n) \tag{4.10}$$

The local gradient of the loss function in Equation (4.2) with respect to the activation potential $v_j^h(n)$ of the j^{th} hidden layer node is denoted by $\delta_j^h(n)$ and is given in Equation (4.11).

$$\delta_j^h(n) = \varphi_h'\left(v_j^h(n)\right)\sum_{k=1}^{\mathrm{M}}\delta_k^o(n)w_{kj}^o(n) + \lambda\cdot sign(y_j^h(n))\varphi_h'\left(v_j^h(n)\right) \tag{4.11}$$

The gradient of the ℓ_1 penalty term $\lambda\|\boldsymbol{y}^h(n)\|_1$ in Equation (4.2) with respect to the activation potential $v_j^h(n)$ is $\lambda \cdot sign(y_j^h(n))\varphi_h'\left(v_j^h(n)\right)$, which is undefined for an activation output $y_j^h(n)$ of zero. On the other hand, given $\varphi_h'\left(v_j^h(n)\right)$ is zero for $v_j^h(n) \leq \lambda$, where $\varphi_h'(\cdot)$ is the derivative of the activation function at the hidden layer, $\lambda \cdot sign(y_j^h(n))\varphi_h'\left(v_j^h(n)\right)$ is zero for $y_j^h(n) = 0$.

4.2 Hyperparameter Selection

The tuning parameter λ in Equation (4.2) imposes sparsity on the hidden layer weights and is optimized using *K*-fold cross-validation (CV). The hyperparameter of the soft-threshold activation function is set equal to the tuning parameter λ of the ℓ_1 norm penalty on the hidden layer output and optimized together, which satisfies the non-differentiable case for $\left\|\boldsymbol{y^h}(n)\right\|_1$ when $\boldsymbol{y^h}(n)$ is zero as explained in Section 4.1. The dataset of size D is partitioned into $\eta = {}^{\mathrm{D}}/_{K}$ disjoint sets or folds. $K-1$ sets are used for training the weights **W** and 1 out of *K* sets is used for computing the cross-validation error as shown in Figure (4.2) for 5-Fold CV [87].

Validation	Training	Training	Training	Training
Training	Validation	Training	Training	Training
.		.		.
.		.		.
.		.		.
Training	Training	Training	Training	Validation

Figure 4.2 5-Fold cross-validation partitioning of data into validation and training sets.

Optimization of the weights in Equation (4.2) is performed K times for all K permutations of training data in Figure (4.2). Training with different data subsets produces a different predictor or set of weights $\mathbf{W}$ and hence, a different validation error for each i^{th} fold out of K folds. All K validation errors are averaged together to approximate the generalization error of the sparse autoencoder, which is referred to as the cross-validation error. Likewise, the training errors pertaining to the K permutations of training data are averaged together to compute an expected value or average value for the training error.

The samples pertaining to the i^{th} fold are denoted by $\{\boldsymbol{x}_{ij}\}_{j=1}^{J}$. Assume that the weights $\mathbf{W}$ of the ℓ_1 norm regularized loss function in Equation (4.2) are trained on all data except for the i^{th} fold producing a predictor $\boldsymbol{y}_i^o$. The cross-validation error is defined in Equation (4.12), where $L(\boldsymbol{d}_{ij}, \boldsymbol{y}_{ij}^o)$ is any loss function in general. $\boldsymbol{d}_{ij}$ is the j^{th} sample of the desired response within the i^{th} fold and $\boldsymbol{y}_{ij}^o$ is the predicted output for the j^{th} sample of the i^{th} fold validation data.

$$\hat{R}_{CV} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{J}\sum_{j=1}^{J} L(\boldsymbol{d}_{ij}, \boldsymbol{y}_{ij}^o) \tag{4.12}$$

The sparse autoencoder performance is compared to the alternating direction method of multipliers (ADMM) LASSO algorithm for BSS described in Chapter 3. Cross-validation is utilized for computing the hyperparameter or tuning parameter λ in Equation (3.1), which is rewritten in Equation (4.13) for convenience.

$$L_\rho(\boldsymbol{z},\boldsymbol{w},\boldsymbol{\mu}) = \frac{1}{M}\|\boldsymbol{x}-\boldsymbol{H}\boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{w}\|_1 + (\boldsymbol{z}-\boldsymbol{w})^H\boldsymbol{\mu} + \rho\|\boldsymbol{z}-\boldsymbol{w}\|_2^2 \tag{4.13}$$

$\boldsymbol{\rho}$ was set to a fixed value equal to 2, while the tuning parameter λ was optimized over a set of values using cross-validation.

A total of $\mathrm{D} = 1000$ samples was used for cross-validation with $K = 5$. The tuning parameter λ ranged over the set $\lambda \in \{.01, \ldots, 3\}$. The total number of co-channel signals over D samples is set equal to 20. The state-transition probabilities p_i and q_i defined in Section 2.2 were set such that an average number of 3 sources are overlapping in time out of the 20 potential co-channel sources.

The ADMM LASSO expected support recovery error based on the training data and validation data versus the hyperparameter λ is shown in Figure 4.3 for SNR values of 15 dB, 20 dB, 25 dB, and 30 dB. The support recovery error is given by $1 - J(\boldsymbol{S},\widehat{\boldsymbol{S}})$, where $J(\boldsymbol{S},\widehat{\boldsymbol{S}})$ is the Jaccard similarity defined in Equation (4.14). Jaccard similarity $J(\boldsymbol{S},\widehat{\boldsymbol{S}})$ is a measure between the support (i.e., non-zero indices) of transmitted signal matrix $\boldsymbol{S}$ containing all user transmission activity over D samples and the estimate of the sparse matrix activity $\widehat{\boldsymbol{S}}$. $\boldsymbol{S}$ and $\widehat{\boldsymbol{S}}$ contain 1's where source activity is present and 0's where no signal transmission takes place.

$$J(\boldsymbol{S},\widehat{\boldsymbol{S}}) = \frac{|\boldsymbol{S}\cap\widehat{\boldsymbol{S}}|}{|\boldsymbol{S}\cup\widehat{\boldsymbol{S}}|} \tag{4.14}$$

The sparse autoencoder expected training error and expected validation error for support recovery versus the hyperparameter λ is shown in Figure 4.4 for SNR values of 15 dB, 20 dB, 25 dB, and 30 dB.

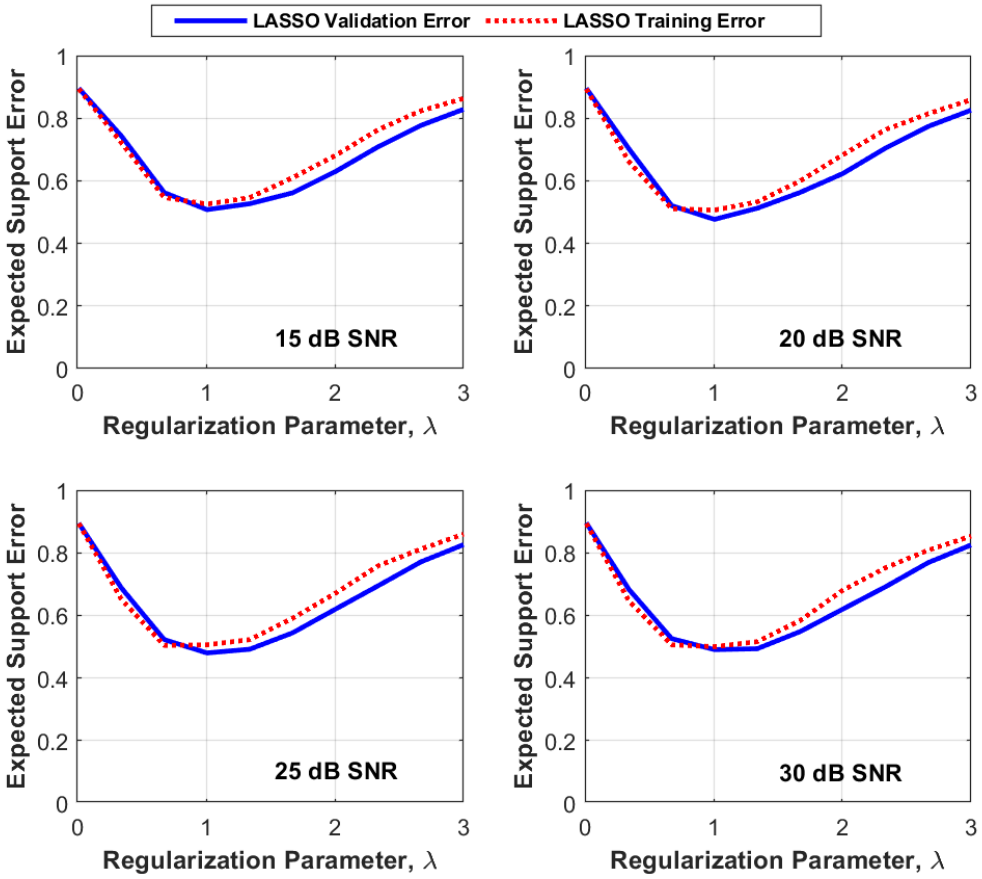

Figure 4.3 ADMM LASSO expected training error and expected validation error for support recovery versus the hyperparameter λ for SNR values of 15 dB, 20 dB, 25 dB, and 30 dB.

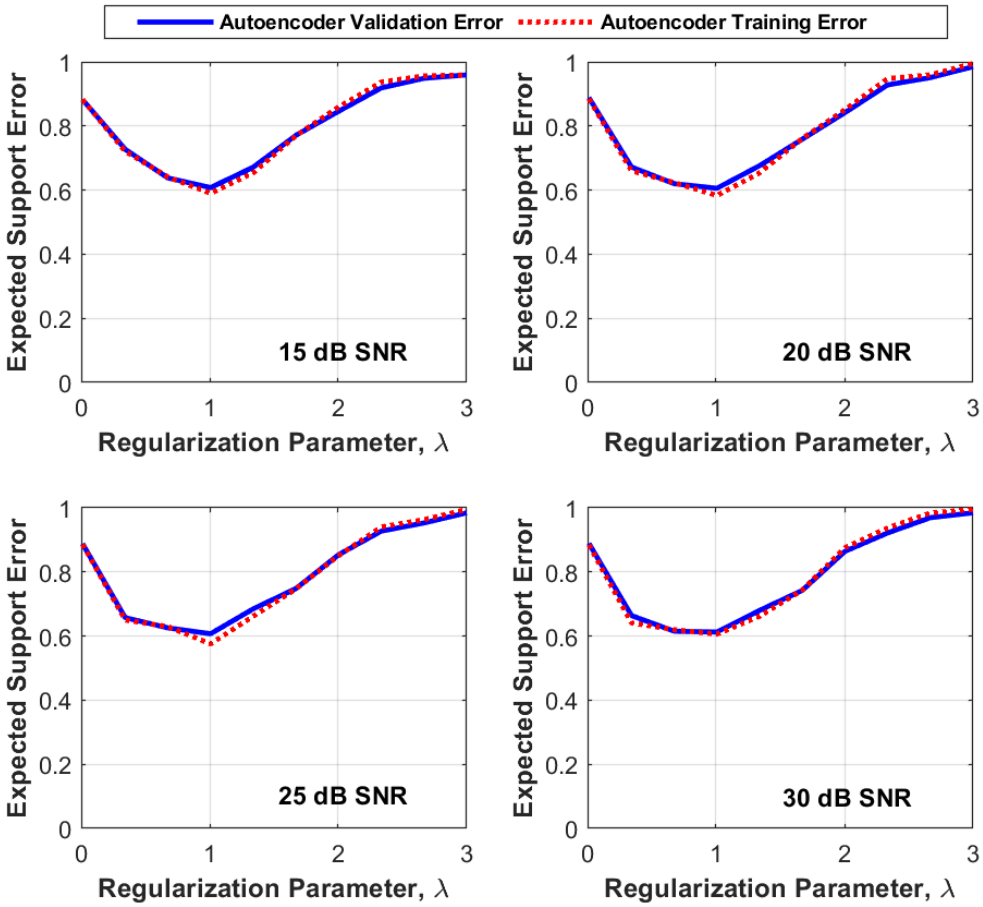

Figure 4.4 Real-valued sparse autoencoder expected training error and expected validation error for support recovery versus the hyperparameter λ for SNR values of 15 dB, 20 dB, 25 dB, and 30 dB.

The optimal hyperparameter setting for λ is equal to 1 for both the ADMM LASSO and sparse autoencoder as shown in Figures 4.3 and 4.4 via 5-fold cross-validation. The optimal value for λ coincides with the variance of the additive white Gaussian noise in Equation (4.1). The cross-validation error for the ADMM LASSO requires additional optimization iterations for solving for the sparse coding while holding the channel fixed. A plausible explanation for the improved performance of the ADMM LASSO on validation data as shown in Figure 4.3 is that the sparse coding is able to move beyond a local minima

when optimizing with new data outside the training set. On the other hand, the sparse autoencoder does not require additional optimization for sparse coding at the hidden layer and the cross-validation performance in Figure 4.4 or generalization error is slightly worse than the training error as expected.

4.3 Generalization Performance

The generalization performance of any machine learning algorithm indirectly depends on the training data and how many examples are in the training data. This is due to the fact that the generalization performance is based on a predictor learned from training data and better generalization is attained with larger sample sets [19], [35]. Therefore, the learning model and optimization of weights for learning a predictor is affected by the number of samples used for training. The sample complexity defines the minimum number of training examples needed in order to generalize well on new data within an error tolerance ϵ and confidence $1-\delta$, where δ represents the probability of the generalization error being larger than ϵ. If the training set is too small there is risk of overfitting to the data such that the training error is small, but the generalization error is large for prediction on new examples. Generalization bounds are discussed in Chapter 5 for the Blind Source Separation model.

Generalization error can also be measured empirically over a given dataset of size D for a range of subset sizes. The mean squared error (MSE) of the output layer of the sparse autoencoder and MSE of the ADMM LASSO on training and validation data (i.e., 5-fold cross-validation) is used as the measure of performance and is plotted in Figure 4.5.

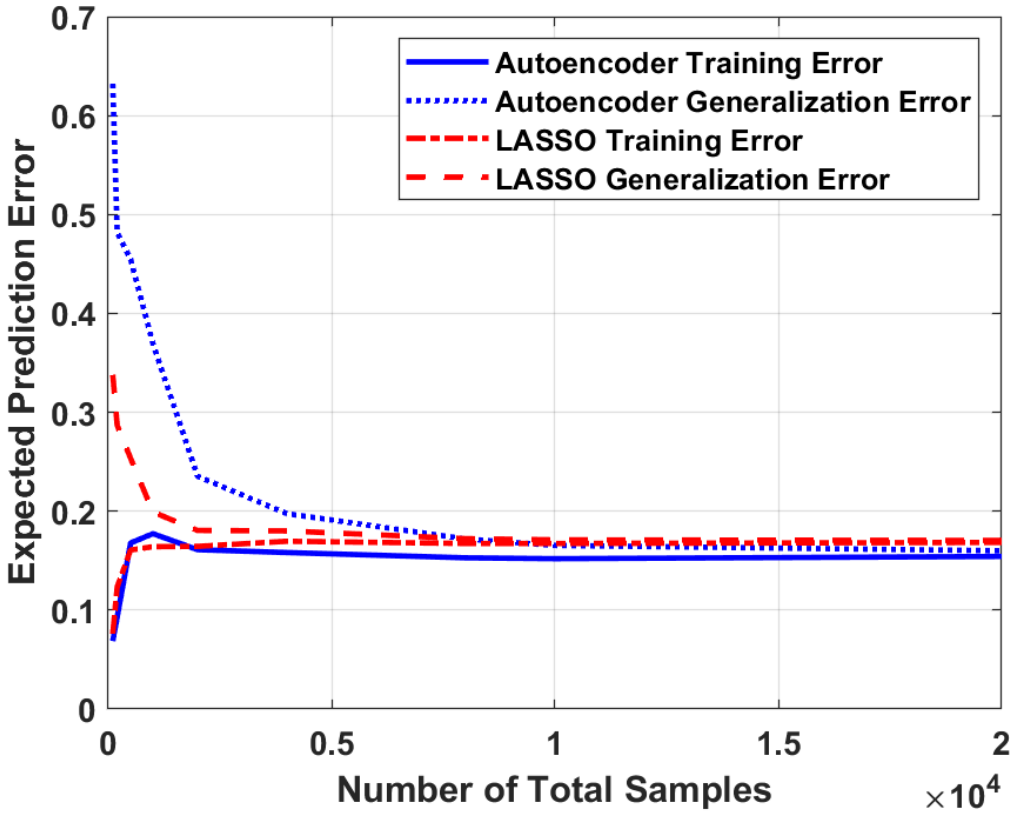

Figure 4.5 The mean squared error (MSE) of the output layer of the sparse autoencoder and MSE of the ADMM LASSO on training and validation data using 5-fold cross-validation.

As shown in Figure 4.5 the sparse autoencoder has a larger spread between the training and validation error, which is indicative of a higher model capacity [19], [35]. The ADMM LASSO generalizes better with a smaller number of examples, but the sparse autoencoder exhibits less bias for a large number of examples as shown in Figure 4.5.

4.4 Support Recovery Performance

The support recovery is measured by the Jaccard similarity index $J(\boldsymbol{S}, \widehat{\boldsymbol{S}})$ given in Equation (4.13). The Jaccard index provides a measure for how well two vectors are correlated

including detection and false alarms. The support recovery for the sparse autoencoder and ADMM LASSO is shown in Figure 4.6 over a range of SNR values from 0 dB to 30 dB. The dataset size contains 5000 samples and 5-fold cross-validation was used.

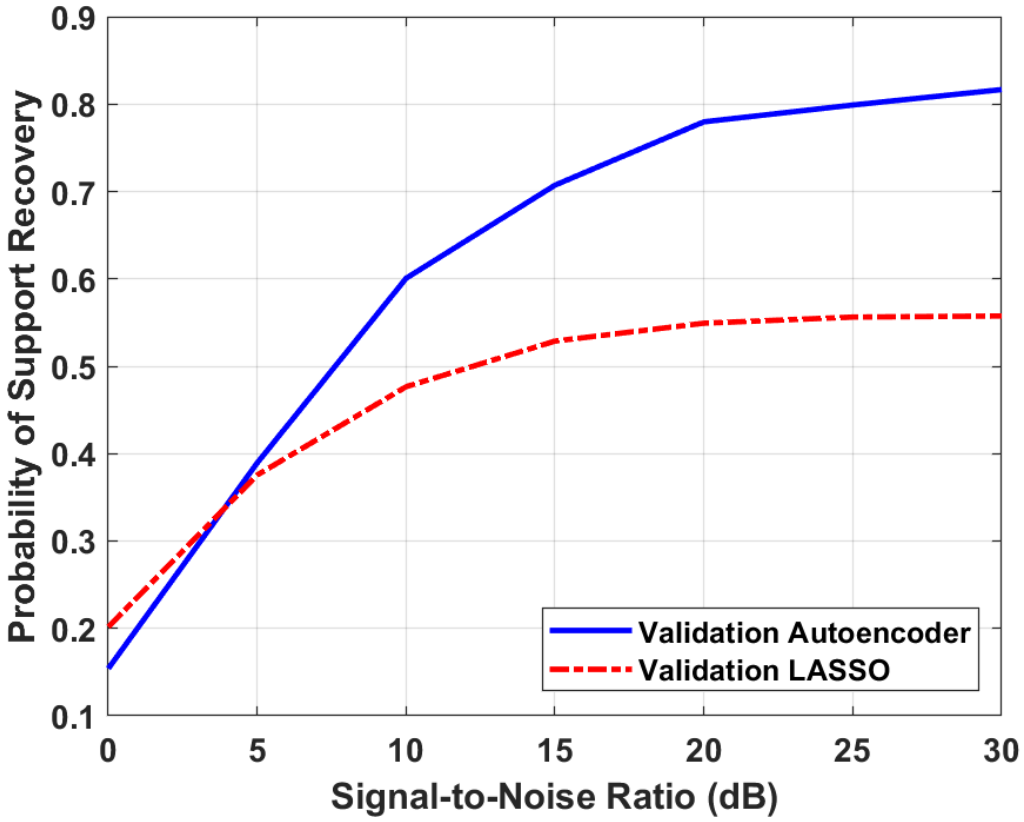

Figure 4.6 Support recovery for the real-valued sparse autoencoder and ADMM LASSO over a range of SNR values from 0 dB to 30 dB. Maximum number of signals is 20.

As shown in Figure 4.6 the sparse autoencoder begins to outperform the ADMM LASSO BSS approach at 5 dB SNR and experience a significant improvement in performance from 20 dB to 30 dB. The sparse autoencoder is a non-linear model and has a higher capacity to learn representations that explain the data as shown from the difference between the training data and generalization data in Figure 4.6.

4.4 Receiver Operating Characteristics

The performance trade-off between the probability of detection and probability of false alarm is known as the Receiver Operating Characteristic (ROC) [86]. The ROC curve for the sparse autoencoder and ADMM LASSO for blind source separation of 20 signals is shown in Figure 4.7.

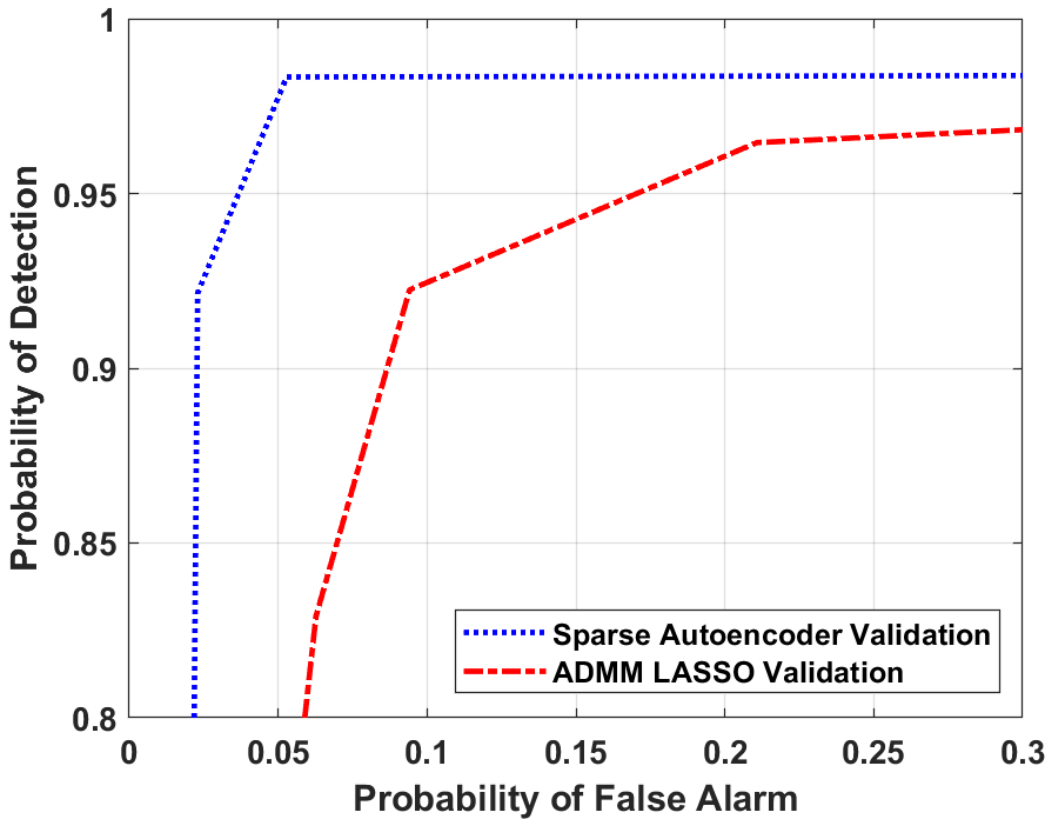

Figure 4.7 ROC curve for the real-valued sparse autoencoder and ADMM LASSO for blind source separation of 20 signals.

CHAPTER 5

SPARSE AUTOENCODER FOR COMPLEX-VALUED SIGNALS

5.1 Complex-Valued Sparse Autoencoder for Source Separation

Chapter 4 describes a real-valued sparse autoencoder for BSS of co-channel signals. In this chapter the complex-valued counterpart of the real-valued sparse autoencoder is defined. The fully complex-valued sparse autoencoder enables shrinkage of the real and imaginary parts of the complex-valued ordered pairs as a group, which is critical for sparse coding of complex-valued data as described in Section 1.4. A complex-valued sparse autoencoder design was proposed for pilot channel estimation for Sparse Code Multiple Access (SCMA) 5G systems using a ReLU activation function for the real and imaginary data separately with an ℓ_1 norm penalty on the weights in the cost function, but the approach does not produce actual zeros within the sparse code of the encoded output [63], [95]. The complex-valued sparse autoencoder proposed in this chapter maintains the phase information of the complex-valued data from the input domain to output range mapping of the activation function and produces actual zeros with a relatively small number of non-zeros at the hidden layer encoded output resulting in true sparse coding. The activation function at the hidden layer maps complex-valued input data to complex-valued output data denoted in general by function $f: \mathbb{C} \to \mathbb{C}$ with an inherent shrinkage function that produces zeros at the hidden layer output for sparse coding. The complex-valued backpropagation algorithm for updating the gradient of the complex weights during optimization or training of the sparse autoencoder is performed in the complex domain [96]. The complex-valued sparse autoencoder architecture is shown in Figure 5.1.

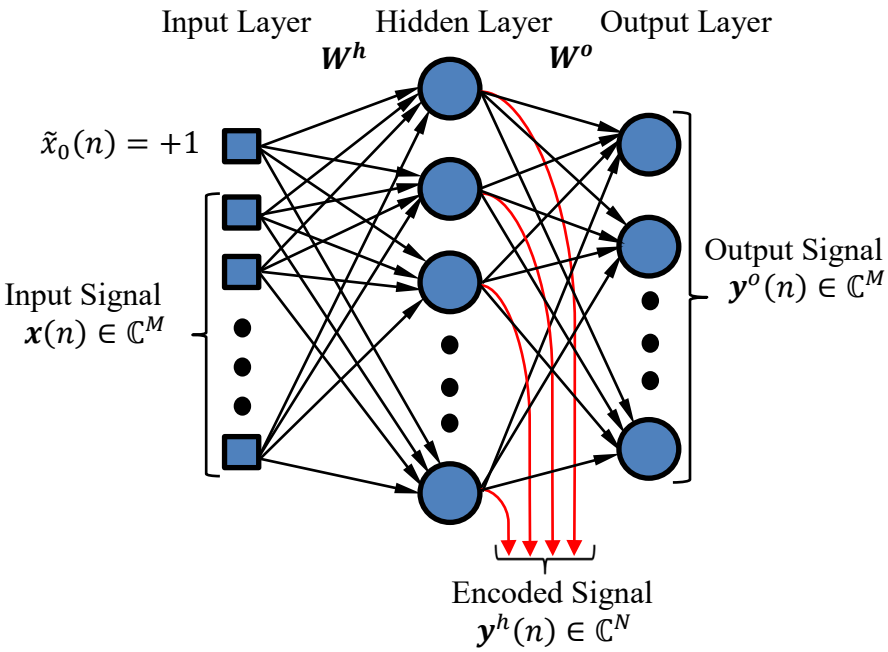

Figure 5.1 Blind sparse autoencoder feedforward neural network architecture with $N > M$ hidden layer nodes. The encoded signal output at the hidden layer provides a complex-valued sparse representation of the transmitted sources.

The received signal $\boldsymbol{x}(n) \in \mathbb{C}^M$ is an M-dimensional vector as defined in Chapter 2 and is fed to the input layer of complex-valued sparse autoencoder as shown in Figure 5.1. The cost function or loss function is optimized by minimizing Equation (5.1) with respect to the complex weights using Wirtinger calculus as described in Section 1.5. The complex-valued function signal vector at the output of the neural network is denoted as $\boldsymbol{y}^o(n) \in \mathbb{C}^M$. The complex-valued desired response $\boldsymbol{d}(n)$ is set equal to the input signal to the neural network $\boldsymbol{x}(n)$. The complex-valued hidden layer signal output vector is denoted by $\boldsymbol{y}^h(n)$, which represents a unique sparse solution via the ℓ_1 norm of $\boldsymbol{y}^h(n)$ as shown in the cost function in Equation (5.1).

$$\mathcal{L}(\boldsymbol{W}) = \frac{1}{|B|}\sum_{\mathrm{n}\in B}\|\boldsymbol{d}(n) - \boldsymbol{y}^{o}(n)\|_2^2 + \lambda\|\boldsymbol{y}^{h}(n)\|_1 \tag{5.1}$$

The gradient of $\mathcal{L}(\boldsymbol{W})$ in Equation (5.1) with respect to the weights of the output layer is given in Equation (5.2). Complex conjugation is denoted by $(\cdot)^*$. After the gradient of the cost function is updated with respect to weights $\boldsymbol{W}$ over a batch B, the weights $\boldsymbol{W}$ are updated using the ADAM algorithm over the real and imaginary parts separately [84].

$$\frac{\partial\mathcal{L}(\boldsymbol{W})}{\partial w_{kj}^{*}(n)} = \frac{1}{|B|}\sum_{n=1}^{B}\delta_k^{o}(n)y_j^{h}(n)^{*} \tag{5.2}$$

The local gradient with respect to the output layer complex conjugate activation potential $v_k^{o}(n)^{*}$ is defined in Equation (5.3). The activation potential $v_k^{o}(n)$ is defined in Equation (5.4).

$$\delta_k^{o}(n) = -[d_k(n) - y_k^{o}(n)]\varphi_o'(v_k^{o}(n)^{*}) \tag{5.3}$$

$$\boldsymbol{v}^{o}(n) = \boldsymbol{W}^{o}(n)\boldsymbol{y}^{h}(n) \tag{5.4}$$

The hidden layer output $\boldsymbol{y}^h(n)$ in Equation (5.4) is given by $\boldsymbol{y}^h(n) = \varphi_h\left(\boldsymbol{v}^h(n)\right)$ and the hidden layer activation potential $\boldsymbol{v}^h(n)$ is given by $\boldsymbol{v}^h(n) = \boldsymbol{W}^h(n)\tilde{\boldsymbol{x}}(n)$ where all variables are considered to be complex-valued.

The activation function at the hidden layer is a modReLU function, which is a complex-valued soft-threshold operator [58], [97]. The modReLU or complex-valued soft-threshold activation function is defined in Equation (5.5), which maintains the phase of the input activation potential $v_i^h(n)$.

$$\varphi_h\left(v_i^h(n)\right) = \frac{v_i^h(n)}{\left|v_i^h(n)\right|}\left(\left|v_i^h(n)\right| - \lambda\right)_+ \tag{5.5}$$

The gradient of the cost function $\mathcal{L}(\boldsymbol{W})$ with respect to the hidden layer weights is defined in Equation (5.6).

$$\frac{\partial \mathcal{L}(\boldsymbol{W})}{\partial w_{ji}^*(n)} = \frac{1}{|B|}\sum_{n=1}^{B} \delta_j^h(n)\tilde{x}_i(n)^* \tag{5.6}$$

The local gradient $\delta_j^h(n)$ with respect to the hidden layer complex conjugate activation potential $v_j^h(n)^*$ is defined in Equation (5.7).

$$\delta_j^h(n) = \varphi_h'\left(v_j^h(n)^*\right)\sum_{k=1}^{\mathrm{M}} \delta_k^o(n) w_{kj}^o(n)^* + \lambda \cdot \left(\frac{y_j^h}{|y_j^h|}\right)\varphi_h'\left(v_j^h(n)^*\right) \tag{5.7}$$

Equation (5.7) is inserted into Equation (5.6) for the gradient update for the hidden layer weight $\boldsymbol{w_{ji}(n)}$.

5.2 Hyperparameter Selection

Optimal selection of the tuning parameter or hyperparameter λ in Equations (5.1) and (5.5) is carried out using K-fold cross-validation as described in Section 3.2. The support recovery error versus λ for $\lambda \in \{.01, \ldots, 3\}$ is shown in Figure 5.2 for the complex-valued ADMM LASSO and Figure 5.3 for the complex-valued sparse autoencoder. The optimal tuning parameter is shown to be equal to the complex-valued Gaussian noise variance of 1 as defined in Equation (2.1).

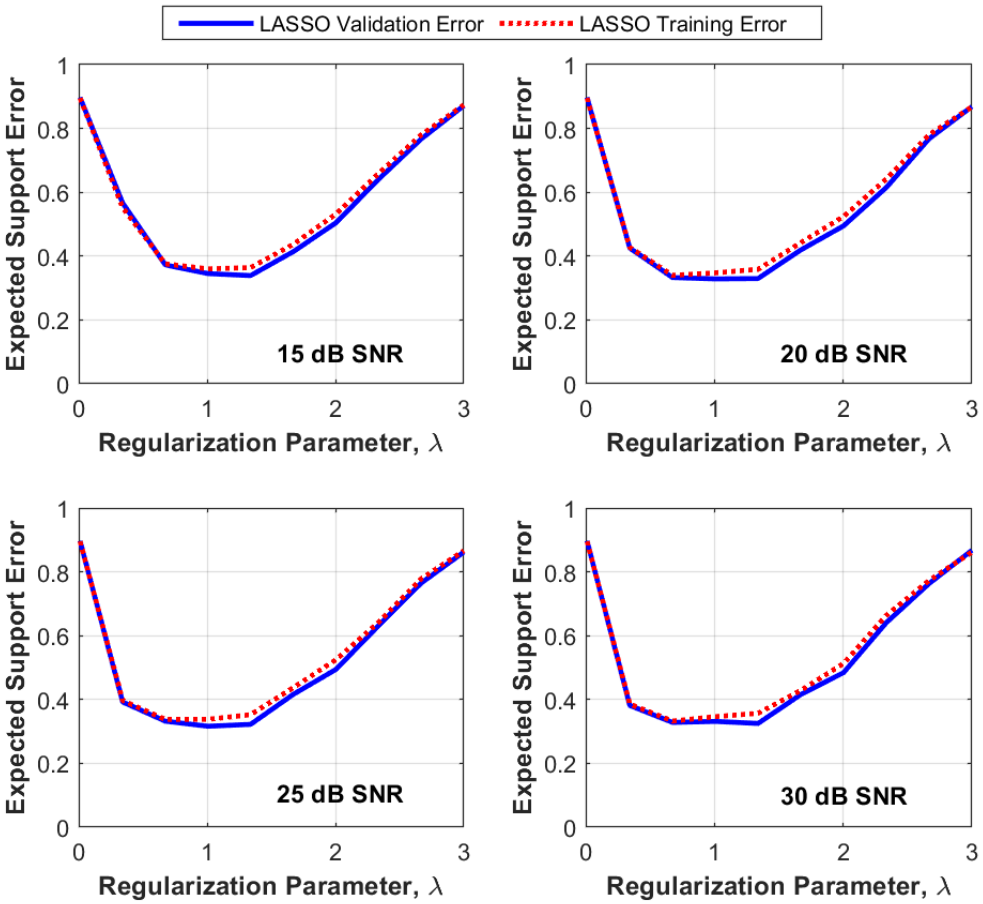

Figure 5.2 Complex-valued ADMM LASSO expected training error and expected validation error for support recovery versus the hyperparameter λ for SNR values of 15 dB, 20 dB, 25 dB, and 30 dB.

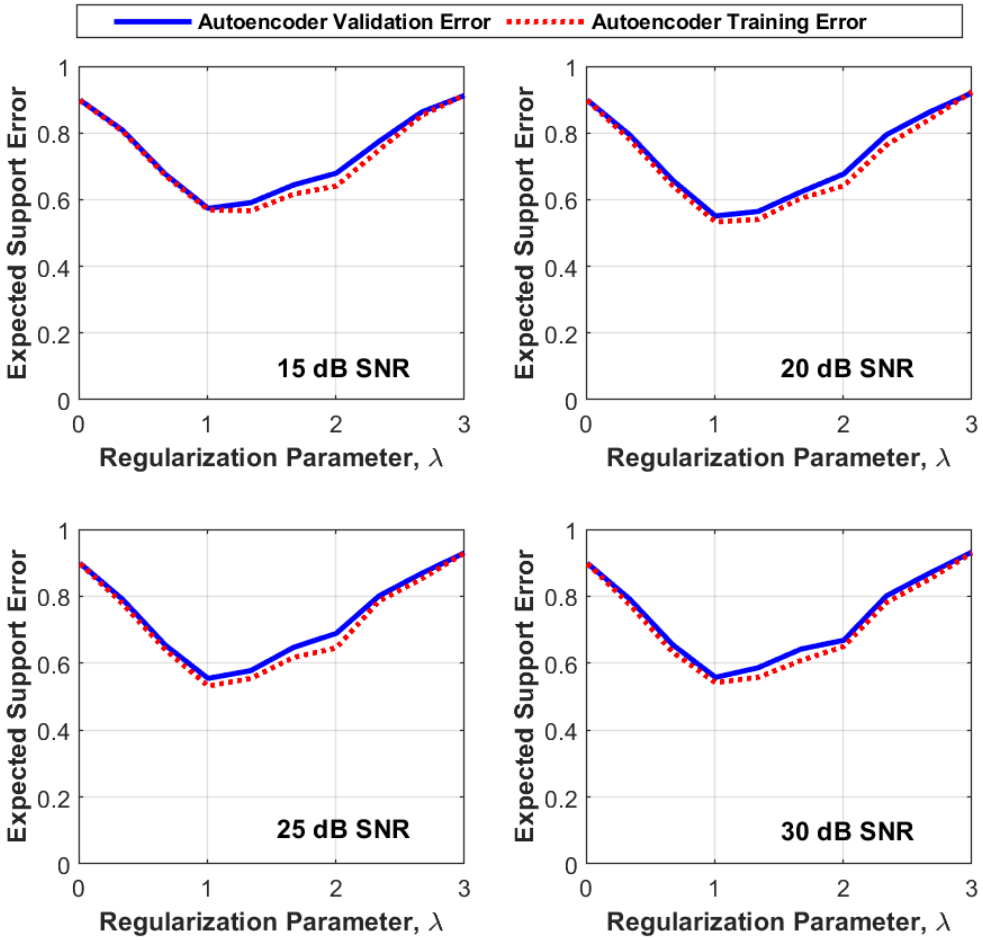

Figure 5.3 Complex-valued sparse autoencoder expected training error and expected validation error for support recovery versus the hyperparameter λ for SNR values of 15 dB, 20 dB, 25 dB, and 30 dB.

5.3 Generalization Performance

The generalization error performance for the complex-valued sparse autoencoder and complex-valued ADMM LASSO for 20 co-channel complex signal sources is carried out similar to the real-valued case as described in Section 4.3. Figure 5.4 shows the signal activity truth data for 20 co-channel source transmissions in the top plot and the recovered signals are shown in the bottom plot via the sparse autoencoder.

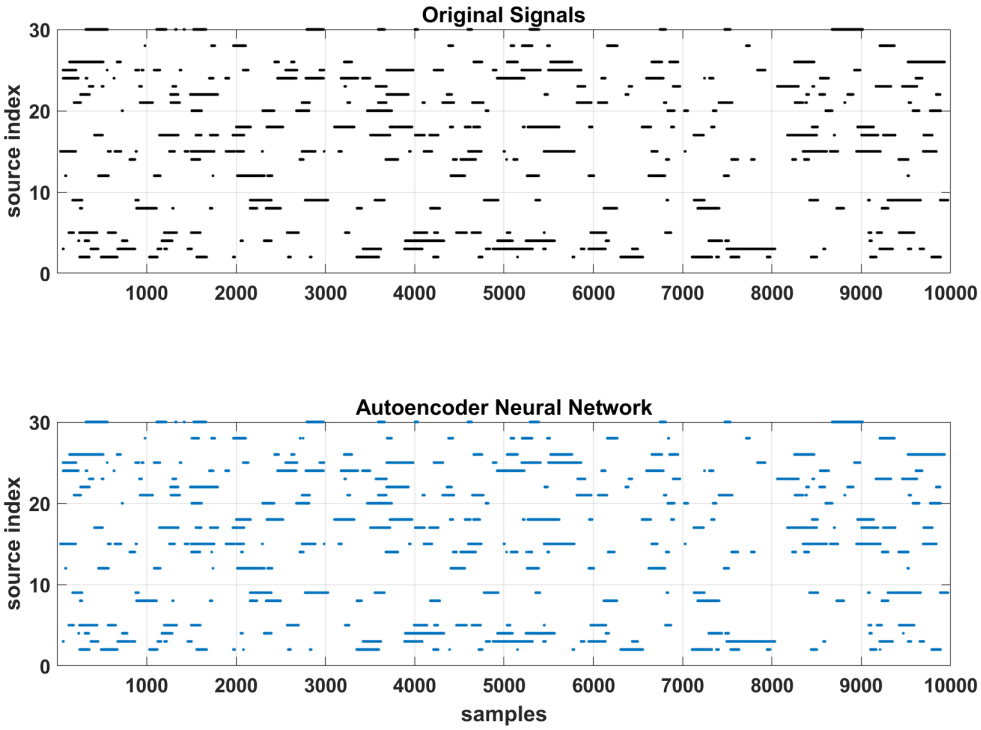

Figure 5.4 Signal activity truth data for 20 co-channel source transmissions in the top plot and the recovered signals via the sparse autoencoder are shown in the bottom plot.

The generalization performance of the prediction error versus number of samples in the dataset is shown in Figure 5.5 for the complex-valued sparse autoencoder and complex-valued ADMM LASSO for blind source separation.

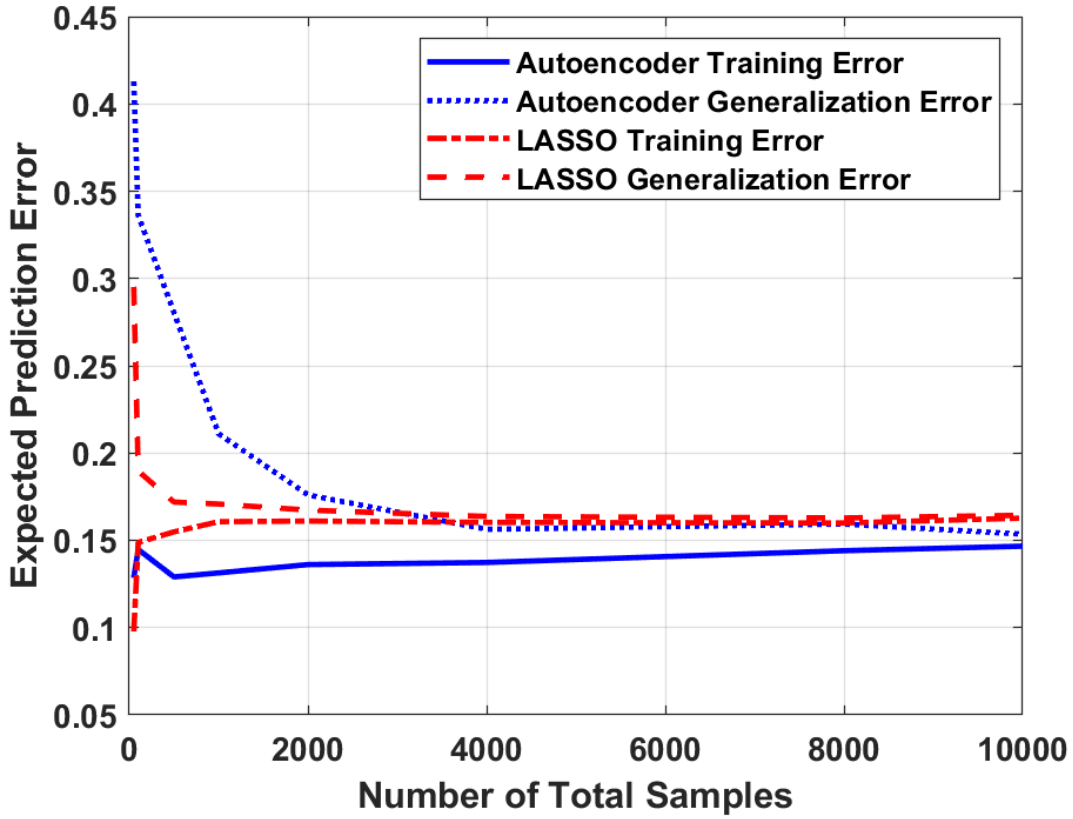

Figure 5.5 The mean squared error (MSE) of the output layer of the complex-valued sparse autoencoder and MSE of the complex-valued ADMM LASSO on training and validation data using 5-fold cross-validation.

The non-linear complex-valued sparse autoencoder is empirically shown to have a higher capacity than the linear LASSO algorithm by observing the difference between the training and validation error. A larger difference in error between the training and generalization error, which is approximated by cross-validation, indicates that the complex-valued sparse autoencoder is a more complex hypothesis class. For large samples the complex-valued autoencoder demonstrates less bias than the ADMM LASSO approach as shown in Figure 5.5.

5.4 Support Recovery Performance

The support recovery performance was carried out using the Jaccard similarity index as described in Section 4.4. The support recovery performance for the complex-valued sparse autoencoder and complex-valued ADMM LASSO is shown in Figure 5.6 using validation data. A total of 10000 samples were used with 5-fold cross-validation.

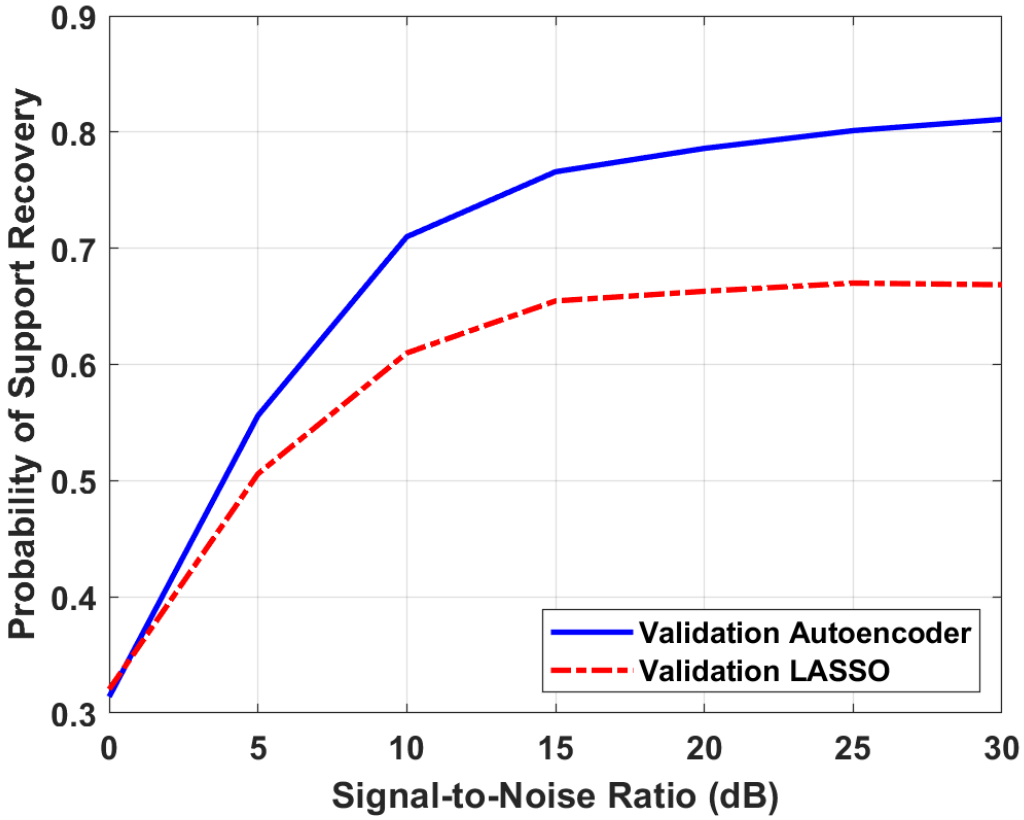

Figure 5.6 Support recovery for the complex-valued sparse autoencoder and complex-valued ADMM LASSO over a range of SNR values from 0 dB to 30 dB. Maximum number of signals is 20.

It is clear from Figure 5.6 that the complex-valued sparse autoencoder has a markedly improved probability of support recovery over the two-stage complex-valued ADMM LASSO for SNR values greater than roughly 3 dB SNR.

5.5 Receiver Operating Characteristics

The ROC curve performance of the probability of detection versus probability of false alarm is shown in Figure 5.7 and is shown to outperform the complex-valued ADMM LASSO approach.

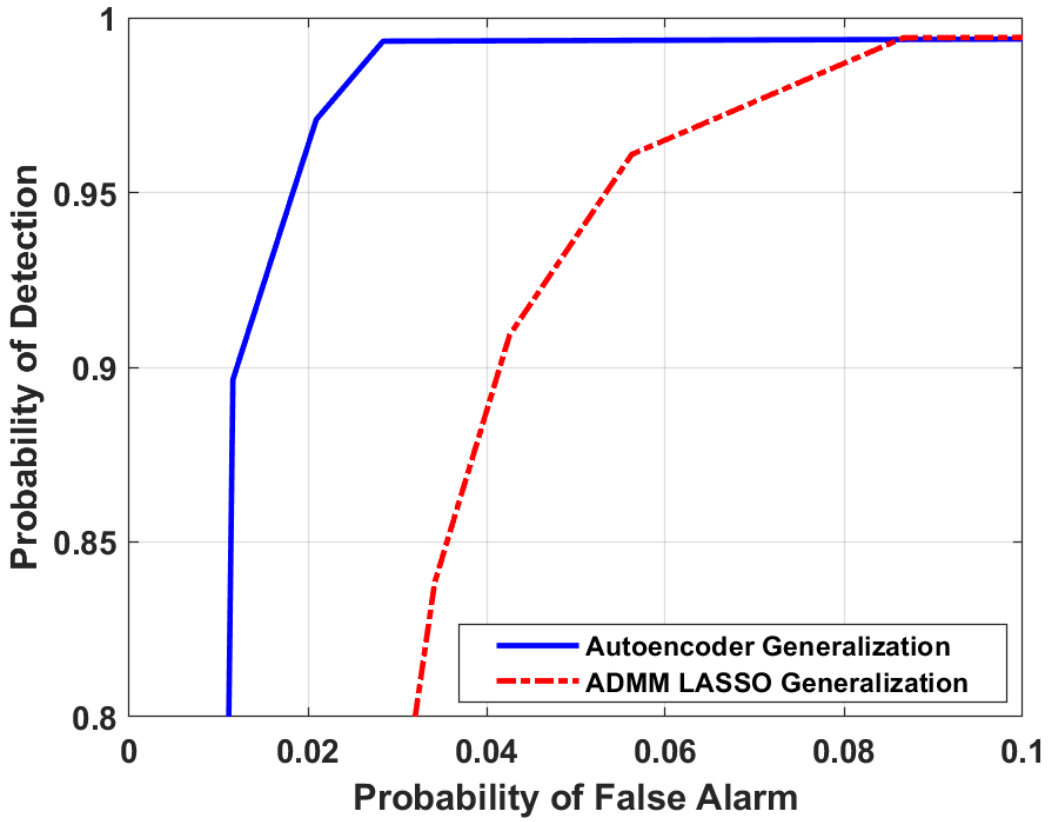

Figure 5.7 ROC curve for the complex-valued sparse autoencoder and complex-valued ADMM LASSO for blind source separation of 20 signals.

CHAPTER 6

GENERALIZATION BOUNDS AND SAMPLE COMPLEXITY

6.1 PAC Learning for Regression

A critical aspect of machine learning is the ability to generalize to unseen new data examples, meaning that the final trained predictor can perform well on unlabeled data inputs presented to the machine learning algorithm. How well the machine learning algorithm performs can be quantified in terms of two key metrics. First the generalization error should be close to the training error and second, the training error should be small. If the training error is small, but the generalization error on new data is very large, then the machine learning algorithm is a poor predictor on new examples. If the machine learning algorithm generalizes well, but has a significantly large error then the learning algorithm model or hypothesis class does not have the capacity to fit the data. Generalization bounds deal with the question of how well a learning algorithm can generalize to new data and is a function of its *capacity* and number of samples required to meet a desired level of performance known as the *sample complexity* [19]. The model *capacity* represents the complexity of the learning model and its degrees of freedom in fitting the data. Probably Approximately Correct (PAC) learning provides a probabilistic guarantee for a hypothesis class such that the generalization error is within a tolerance ϵ of the training error with probability $1 - \delta$ if the sample complexity is satisfied.

The generalization loss, generalization error, or generalization risk $R_{\mathcal{P}}(h)$ defined in Equation (6.1) is the expected error between the predictor $h(x)$ and underlying target function plus noise values t, where $(x, t) \sim \mathcal{P}$. $\mathcal{P}$ is an unknown probability distribution

over the input domain space $\mathcal{X}$ and target space $\mathcal{T}$ denoted by the cartesian product $\mathcal{X} \times \mathcal{T}$, where $x \in \mathcal{X}$ and $t \in \mathcal{T}$.

$$R_{\mathcal{P}}(h) = \mathbb{E}_{(x,t)\sim\mathcal{P}}[L(h(x), t)] \tag{6.1}$$

The loss function $L(h(x), t)$ in Equation (6.1) is the squared loss as is common for regression problems and is given in Equation (6.2) [20].

$$L(h(x), t) = (h(x) - t)^2 \tag{6.2}$$

The optimal predictor that minimizes Equation (6.1) cannot be determined directly given $\mathcal{P}$ is unknown [35]. On the other hand, a set of D independent and identically distributed (i.i.d.) training examples $S = \big((x_1, t_1), \ldots, (x_{\mathrm{D}}, t_{\mathrm{D}})\big) \in (\mathcal{X} \times \mathcal{T})^{\mathrm{D}}$ drawn according to $\mathcal{P}$ are used to find an optimal predictor that minimizes the mean squared error on the training examples and is formerly known as *Empirical Risk Minimization* (ERM) [20]. The empirical loss $\hat{R}_S(h)$ for a predictor $h \in \mathcal{H}$ on a set of samples S is given in Equation (6.3) and the ERM predictor h_S^{ERM} that minimizes $\hat{R}_S(h)$ is defined in Equation (6.4).

$$\hat{R}_S(h) = \frac{1}{\mathrm{D}} \sum_{i=1}^{\mathrm{D}} L(h(x_i), t_i) \tag{6.3}$$

$$h_S^{ERM} \in \arg\min_{h \in \mathcal{H}} \hat{R}_S(h) \tag{6.4}$$

Hoeffding's inequality can be used to provide a generalization bound for regression problems with a finite hypothesis class. Hoeffding's inequality is given in Equation (6.5).

$$\mathbb{P}\left[R_{\mathcal{P}}(h) - \hat{R}_S(h) > \epsilon\right] \leq e^{-\frac{2\mathrm{D}\epsilon^2}{\Gamma^2}} \tag{6.5}$$

Equation (6.5) states that for any $h \in \mathcal{H}$ the probability that the generalization error deviates from the empirical error by more than ϵ is less than or equal to an exponentially decreasing quantity that is a function of the number of samples D, the tolerance ϵ, and the maximum of the bounded loss function in Equation (6.2) denoted by $L(h(x), t) \leq \Gamma$. The maximum of the loss function in Equation (6.2) for one receive signal as defined in Equation (2.1) from the antenna array considering a prediction output $h(x)$ of zero gives $\Gamma = P_s + \sigma^2$, which is a function of the signal power P_s and noise variance σ^2. Γ can also be rewritten as a function of the SNR given by $\Gamma = \sigma^2(\mathrm{SNR} + 1)$.

The union bound is used to derive a generalization bound for all $h \in \mathcal{H}$, which implies that the probability of the union of all events is less than or equal to the sum of the

individual event probabilities. The union bound is applied to the right-hand side of Equation (6.6a) in combination with Equation (6.5) for each hypothesis yielding the final generalization bound in Equation (6.6b). Equation (6.6b) provides a generalization bound $\forall h \in \mathcal{H}$ based on the capacity or complexity of $\mathcal{H}$.

$$\mathbb{P}\left[\exists h \in \mathcal{H}: \left|R_{\mathcal{P}}(h) - \hat{R}_S(h)\right| > \epsilon\right] = \mathbb{P}\left[\bigcup_{h\in\mathcal{H}}\left(\left|R_{\mathcal{P}}(h) - \hat{R}_S(h)\right| > \epsilon\right)\right] \tag{6.6a}$$

$$\leq \sum_{h\in\mathcal{H}} \mathbb{P}\left[\left|R_{\mathcal{P}}(h) - \hat{R}_S(h)\right| > \epsilon\right] = 2|\mathcal{H}|e^{-\frac{2\mathrm{D}\epsilon^2}{(P_S+\sigma^2)^2}} \tag{6.6b}$$

For binary classification the Vapnik-Chervonenkis (VC) dimension can be used for defining the capacity of $\mathcal{H}$, but that does not translate to regression prediction problems [20], [90]. However, the pseudo-dimension can be applied to regression problems that transforms continuous variables into binary states resulting in a pseudo-VC dimension [87]-[89], [90]. The pseudo-dimension for a continuous-valued hypothesis class is the largest set pseudo-shattered by $\mathcal{F}_{\mathrm{D}}$ denoted by $\mathrm{Pdim}(\mathcal{H})$ [87], [88], [90].

The set $\{x_1, \ldots, x_{\mathrm{D}}\} \subseteq \mathcal{X}$ is considered pseudo-shattered by $\mathcal{F}_{\mathrm{D}}$ if $\exists\, \{f_{\mathrm{d}}: f_{\mathrm{d}} \in \mathcal{F}_{\mathrm{D}}\}$ that satisfies all dichotomies for D-points given by Equation (6.7) where $t_1^w, \ldots, t_{\mathrm{D}}^w \in \mathbb{R}$ witness the shattering [87], [88], [90].

$$\left|\left\{\begin{bmatrix}\operatorname{sgn}(f_d(x_1)-t_1^w)\\ \vdots \\ \operatorname{sgn}(f_d(x_{\mathrm{D}})-t_{\mathrm{D}}^w)\end{bmatrix}: f_d \in \mathcal{F}_{\mathrm{D}}\right\}\right| = 2^{\mathrm{D}} \tag{6.7}$$

Equation (6.6b) provides a combinatorial bound that requires a model capacity approximation for continuous-valued functions via the pseudo-dimension, and is valid for any data distribution $\mathcal{P}$. An alternative approach is to provide a data-dependent bound using the Rademacher complexity that does not require the model capacity to be explicitly defined [91], which is described in Section 6.2.

6.2 Rademacher Complexity Generalization Bound

The Rademacher complexity can be used to measure the capacity or complexity of a hypothesis class and provide a data-dependent generalization bound. The empirical Rademacher complexity of the family of loss functions $\mathcal{F}_{\mathcal{L}}$ associated with the hypothesis class $\mathcal{H}$ with respect to the sample set $S = \big((x_1, t_1), \ldots, (x_{\mathrm{D}}, t_{\mathrm{D}})\big) \in (\mathcal{X} \times \mathcal{T})^{\mathrm{D}}$ is given in Equation (6.8). The family of loss functions denoted by $\mathcal{F}_{\mathcal{L}}$ is defined as $\mathcal{F}_{\mathcal{L}} = \{(x, t) \mapsto \mathcal{L}(h(x), t): h \in \mathcal{H}\}$, which is a function of the data distribution $\mathcal{P}$ and hypothesis class $\mathcal{H}$.

$$\widehat{\mathfrak{R}}_S(\mathcal{F}_{\mathcal{L}}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}}} \frac{1}{\mathrm{D}} \sum_{i=1}^{\mathrm{D}} \sigma_i f_{\mathcal{L}}(x_i, t_i)\right] \tag{6.8}$$

The set of parameters $\{\sigma_i\}_{i=1}^{\mathrm{D}}$ are called Rademacher random variables and are i.i.d. random variables with $\mathbb{P}(\sigma_i = +1) = \mathbb{P}(\sigma_i = -1) = {}^{1}/_{2}$. The Rademacher complexity in Equation (6.8) captures the richness of a family of functions whereby a function class that correlates better with random noise has a higher model complexity. Another way to interpret the Rademacher complexity is to consider the gap between the generalization error and training error, whereby a larger gap implies a higher model complexity in comparison to a lower model capacity for a given sample set $S \sim \mathcal{P}^{\mathrm{D}}$. To better understand this notion the Rademacher complexity will be explained from an ϵ-representative sample perspective, which states that a training set $S \sim \mathcal{P}^{\mathrm{D}}$ is ϵ-representative if it satisfies Equation (6.9) $\forall h \in \mathcal{H}$ [20].

$$\sup_{h \in \mathcal{H}} \left| R_{\mathcal{P}}(h) - \hat{R}_S(h) \right| \leq \epsilon, \qquad \forall h \in \mathcal{H} \tag{6.9}$$

The representativeness of a training set S with respect to $\mathcal{F}_{\mathcal{L}}$ is defined as the supremum of the difference between the generalization error of a function $f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}}$ and its empirical error or training error and is given in Equation (6.10) [20]. The generalization error $R_{\mathcal{P}}(f_{\mathcal{L}})$ and empirical risk $\hat{R}_S(f_{\mathcal{L}})$ in Equation (6.10) for a function $f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}}$ are defined similarly with respect to Equations (6.1) and (6.3), but rather as a function of $f_{\mathcal{L}}$ that maps to the loss function.

$$\mathrm{Rep}_{\mathcal{P}}(\mathcal{F}_{\mathcal{L}}, S) = \sup_{f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}}} \left(R_{\mathcal{P}}(f_{\mathcal{L}}) - \hat{R}_S(f_{\mathcal{L}}) \right) \tag{6.10}$$

The validation error can be used to approximate the generalization error of an ERM predictor or hypothesis $h_S^{ERM} \in \arg\min_{h \in \mathcal{H}} \hat{R}_S(h)$ using a subset or holdout set from S [19]. Assume that $S \sim \mathcal{P}^{\mathrm{D}}$ is split into a validation set $S_V \sim \mathcal{P}^{\mathrm{D}/2}$ and training set $S_T \sim \mathcal{P}^{\mathrm{D}/2}$, where $S = S_V \cup S_T$. The validation error and training error based on sample set $S \sim \mathcal{P}^{\mathrm{D}}$ provides an approximation of the representativeness of S denoted as $\widehat{\mathrm{Rep}_{\mathcal{P}}}(\mathcal{F}_{\mathcal{L}}, S)$ as defined in Equation (6.11) [20].

$$\widehat{\mathrm{Rep}_{\mathcal{P}}}(\mathcal{F}_{\mathcal{L}}, S) = \sup_{f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}}} \left(\hat{R}_{S_V}(f_{\mathcal{L}}) - \hat{R}_{S_T}(f_{\mathcal{L}}) \right) \tag{6.11}$$

Equation (6.11) can be re-written more compactly as given in Equation (6.12) assuming that $S_V = \{(x_i, t_i) : \sigma_i = +1\}$ and $S_T = \{(x_i, t_i) : \sigma_i = +1\}$.

$$\sup_{f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}}} \frac{2}{\mathrm{D}} \sum_{i=1}^{\mathrm{D}} \sigma_i f_{\mathcal{L}}(x_i, t_i) \tag{6.12}$$

Equation (6.12) is shown to be similar to the empirical Rademacher complexity as defined in Equation (6.5), where Equation (6.5) takes the expectation over the Rademacher random variables $\boldsymbol{\sigma}$.

The data-dependent generalization bound for all $f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}}$ based on the empirical Rademacher complexity for D samples S with confidence $1-\delta$ is defined in Equation (6.13) where the maximum squared error loss associated with each received signal is given by $L(h(x), t) \leq \Gamma = P_s + \sigma^2 = \sigma^2(\text{SNR} + 1)$.

$$R_{\mathcal{P}}(f_{\mathcal{L}}) \leq \hat{R}_S(f_{\mathcal{L}}) + 2\hat{\mathfrak{R}}_S(\mathcal{F}_{\mathcal{L}}) + 3\Gamma\sqrt{\frac{1}{2\text{D}}\ln\frac{2}{\delta}}$$

$$R_{\mathcal{P}}(f_{\mathcal{L}}) \leq \hat{R}_S(f_{\mathcal{L}}) + 2\hat{\mathfrak{R}}_S(\mathcal{F}_{\mathcal{L}}) + 3(P_s + \sigma^2)\sqrt{\frac{1}{2\text{D}}\ln\frac{2}{\delta}} \tag{6.13}$$

The loss function over M receive antenna elements averaged over D examples denoted by the sample set $S = \big((x_1, t_1), \ldots, (x_{\text{D}}, t_{\text{D}})\big) \in (\mathcal{X} \times \mathcal{T})^{\text{D}}$ is defined in Equation (6.14).

$$\frac{1}{\text{D}}\sum_{i=1}^{\text{D}}\sum_{j=1}^{M} L(h_j(\boldsymbol{x}_i), t_i^j) \tag{6.14}$$

The received signals as defined in Chapter 2 are i.i.d. and the maximum of the loss function in Equation (6.14) for each receive signal is given by $\max\big(L(h_j(\boldsymbol{x}_i), t_i^j)\big) = P_s + \sigma^2$. The Rademacher complexity data-dependent generalization bound for M receive antennas is defined in Equation (6.15). If the loss function in Equation (6.14) is also averaged over M antenna elements by including the scalar multiple $1/M$ then the Rademacher complexity data-dependent generalization bound defaults to Equation (6.13).

$$R_{\mathcal{P}}(f_{\mathcal{L}}) \leq \hat{R}_S(f_{\mathcal{L}}) + 2\widehat{\mathfrak{R}}_S(\mathcal{F}_{\mathcal{L}}) + 3M(P_s + \sigma^2)\sqrt{\frac{1}{2\mathrm{D}}\ln\frac{2}{\delta}} \tag{6.15}$$

The Rademacher complexity generalization bound is derived based on McDiarmid's inequality, which is used to bound the representativeness $\mathrm{Rep}_{\mathcal{P}}(\mathcal{F}_{\mathcal{L}}, S)$ as defined in Equation (6.10) of an independent sample set S [20], [87]. For convenience of notation $\Phi(S)$ is defined as $\Phi(S) = \mathrm{Rep}_{\mathcal{P}}(\mathcal{F}_{\mathcal{L}}, S)$. McDiarmid's inequality applied to $\Phi(S)$ is defined in Equation (6.16a) for a real-valued function $\Phi(S)\colon \mathcal{Z}^{\mathrm{D}} \to \mathbb{R}$ with $S = \{z_i = (x_i, t_i)\colon z_i \sim \mathcal{P}, i = 1, \ldots, \mathrm{D}\}$. The parameter $c_i = \frac{1}{\mathrm{D}}(P_s + \sigma^2)$ in Equation (6.16a) is derived in Equation (6.16b) and is an upper bound of $|\Phi(S) - \Phi(S')| \leq c_i$, which makes use of the property that the supremum of the difference is greater than or equal to the difference of supremum [87]. S and S' denote two sample sets that only differ between samples z_i and z_i'.

$$\mathbb{P}[|\Phi(S) - \mathbb{E}_S[\Phi(S)]| \geq \epsilon] \leq 2e^{-\frac{2\epsilon^2}{\sum_{i=1}^{\mathrm{D}} c_i^2}} = 2e^{-\frac{2\mathrm{D}\epsilon^2}{(P_s+\sigma^2)^2}} \tag{6.16a}$$

$$\begin{aligned} |\Phi(S) - \Phi(S')| &\leq \left|\sup_{f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}}} \left(\hat{R}_S(f_{\mathcal{L}}) - \hat{R}_{S'}(f_{\mathcal{L}})\right)\right| = \left|\sup_{f_{\mathcal{L}} \in \mathcal{F}_{\mathcal{L}}} \frac{f_{\mathcal{L}}(z_i) - f_{\mathcal{L}}(z_i')}{\mathrm{D}}\right| \leq c_i \\ &= \frac{1}{\mathrm{D}}(P_s + \sigma^2) \end{aligned} \tag{6.16b}$$

McDiarmid's inequality can be used to derive a $1-\delta$ confidence bound for $\Phi(S)$ by setting Equation (6.16a) equal to δ and solving for ϵ and then rewriting Equation (6.16a) as $\mathbb{P}[|\Phi(S)-\mathbb{E}_S[\Phi(S)]| \leq \epsilon]$ resulting in Equation (6.17).

$$\Phi(S) \leq \mathbb{E}_S[\Phi(S)] + (P_s + \sigma^2)\sqrt{\frac{1}{2\mathrm{D}}\ln\left(\frac{2}{\delta}\right)} \tag{6.17}$$

The expectation term in Equation (6.17) $\mathbb{E}_S[\Phi(S)]$ is upper bounded by the expected Rademacher complexity as given in Equation (6.18) [20], [87].

$$\mathbb{E}_S[\Phi(S)] \leq 2\,\underset{S\sim\mathcal{P}^{\mathrm{D}}}{\mathbb{E}}\left[\widehat{\mathfrak{R}}_S(\mathcal{F}_{\mathcal{L}})\right] = 2\mathfrak{R}_{\mathrm{D}}(\mathcal{F}_{\mathcal{L}}) \tag{6.18}$$

The expectation term $\mathbb{E}_S[\Phi(S)]$ on the right-hand side of Equation (6.17) can be replaced by the upper bound of Equation (6.18) resulting in a Rademacher complexity bound given in Equation (6.19) that is a function of the expected Rademacher complexity over all samples of size D.

$$R_{\mathcal{P}}(f_{\mathcal{L}}) \leq \hat{R}_S(f_{\mathcal{L}}) + 2\mathfrak{R}_{\mathrm{D}}(\mathcal{F}_{\mathcal{L}}) + (P_s + \sigma^2)\sqrt{\tfrac{1}{2\mathrm{D}}\ln\tfrac{2}{\delta}} \tag{6.19}$$

Using McDiarmid's inequality again to bound the expected Rademacher complexity $\mathfrak{R}_{\mathrm{D}}(\mathcal{F}_{\mathcal{L}})$ results in the inequality in Equation (6.20) that is upper bounded by the empirical Rademacher complexity and $\epsilon = (P_s + \sigma^2)\sqrt{\frac{1}{2\mathrm{D}}\ln\left(\frac{2}{\delta}\right)}$.

$$\mathfrak{R}_{\mathrm{D}}(\mathcal{F}_{\mathcal{L}}) \leq \widehat{\mathfrak{R}}_{S}(\mathcal{F}_{\mathcal{L}}) + (P_s + \sigma^2)\sqrt{\frac{1}{2\mathrm{D}}\ln\left(\frac{2}{\delta}\right)} \tag{6.20}$$

Substituting the upper bound in Equation (6.20) for $\mathfrak{R}_{\mathrm{D}}(\mathcal{F}_{\mathcal{L}})$ in Equation (6.19) gives the Rademacher complexity data-dependent generalization bound defined in Equation (6.13) that is a function of the empirical Rademacher complexity. The derivation of the Rademacher complexity generalization bound defined in Equation (6.15) for M receive antennas follows similarly by replacing c_i in Equations (6.16a) and (6.16b) with $c_i = \frac{M}{\mathrm{D}}(P_s + \sigma^2)$.

The sample complexity defines the number of required samples D needed to achieve a certain generalization error tolerance τ with respect to the training error and empirical Rademacher complexity for a confidence parameter δ that defines the probability of violating the error tolerance τ. Setting the last term of the right-hand side of inequality Equation (6.15) equal to τ as given in Equation (6.21) and solving for D provides a sample complexity as a function of the signal-to-noise ratio (SNR), noise power σ^2, confidence parameter δ, error tolerance τ, and number of received signals M, which is defined in Equation (6.22).

$$R_{\mathcal{P}}(f_{\mathcal{L}}) \leq \hat{R}_S(f_{\mathcal{L}}) + 2\hat{\mathfrak{R}}_S(\mathcal{F}_{\mathcal{L}}) + \tau$$

$$\tau = 3M(P_s + \sigma^2)\sqrt{\frac{1}{2\mathrm{D}}\ln\frac{2}{\delta}} \tag{6.21}$$

$$\mathrm{D} \geq \frac{1}{2}\left[\frac{3M(P_s + \sigma^2)}{\tau}\right]^2 \ln\left(\frac{2}{\delta}\right) = \frac{1}{2}\left[\frac{3M\sigma^2(\mathrm{SNR} + 1)}{\tau}\right]^2 \ln\left(\frac{2}{\delta}\right) \tag{6.22}$$

The empirical Rademacher complexity generalization error data-dependent bound for the BSS complex-valued sparse autoencoder hypothesis class with error tolerance $\tau = .1$ and confidence parameter $\delta = .1$ in Equation (6.21) is shown in Figure 6.1. The simulation signal model is comprised of twenty co-channel QPSK signal sources as described in Chapter 2, which includes a time-invariant Rayleigh fading MIMO channel and SNR of 20 dB with twenty receive antenna elements. The loss function in Equation (6.14) was additionally scaled by the number of antennas and thus, $M = 1$ in Equation (6.21). The 5-Fold cross-validation generalization error and training error from Figure 5.5 for the BSS complex-valued sparse autoencoder are also included in Figure 6.1 for comparison with the empirical Rademacher complexity generalization bound. The empirical Rademacher complexity generalization bound is representative for all $h \in \mathcal{H}$.

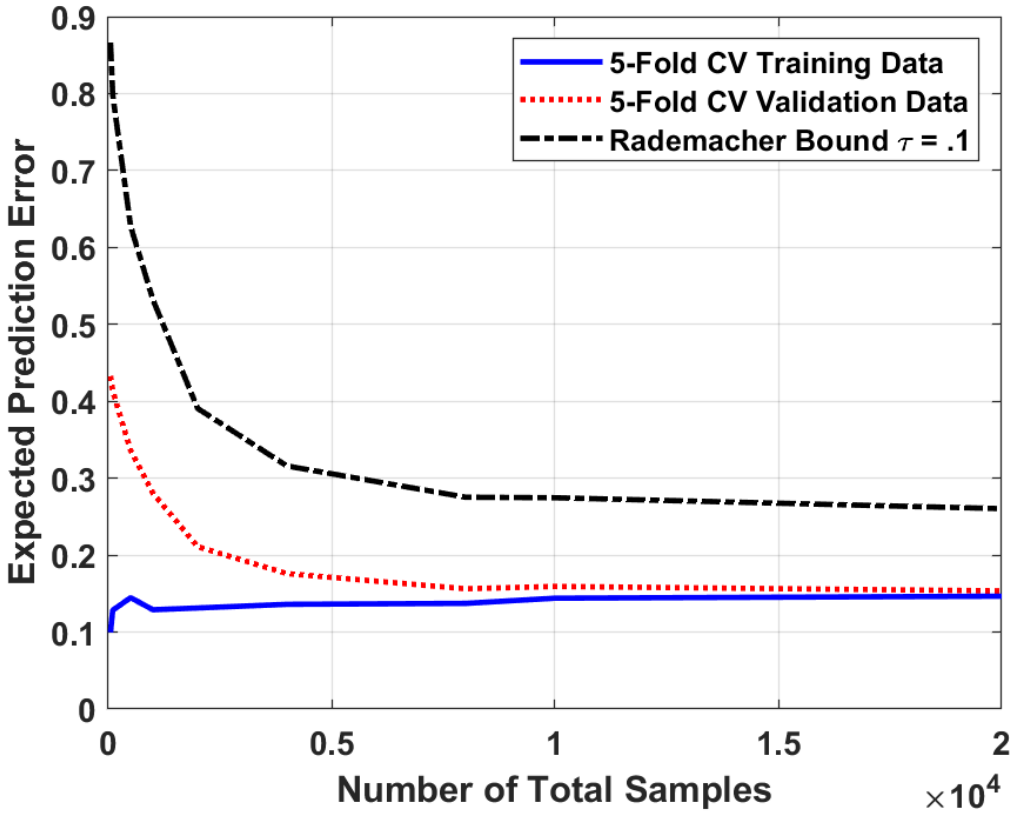

Figure 6.1 The empirical Rademacher complexity generalization error bound with error tolerance $\tau = .1$ and confidence parameter $\delta = .1$ for the BSS complex-valued sparse autoencoder hypothesis class. The simulation signal model includes twenty QPSK co-channel signal sources, Rayleigh fading MIMO channel with twenty receive antenna elements, and a signal-to-noise ratio of 20 dB.

CHAPTER 7

CONCLUSION

A one-stage blind source separation algorithm was developed using a sparse autoencoder framework for separation of multiple co-channel radio frequency signal sources. The one-stage BSS algorithm was successful in separating twenty co-channel overlapping quadrature phase-shift keying sources with each at the same power level. The BSS sparse autoencoder is able to solve for the channel matrix and signal sources within one optimization stage without a-priori knowledge of the channel or transmitted signals. The MIMO channel was considered to be a time-invariant Rayleigh flat-fading channel and the signal sources were assumed to exhibit sparse activity. Therefore, over a long-term time duration many signals are present, but over a short-term time scale there are only a minimal number of active transmissions such that the received signal takes on a sparse representation. Sparsity was exploited for separating the transmitted sources and is considered the *inductive bias* of the sparse autoencoder learning model.

The performance of the one-stage sparse autoencoder was compared to a two-stage learning model whereby the channel matrix and source signals are recovered using alternate optimization. Therefore, the two-stage process requires the BSS source recovery problem to be solved in parts where the ADMM LASSO was used for sparse coding and the method of optimal directions was used for dictionary learning of the channel matrix. On the other hand, the proposed BSS sparse autoencoder is able to solve for the channel matrix and source signals in one-stage and demonstrated superior performance over the two-stage BSS approach. The generalization performance showed that the sparse

autoencoder has a higher capacity to fit the data, but also requires more examples than the two-stage learning algorithm to generalize well. In addition, the support recovery versus signal-to-noise ratio and probability of detection versus false alarm were used as measures of performance as well. The support recovery is given by the Jaccard similarity index and is taken as the intersection of two vectors divided by the union of the two vectors. Hence, the Jaccard similarity provides an apposite measure of support recovery including detections and false alarms within the performance measure. The receiver operating characteristic curve provides a performance trade-off between the probability of detection and probability of false alarm. The one-stage BSS sparse autoencoder algorithm was shown to outperform the two-stage ADMM LASSO for both the support recovery and ROC curve.

Three factors were included in the BSS sparse autoencoder to impose sparsity on the hidden layer output of the encoder. First the sparse autoencoder in designed to have a wider hidden layer width than the input layer and output layer of the neural network. Second the cost function or loss function includes an ℓ_1 norm penalty on the hidden layer outputs. Third the hidden layer activation function is a soft-threshold operator also known as modReLU that supports complex-valued and real-valued signals.

The ability to generalize to new data is predominately what machine learning is all about. Generalization bounds provide an inequality that upper bounds the generalization error in terms of the training error, model capacity, number of samples, and probability that the deviation between the generalization and training error is greater than ϵ is some small value. Two generalization bounds were derived for regression using Hoeffding's inequality and the Rademacher complexity. Hoeffding's inequality was used for deriving a generalization bound that includes the capacity of the model. The Rademacher

generalization bound is a data-dependent bound and does not require the explicit cardinality of the hypothesis class to be defined. Both bounds include the signal power and noise power as part of the bounds. As the noise increases the generalization error gets worse as expected and is incorporated into the bounds.

Not only was the one-stage sparse autoencoder successful in separating RF co-channel signal sources, but it is extremely efficient in sparse coding of new examples via a simple matrix-vector product calculation that does not require any additional optimization steps as is the case in the two-stage process.

REFERENCES

[1] A. Naeem and H. Arslan, "Joint radar and communication based blind signal separation using a new non-linear function for fast-ICA," *IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 2021, pp. 1–5.

[2] D. A. Schuyler, B. A. Johnson, and M. D. McGough, "Blind co-channel source separation for pulse-on-pulse interference," *53rd Asilomar Conference on Signals, Systems, and Computers*, 2019, pp. 118–122.

[3] B. A. Johnson and D. A. Schuyler, "Blind co-channel source separation in sparse interferometric arrays," *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2018, pp. 1124–1128.

[4] L. Jiang, L. Li, and G. Zhao, "Pulse-compression radar signal sorting using the blind source separation algorithms," *International Conference on Estimation, Detection and Information Fusion (ICEDIF)*, 2015, pp. 268–271.

[5] Z. Luo, C. Li, and L. Zhu, "A comprehensive survey on blind source separation for wireless adaptive processing: principles, perspectives, challenges and new research directions," *IEEE Access*, vol. 6, pp. 66685–66708, 2018.

[6] C. Xu, T. Yang, and H. Song, "Spectrum sensing of cognitive radio for cubesat swarm network," *IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, 2021, pp. 1–8.

[7] Z. Luo, C. Li, and L. Zhu, "Full-duplex cognitive radio using guided independent component analysis and cumulant criterion," *IEEE Access*, vol. 7, pp. 27065–27074, 2019.

[8] M. E. Fouda, C.-A. Shen, and A. E. Eltawil, "Blind source separation for full-duplex systems: potential and challenges," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 1379–1389, 2021.

[9] M. E. Fouda, S. Shaboyan, A. Elezabi, and A. Eltawil, "Application of ICA on self-interference cancellation of in-band full duplex systems," *IEEE Wireless Communications Letters*, vol. 9, no. 7, pp. 924–927, 2020.

[10] H. Yang, H. Zhang, J. Zhang, and L. Yang, "Digital self-interference cancellation based on blind source separation and spectral efficiency analysis for the full-duplex communication systems," *IEEE Access*, vol. 6, pp. 43946–43955, 2018.

[11] S. Haykin and Z. Chen, "The cocktail party problem," *Neural Computation*, vol. 17, no. 9, pp. 1875-1902, 1 Sept. 2005.

[12] J. Yin, Z. Liu, Y. Jin, D. Peng and J. Kang, "Blind source separation and identification for speech signals," *International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC), 2017*, pp. 398-402.

[13] M. S. Pedersen, D. Wang, J. Larsen and U. Kjems, "Two-microphone separation of speech mixtures," *IEEE Transactions on Neural Networks*, vol. 19, no. 3, pp. 475-492, March 2008.

[14] R. Sharma, "Musical instrument sound signal separation from mixture using dwt and fast ICA based algorithm in noisy environment," *Materials Today: Proceedings*, vol. 29, pp. 536–547, 2020.

[15] N. Oosugi, K. Kitajo, N. Hasegawa, Y. Nagasaka, K. Okanoya, and N. Fujii, "A new method for quantifying the performance of EEG blind source separation algorithms by referencing a simultaneously recorded ECoG signal," *Neural Networks*, vol. 93, pp. 1–6, 2017.

[16] I. Daly, "Neural component analysis: A spatial filter for electroencephalogram analysis," *Journal of Neuroscience Methods*, vol. 348, p. 108987, Jan. 2021.

[17] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648-664, Dec. 2018.

[18] E. Alpaydin, *Introduction to Machine Learning.* 4th ed. Cambridge, MA, USA: The MIT Press, 2020.

[19] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. Pasadena, CA, USA: AMLBook, 2012.

[20] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, UK: Cambridge University Press, 2014.

[21] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York, USA: John Wiley & Sons, Inc., 2001.

[22] E. Bingham and A. Hyvärinen, "A fast fixed-point algorithm for independent component analysis of complex valued signals," *International Journal of Neural Systems*, vol. 10, no. 01, pp. 1–8, Feb. 2000.

[23] A. Hyvarinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[24] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[25] K. Abed-Meraim, Yong Xiang, J. H. Manton and Yingbo Hua, "Blind source-separation using second-order cyclostationary statistics," *IEEE Transactions on Signal Processing*, vol. 49, no. 4, pp. 694-701, April 2001.

[26] A. Ferreol, P. Chevalier and L. Albera, "Second-order blind separation of first- and second-order cyclostationary sources-application to AM, FSK, CPFSK, and deterministic sources," *IEEE Transactions on Signal Processing*, vol. 52, no. 4, pp. 845-861, April 2004.

[27] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948-958, June 2010.

[28] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," *IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP99 (Cat. No.99CH36258)*, Mar. 1999, vol. 5, pp. 2443–2446.

[29] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[30] L. N. Smith and M. Elad, "Improving dictionary learning: multiple dictionary updates and coefficient reuse," *IEEE Signal Processing Letters*, vol. 20, no. 1, pp. 79–82, Jan. 2013.

[31] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," *26th Annual International Conference on Machine Learning - ICML '09*, Montreal, Quebec, Canada, 2009, pp. 1–8.

[32] A. Dong, O. Simeone, A. M. Haimovich, and J. A. Dabin, "Blind sparse estimation of intermittent sources over unknown fading channels," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 9861–9871, Oct. 2019.

[33] P. Bofill and M. Zibulevsky, "Underdetermined blind source separation using sparse representations," *Signal Processing*, vol. 81, no. 11, pp. 2353–2362, Nov. 2001.

[34] P. Georgiev, F. Theis, and A. Cichocki, "Sparse component analysis and blind source separation of underdetermined mixtures," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 992–996, Jul. 2005.

[35] O. Simeone, "A Brief Introduction to Machine Learning for Engineers." *Foundations and Trends® in Signal Processing*, vol. 12, no. 3–4, pp. 200–431, 2018.

[36] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, Jan. 1998.

[37] D. L. Donoho and M. Elad, “Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization,” *Proceedings of the National Academy of Sciences*, vol. 100, no. 5, pp. 2197–2202, 2003.

[38] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Math. Control Signal Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[39] K. Hornik, M. B. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[40] C. C. Aggarwal, Neural networks and deep learning: a textbook. Cham, Switzerland: Springer, 2018.

[41] J. A. Dabin, A. M. Haimovich, J. Mauger and A. Dong, "Blind source separation with l1 regularized sparse autoencoder," *29th Wireless and Optical Communications Conference (WOCC)*, 2020, pp. 1-5.

[42] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed. Cambridge, MA: MIT Press, 2016.

[43] Z. Chen, C. K. Yeo, B. S. Lee and C. T. Lau, "Autoencoder-based network anomaly detection," *2018 Wireless Telecommunications Symposium (WTS)*, 2018, pp. 1-5.

[44] X. Wang, J. Gao, L. Gao and F. Wang, "Modulation recognition of emitter signals based on SDAE-ODCNN," *IET International Radar Conference (IET IRC 2020)*, 2020, pp. 1302-1307.

[45] S. Subray, S. Tschimben and K. Gifford, "Towards enhancing spectrum sensing: signal classification using autoencoders," *IEEE Access*, vol. 9, pp. 82288-82299, 2021.

[46] T. O’Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563-575, Dec. 2017.

[47] J. Bassey, X. Li and L. Qian, "An experimental study of multi-layer multi-valued neural network," *2nd International Conference on Data Intelligence and Security (ICDIS)*, 2019, pp. 233-236.

[48] J. W. Brown and R. V. Churchill, *Complex Variables and Applications*, 9th ed. New York, NY: McGraw-Hill Education, 2014.

[49] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. New York, NY, USA: Springer, 2010.

[50] M. Sadeghi, M. Babaie-Zadeh, and C. Jutten, “Dictionary learning for sparse representation: a novel approach,” *IEEE Signal Processing Letters*, vol. 20, no. 12, pp. 1195–1198, Dec. 2013.

[51] D. L. Donoho and I. M. Johnstone, “Ideal spatial adaptation by wavelet shrinkage,” *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[52] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*. Boca Raton, FL, USA: Chapman & Hall/CRC, 2015.

[53] R. Remi and K. Schnass, “Dictionary identification—sparse matrix-factorization via ℓ_1-minimization,” *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3523–3539, Jul. 2010.

[54] J. Huang and T. Zhang, “The Benefit of Group Sparsity,” *The Annals of Statistics*, vol. 38, no. 4, pp. 1978–2004, 2010.

[55] A. Maleki, L. Anitori, Z. Yang, and R. G. Baraniuk, “Asymptotic analysis of complex lasso via complex approximate message passing (camp),” *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4290–4308, Jul. 2013.

[56] G. Taubock and F. Hlawatsch, "A compressed sensing technique for OFDM channel estimation in mobile environments: Exploiting channel sparsity for reducing pilots," *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 2885-2888.

[57] A. Balatsoukas-Stimming, "Non-linear digital self-interference cancellation for in-band full-duplex radios using neural networks," *IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1-5.

[58] M. Arjovsky, A. Shah, and Y. Bengio, “Unitary evolution recurrent neural networks,” *33rd International Conference on International Conference on Machine Learning - Volume 48*, New York, NY, USA, 2016, pp. 1120–1128.

[59] T. Scarnati and B. Lewis, "Complex-valued neural networks for synthetic aperture radar image classification," *IEEE Radar Conference (RadarConf21)*, 2021, pp. 1-6.

[60] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. Pal, “Deep complex networks,” *Sixth International Conference on Learning Representations*, 2018, pp. 1-19.

[61] S. Li, W. Zhang and Y. Cui, "Jointly sparse signal recovery via deep auto-encoder and parallel coordinate descent unrolling," *IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1-6.

[62] S. Li, W. Zhang, Y. Cui, H. V. Cheng and W. Yu, "Joint design of measurement matrix and sparse support recovery method via deep auto-encoder," *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1778-1782, Dec. 2019.

[63] Q. Yuan, D. Li, Z. Wang, C. Liu and C. He, "Channel estimation and pilot design for uplink sparse code multiple access system based on complex-valued sparse autoencoder," *IEEE Access*, 2019.

[64] L. V. Ahlfors, Complex Analysis: *An Introduction to the Theory of Analytic Functions of One Complex Variable*, 3rd ed., New York, NY, USA: McGraw-Hill, 1979.

[65] D. H. Brandwood, "*A Complex Gradient Operator and Its Application in Adaptive Array Theory*," IEEE Proceedings, F: Communications, Radar and Signal Processing, Vol. 130, No. 1, 1983, p. 1116.

[66] J. W. Brown and R. V. Churchill. *Complex Variables and Applications.* 9th ed. New York, NY, USA: McGraw-Hill, 2014.

[67] E. G. Larsson and P. Stoica, *Space-Time Block Coding for Wireless Communications*. New York, NY, USA: Cambridge University Press, 2003.

[68] W. C. Jakes, *Microwave Mobile Communications (An IEEE Press Classic Reissue)*. 2nd ed. New York, NY, USA: Wiley-IEEE Press, 1994.

[69] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Communications*, Vol. 6, No. 3, 1998, pp. 311-335.

[70] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, N.J., USA: Prentice Hall PTR, 2002.

[71] J. R. Hampton, *Introduction to MIMO Communications*. New York, NY, USA: Cambridge University Press, 2013.

[72] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4-16, Jan 1986.

[73] K. P. Murphy, *Machine Learning: A Probabilistic Perspective.* Cambridge, MA, USA: MIT Press, 2012.

[74] D. Kounades-Bastian, L. Girin, X. Alameda-Pineda, R. Horaud and S. Gannot, "Exploiting the intermittency of speech for joint separation and diarization," IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017, pp. 41-45.

[75] A. T. Koc, S. C. Jha, R. Vannithamby and M. Torlak, "Device power saving and latency optimization in LTE-A networks through DRX configuration," *IEEE Transactions on Wireless Communications*, vol. 13, no. 5, pp. 2614-2625, May 2014.

[76] H. S. Jang, H. Jin, B. C. Jung and T. Q. S. Quek, "Resource-optimized recursive access class barring for bursty traffic in cellular IoT networks," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11640-11654, 15 July15, 2021.

[77] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran and S. Guizani, "Internet-of-things-based smart cities: recent advances and challenges," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 16-24, Sept. 2017.

[78] D. Barber and A. T. Cemgil, "Graphical models for time-series," *IEEE Signal Processing Magazine*, vol. 27, no. 6, pp. 18-28, Nov. 2010.

[79] D. P. Bertsekas, and J. N. Tsitsiklis, *Introduction to Probability*. Belmont, Massachusetts, USA: Athena Scientific, 2008.

[80] B. Sklar, and F. Harris, *Digital Communications: Fundamentals and Applications*. 3rd ed., Upper Saddle River, NJ, USA: Pearson, 2020.

[81] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *Fourteenth International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, Apr. 2011, vol. 15, pp. 315–323.

[82] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," *27th International Conference on International Conference on Machine Learning*, Madison, WI, USA, 2010, pp. 399–406.

[83] S. Haykin, *Neural networks and learning machines*. 3rd ed., Upper Saddle River, NJ, USA: Pearson Education, 2009.

[84] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," *3rd International Conference on Learning Representations*, ICLR 2015, San Diego, CA, USA, May 7-9, 2015.

[85] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization*. 2nd ed., Philadelphia, PA, USA: SIAM, 2009.

[86] S. M. Kay, *Fundamentals of Statistical Signal Processing: Vol. 2*. Upper Saddle River, NJ, USA: Prentice-Hall PTR, 1998.

[87] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd ed., Cambridge, MA, USA: MIT Press, 2018.

[88] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian, "Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks," *Journal of Machine Learning Research*, vol. 20, no. 63, pp. 1–17, 2019.

[89] D. Pollard, "Empirical processes: theory and applications," *NSF-CBMS Regional Conference Series in Probability and Statistics*, vol. 2, pp. i–86, 1990.

[90] M. Anthony and P. L. Bartlett, *Neural Network Learning: Theoretical Foundations*, 1st ed. Cambridge, UK: Cambridge University Press, 1999.

[91] V. Koltchinskii, "Rademacher penalties and structural risk minimization," *IEEE Transactions on Information Theory*, vol. 47, no. 5, pp. 1902–1914, 2001.

[92] M. Ranzato, C. Poultney, S. Chopra, and Y. Cun, "Efficient learning of sparse representations with an energy-based model," *Advances in Neural Information Processing Systems*, 2007, vol. 19.

[93] M. Ranzato, Y. Boureau, and Y. Cun, "Sparse feature learning for deep belief networks," *Advances in Neural Information Processing Systems*, 2008, vol. 20.

[94] B. O. Ayinde and J. M. Zurada, "Deep learning of constrained autoencoders for enhanced understanding of data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 3969–3979, 2018.

[95] Z. Liu and L.-L. Yang, "Sparse or dense: a comparative study of code-domain noma systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 4768–4780, 2021.

[96] H. Li and T. Adali, "Optimization in the complex domain for nonlinear adaptive filtering," *Fortieth Asilomar Conference on Signals, Systems and Computers*, 2006, pp. 263–267.

[97] D. Angelosante, G. B. Giannakis, and N. D. Sidiropoulos, "Estimating multiple frequency-hopping signal parameters via sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5044–5056, 2010.

[98] S. G.Sathyanarayana, B. Ning, S. Hu, and J. A. Hossack, "Comparison of dictionary learning methods for reverberation suppression in photoacoustic microscopy : Invited presentation," *53rd Annual Conference on Information Sciences and Systems (CISS)*, 2019, pp. 1–4.