Improving and Unfolding Statistical Models of Nonstationary Signals

Scott Thomas Wisdom

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2017

Reading Committee:

Les E. Atlas, Chair

James Pitton, Chair

Mari Ostendorf

Program Authorized to Offer Degree:
Department of Electrical Engineering

University of Washington

Abstract

Improving and Unfolding Statistical Models of Nonstationary Signals

Scott Thomas Wisdom

Co-Chairs of the Supervisory Committee:
Professor Les E. Atlas
Electrical Engineering

Affiliate Associate Professor James Pitton
Electrical Engineering

Improving the modeling and processing of nonstationary signals remains an important yet challenging problem. In the past, the most effective approach for processing these signals has been statistical modeling. Statistical models can effectively encode domain knowledge and lead to principled algorithms for the fundamental tasks of enhancement, detection, and classification. However, the performance of statistical models can be limited because they inherently make assumptions about the distribution of the data. Deep neural networks, in contrast, have recently outperformed state-of-the-art statistical models of nonstationary signals. Deep neural networks are completely data-driven, and learn to set their parameters by training on large datasets that are assumed to match the distribution of the data.

This dissertation follows two approaches for improving modeling and processing of nonstationary signals. The first approach examines conventional model assumptions and suggests improvements that lead to improved performance for processing nonstationary signals. Specifically, noncircular distributions of the complex-valued short-time Fourier transform are shown to improve detection of realistic nonstationary signals. Then the parameterization of a recently-proposed recurrent neural network for processing nonstationary signals is reexamined. By using an optimization method that preserves the capacity of the recurrence matrix,

superior performance is achieved on a battery of benchmarks that test the ability of recurrent neural networks to process nonstationary signals.

The second approach uses the recently-proposed framework of deep unfolding, which provides a principled means of transforming statistical model inference algorithms into deep networks. This dissertation expands the deep unfolding framework specifically for nonstationary signals. Using this framework, a model-based explanation is provided for state-of-the-art recurrent neural architectures, including gated recurrent unit and unitary recurrent neural networks. Additionally, deep unfolding results in deep network architectures that arise in principled ways from statistical model assumptions. This statistical model foundation provides initializations for the unfolded networks, which lead to better generalization, faster training, and competitive or superior performance on a variety of tasks, including single- and multichannel acoustic source separation and classification of acoustic signals.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would first like to thank both of my advisors, Les Atlas and Jim Pitton. Les believed in me early on, and he has been a consistent source of encouragement and support over the past 5 years. I am also very grateful for the academic freedom he has allowed me. Jim has been an amazing and supportive mentor, always giving freely of his valuable time, and I have grown so much as a researcher because of our many long discussions. Thanks also to the members of my PhD committee, including Mari Ostendorf, John Hershey, Mike Seltzer, and Marina Meila, who have all provided important advice throughout my grad school career. I am especially grateful to John Hershey for bringing me on as an intern at MERL in the winter of 2015 and inviting me to join the Far Field Speech team at the 2015 Jelinek Workshop held in summer 2015 in Seattle. I learned an incredible amount through these experiences and countless technical discussions with John. Thanks to Jonathan Le Roux, who also hosted me at MERL and has also been a continual source of advice and technical knowledge. Through my internship and the workshop, I was very fortunate to collaborate with and learn from many exceptional individuals, including Shinji Watanabe, Hakan Erdogan, Mike Mandel, and Zhuo Chen.

This work was funded by the Office of Naval Research. I am very grateful to John Tague of ONR for his thoughtful suggestions and advice, and for providing funding for my entire graduate school career.

I would also like to thank my past and present labmates and colleagues at UW, including Tommy Powers, Greg Okopal, Brad Ekin, Elliot Saba, Nicole Nichols, Brian King, Pascal Clark, Kai Wei, Bill Kooiman, Majid Mirbagheri, Tyler Ganter, David Dolengewicz, Eldridge Alcantara, and Ruobai Wang. I am especially grateful to Tommy and Greg, who were always

willing to listen and discuss my latest crazy idea. Thanks also to all the guys at APL for advice and comradery during coffee hour, including David Krout, Jack McLaughlin, Lane Owsley. I would also like to thank Josh Smith, Alanson Sample, and Ben Waters, who I worked with when I first came to UW. They helped me realize my passion for research and inspired me to pursue a PhD.

Finally, thank you to my family and friends for all their love and support over the years. My deepest thanks go to my wonderful wife Lindsay, for her incredible patience, kind encouragement, and steadfast support through grad school. I couldn't have done it without you. Thanks to my parents, my sister Rosie, my grandparents, the Rogers, the Myersons, and the Johnsons, who have always supported and encouraged me.

DEDICATION

To my amazing wife Lindsay.

Chapter 1

INTRODUCTION

Statistical modeling of nonstationary signals, especially acoustic signals like human speech, remains an important and challenging problem. Such models enable detection, estimation, and classification of such signals, which are essential for many applications. For example, accurate automatic speech recognition systems require reliable detection of speech signals in the presence of noise, effective enhancement of noisy speech, accurate parameter estimation of spatial parameters for effective beamforming from multimicrophone recordings, and classification of speakers and noise environments. Passive sonar applications require detection and classification of targets of interest in acoustically cluttered environments.

This dissertation is primarily concerned with nonstationary signals, which constitute a very large and infinite set. This set is vast because nonstationary signals are defined not by a property they possess, but by one that they lack, namely stationarity. Stationarity means that, in expectation, the statistical moments of these signals do not change over time. For example, if each sample of a discrete-time signal is drawn independently from some distribution with constant parameters, then the signal is stationary because the statistics of the distribution are not dependent on the time index of the sample.

One approach for stationary signals is to define an explicit statistical model. For example, a wider class of stationary signals are the second-order stationary, or wide-sense stationary (WSS) signals, whose first-order moment (i.e., the mean) and autocovariance do not depend on the absolute time index. This is a weaker version of stationary, which requires that all statistical moments are time-invariant. WSS signal models have been very popular historically because they are tractable, in the sense that their estimators are simple and it is easy to derive statistically optimal algorithms based on the model. Because of this ease,

nonstationary signals are often modeled as locally WSS. This locally WSS assumption holds to some degree for real-world nonstationary signals, but often these methods are limited because they do not model the change or innovation between the short-time, locally WSS segments. Moreover, the local duration for which the WSS assumption holds can vary over time. This example reveals a limitation of statistical models: though statistical assumptions encode domain knowledge, these assumptions can be rigid and restrictive. Performance of algoritms based on these models can degrade if the statistical assumptions are mismatched to the data's distribution.

Another type of model often used for nonstationary signals is a neural network. Recurrent neural networks, which are designed to process sequences of feature vectors, are well-suited for nonstationary signals. Neural networks are a totally data-driven approach, and the parameters of these neural networks need to be trained on a large dataset. Thus, they do not require a specific statistical model; rather, they automatically learn a setting of their parameters such that their output minimizes a training loss function.

As long as enough representative data is available to train the parameters of the neural network, these models work well. One could argue that neural networks are able to adaptively learn implicit statistical assumptions about the data during training, or at least to modify the implicit assumptions imposed by the network architecture. However, since neural networks are agnostic to explicit knowledge of the data's distribution, this ability to adapt comes at a price: neural networks are inherently "black-box" models, and unlike explicit statistical model parameters, neural network parameters are not particularly interpretable. Because of this, the choice of network architecture is a nontrivial problem, since specific prior domain knowledge about the data cannot be used to guide design choices. Furthermore, improvement of existing network architectures is an empirical process of trial-and-error.

1.1 Main approaches

Given this outlook, this dissertation takes two specific approaches to improve modeling and processing of nonstationary signals.

1.1.1 Approach 1: Improving Models of Nonstationary Signals by Addressing Conventional Assumptions

The first approach consists of examining conventional assumptions of models that are used to represent and process nonstationary signals. This approach is applied to two different assumptions.

The first assumption is based on locally WSS reasoning: that complex-valued short-time frequency spectra exhibit second-order circularity. By relaxing this assumption, we will show that the second-order noncircularity of short-time spectra can be exploited to improve statistical signal processing of certain subclasses of nonstationary signals, particularly harmonic tones and transients.

The second assumption this dissertation examines is the representational capacity of a recently-proposed recurrent neural network (RNN), the unitary RNN (uRNN) to model the dynamics of nonstationary data over time. The original uRNN uses a particular parameterization of the unitary recurrence matrix. We will see that this parameterization is actually suboptimal, in the sense that it provably cannot represent all possible unitary matrices. A different method of optimization is proposed for this unitary matrix, which allows all possible unitary matrices to be optimized over. This full representational capacity leads to improved performance on a variety of benchmark tasks designed to test the ability of RNNs to process nonstationary input signals.

1.1.2 Approach 2: Unfolding Statistical Model Inference Algorithms to Deep Networks

The second approach taken in this dissertation relies on learning and refining models of nonstationarity directly from data. This approach leverages recent advances in deep learning, which is a method for automatically tuning complicated nonlinear deep neural networks on large datasets. As long as the networks have enough representational capacity, they are able to learn and set their parameters such that they optimize performance for a particular regression or classification task.

However, this dissertation goes one step beyond conventional deep learning approaches. Conventional deep learning constructs neural networks by combining components that have been found to work well empirically by the deep learning community. Once conventional building blocks have been combined, their weights are randomly initialized using schemes that have been shown to work well empirically.

This dissertation improves on this empirical procedure by instead using existing statistical models and their inference algorithms as a starting point for constructing and initializing deep architectures. This framework, called *deep unfolding*, was recently proposed by Hershey et al. [65], and exploits model-based thinking to inspire, build, initialize, and improve deep networks. This dissertation expands on Hershey et al.'s initial work, providing new connections between statistical models and deep networks through the idea of deep unfolding. These new connections particularly focus on sequential models, which are well-suited for processing sequences of features extracted from nonstationary signals. We will find that not only can new types of deep networks be created through a principled method, but also that many of the computational structures encountered during this process help explain the success of existing deep network architectures. Deep unfolding also facilitates better interpretability because the unfolded networks are constructed through model-based thinking. Furthermore, the model-based foundation of unfolded networks provides interpretability of the trained weights and prescribes improved initializations, which speed up training convergence and promote better solutions once training has converged.

1.2 Organization

This dissertation is organized as follows. Chapter 2 will review necessary background concepts and survey relevant literature. Then, chapters 3, 4, and 5 present the first main contribution of this dissertation: improving models of nonstationary signals by addressing conventional assumptions. Chapter 3 describes noncircular distributions for complex-valued random variables, describes how certain types of nonstationary signals exhibit noncircularity in the short-time Fourier transform, and provides practical estimators for the parameters of

noncircular distributions from the short-time Fourier transform. Chapter 4 describes the application of noncircular distributions and their estimators to detection of two important types of nonstationary signals: speech and acoustic transients. Chapter 5 describes an improvement to the recently-proposed unitary recurrent neural network, which results in improved performance on several benchmarks that test recurrent neural networks' ability to process nonstationary input sequences.

Next, chapters 6 and 7 describe the second main contribution of this dissertation, which is deep unfolding of statistical models, particularly for nonstationary signals. Chapter 6 describes the general framework of deep unfolding and provides several examples of unfolding statistical models to deep networks. Through these examples, we will also see how certain computational structures that are canonical in deep learning actually arise from model-based thinking. Then, chapter 7 describes the application of deep unfolding to practical problems involving nonstationary signals. These applications include compressed sensing of images, single and multichannel speech separation, and classification of nonstationary acoustic signals. Finally, chapter 8 provides a summary of the main results and directions for future work.

1.3 Notation and Conventions

Lowercase bold variables are vectors, e.g. $\mathbf{x}$, and uppercase bold variables are matrices or tensors, e.g. $\mathbf{X}$. Non-bold lowercase variables with a subscript refer to elements of a vector, e.g. x_i is the ith element of $\mathbf{x}$. Non-bold uppercase variables with (multiple) subscripts refer to an individual element of a matrix or tensor, e.g. $X_{i,j}$ is the (i,j)th element of $\mathbf{X}$. Constant scalars that indicate the number of elements are non-bold and uppercase, while an index within the set of elements is non-bold and lowercase. For example, for a D-dimensional vector $\mathbf{x} \in \mathbb{R}^D$, the sum of the elements is indicated by $\sum_{d=0}^{D-1} x_d$.

Bold uppercase variables with one or more colons in the subscript refer to slices or subsets of a matrix or tensor; e.g., for a matrix $\mathbf{S} \in \mathbb{R}_+^{F \times T}$, the tth column will be referred to as $\mathbf{S}_t$, $\mathbf{S}_{:,t}$, or $\mathbf{S}_{1:F,t}$, and the fth row as $\mathbf{S}_{f,:}$ or $\mathbf{S}_{f,1:T}$. Horizontal concatenation will be indicated by

a comma "," and vertical concatenation will be indicated by a semicolon ";". The notation $(\cdot)^*$ means element-wise complex conjugation. A $(\cdot)^\mathsf{T}$ indicates the transpose of a vector or matrix, and $(\cdot)^\mathsf{H}$ is the conjugate transpose of a vector of matrix, equivalent to $(\cdot)^{*\mathsf{T}}$ or $(\cdot)^{\mathsf{T}*}$. The symbol "$\odot$" means elementwise multiplication between two vectors, matrices, or tensors. For elementwise multiplication, broadcasting of the dimensions is assumed. For example, the tensor $\mathbf{Z} = \mathbf{X} \odot \mathbf{Y}$, where $\mathbf{X} \in \mathbb{R}^{1\times M\times P}$ and $\mathbf{Y} \in \mathbb{R}^{N\times 1\times P}$, is of shape $N \times M \times P$.

The variable ξ is reserved for continuous frequency in units of Hertz, while f is usually used as an index for the frequency bins of a discrete Fourier transform.

1.4 Acknowledgements and Relation to Other Published Works

Some content in this dissertation has also been published in peer-reviewed papers. In this section, co-authors of these papers are acknowledged and the specific source papers are specified. Sections 3.3, 3.3.2, 4.1, 4.1.1, 4.1.2, and 4.1.3 include text from a paper presented at the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2015 [162] with Greg Okopal, Les Atlas, and James Pitton as co-authors. Sections 3.2, 3.3, 3.3.1, and 4.2 include text from an invited paper presented at the IEEE Sensor and Multichannel (SAM) 2016 conference [160] with Les Atlas and James Pitton as co-authors, and an invited paper presented at the Asilomar Conference on Signals, Systems, and Computers 2016 [157], with Les Atlas, James Pitton, and Greg Okopal as co-authors. Sections 3.3.3, 4.2, and 4.2.1 include text from an invited paper presented at the Asilomar Conference on Signals, Systems, and Computers 2016 [157], with Les Atlas, James Pitton, and Greg Okopal as co-authors. Chapter 5 includes material from a paper presented at the Advances in Neural Information Processing Systems (NIPS) 2016 [164] with Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas as co-authors. Sections 6.1, 7.3, 7.3.1, 7.3.2, 7.3.3, and 7.3.4 include text from a paper presented ICASSP 2016 [161] with John Hershey, Jonathan Le Roux, and Shinji Watanabe as co-authors. Section 7.1.1 includes text from a paper presented at the NIPS 2016 Workshop on Interpretable Machine Learning for Complex Systems [165] with Thomas Powers, James Pitton, and Les Atlas as co-authors. Sections 2.4, 6.3.3, and 7.1

include text from a paper presented at ICASSP 2017 [158] with Thomas Powers, James Pitton, and Les Atlas as co-authors. Sections 6.1, 7.2, 7.2.1, 7.2.2, 7.2.3, and 7.2.4 include text from a paper to be presented at the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) 2017 [166] with Thomas Powers, James Pitton, and Les Atlas as co-authors.

The SISTA and SISTA-RNN work presented in chapters 6 and 7 also deserve an acknowledgement, since they were inspired by personal communication with John Hershey. After we published the NIPS 2016 paper [164] on unitary recurrent neural networks together, he asked the question, "what is the model that unfolds to the unitary recurrent neural network?" The answer, as this dissertation describes, is sequential sparse coding with a constraint on the sparse coding dictionary, which leads to an effective class of unfolded deep recurrent neural networks.

Chapter 2
BACKGROUND

This chapter covers required background material and reviews relevant prior work from the literature. First, the short-time Fourier transform is described in section 2.1, which will be the primary representation we will use in this dissertation to represent nonstationary signals. Then, section 2.2 describes the general framework of statistical modeling that serves as a foundation for the ideas in this dissertation. Section 2.3 reviews specific statistical models for the short-time Fourier transform. Next, section 2.4 reviews the use of sparsity in statistical models, which is an important concept in this dissertation. Section 2.5 provides an overview of deep learning and relevant concepts. Finally, section 2.6 introduces the recently-proposed deep unfolding framework. This chapter then concludes with a summary of related work, the problem statement, and the main contributions of this dissertation.

2.1 The Short-Time Fourier Transform (STFT)

Most generative models of nonstationary signals represent these signals in the time-frequency domain. The short-time Fourier transform (STFT) is arguably the most popular choice, especially for enhancement and separation of audio. The STFT is complex-valued, and consists of the discrete Fourier transform (DFT) of short windowed frames of the original sampled time series. For D samples of real-valued a discrete-time signal x_d, $d = 0..D-1$, the STFT of $x_{0:D-1}$ using window $\mathbf{w} \in \mathbb{R}^M$ with hop M_{hop} and DFT length M_{DFT} is defined as

$$X_{f,t} = \text{STFT}_{f,t}\{x_{1:D}\} = \sum_{m=0}^{M_{DFT}-1} x_{m+tM_{hop}} w_m \exp\left\{-\text{j}2\pi m \frac{f}{M_{DFT}}\right\}, \tag{2.1}$$

with $f = 0..F-1$, where $F = (M_{DFT}-1)/2$ is the dimension of the one-sided DFT. The variable F is used instead of M_{DFT} because for real-valued signals, the DFT coefficients

at negative frequencies are equal to the conjugate of the coefficients at the corresponding positive frequencies: $X_{f,t} = X^*_{-f,t}$, so only the DC term with $f = 0$ and the $F - 1$ positive frequencies are necessary to represent the short-time spectrum. We will restrict our attention to real-valued signals in this dissertation.

For speech recognition, the standard Mel-frequency cepstral coefficients (MFCCs) features can be computed from the STFT by applying triangular Mel filters to the magnitude of the STFT, taking the log, and performing an inverse discrete cosine transform (DCT). Log-filterbank energy features are simply the application of triangular Mel filters to the magnitude of the STFT, followed by taking the log.

2.2 Statistical Modeling

Since both main contributions of this dissertation rely on the concept of statistical models, and since the conventional approach to derive many algorithms for estimating, detecting, and classifying data has been to rely on such models, let us review the basic framework. A statistical model of some observed data consists of a set of assumptions about the data's distribution. These assumptions define the random variables and parameters of the statistical model, as well as the likelihood function for any particular setting of the random variables.

To illustrate the formulation of a statistical model, consider a simple example where we are able to take scalar measurements x_i of some physical process. Then we can model this physical process using a random variable x with probability distribution $p(x|\theta)$ with parameters θ. For example, we could choose $p(x|\theta)$ to be a Gaussian distribution, with parameters $\theta = \{\mu, \sigma^2\}$, where μ is the mean and σ^2 is the variance. Then the likelihood of the parameters θ given that the random variable x equals some value x_0 is

$$p(x = x_0|\theta = \{\mu, \sigma^2\}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_0 - \mu)^2}{2\sigma^2}\right). \tag{2.2}$$

If we do not know the values of the parameters μ and σ^2 *a priori*, then we can estimate them from data, assuming we have I samples of x, x_i for $i = 0..I - 1$. The maximum

likelihood estimator $\hat{t}$ of a parameter $t \in \theta$ is the solution of the optimization problem

$$\hat{t} = \underset{t}{\operatorname{argmax}} \quad \prod_{i=0}^{I-1} p\big(x = x_i \mid \theta = \{t\} \cup (\theta \setminus t)\big). \tag{2.3}$$

Often, the log of the likelihood has a simpler or more tractable form. Since the log function is monotonic, then minimizing the negative log-likelihood is equivalent to maximizing the likelihood. Thus, we can write the maximum likelihood estimation problem as

$$\hat{t} = \underset{t}{\operatorname{argmin}} \quad -\sum_{i=0}^{I-1} \log p\big(x = x_i \mid \theta = \{t\} \cup (\theta \setminus t)\big). \tag{2.4}$$

For example, when $p(x|\theta)$ is a Gaussian distribution, maximum likelihood estimation of the mean μ is a least-squares problem:

$$\hat{\mu} = \underset{\mu}{\operatorname{argmin}} \quad \sum_{i=0}^{I-1} (x_i - \mu)^2 \tag{2.5}$$

which yields a closed-form solution of $\hat{\mu} = \frac{1}{I} \sum_i x_i$.

In a Bayesian setting, parameters θ are also assigned a prior distribution $p(\theta)$, which serves as a regularizer in the estimation problem (2.4). This is especially easy to see for the negative log-likelihood:

$$-\log\{p(x|\theta)p(\theta)\} = -\log p(x|\theta) - \log p(\theta). \tag{2.6}$$

For example, if we know that the mean μ is sparse (that is, has only a few nonzero elements), then we can assign a Laplacian distribution prior with location $a = 0$ and scale b:

$$p(\mu) = \frac{1}{2b} \exp\big(-\frac{|\mu|}{b}\big). \tag{2.7}$$

Assigning this prior to μ leads to the following regularized optimization problem for estimation of μ:

$$\hat{\mu} = \underset{\mu}{\operatorname{argmin}} \quad \sum_{i=0}^{I-1} (x_i - \mu)^2 + \lambda|\mu|, \tag{2.8}$$

where $\lambda = 1/b$ is the regularization parameter, equivalent to the inverse scale of the Laplacian distribution. We will encounter this problem later in this dissertation, where algorithms it will be discussed.

Sometimes we cannot directly observe some mechanisms of a physical process. For example, in speech we can only directly observe the audio signal, and not the words that are being spoken. To account for these mechanisms in a statistical model, we can use hidden variables, which we will denote by ϕ. These variables are not directly observed, but by including them in the model we can derive statistically optimal inference algorithms to estimate their value given the observed data and the statistical model parameters. If we also do not know the values of the model's parameters, we can also make the parameters of the model hidden random variables that can be inferred. In regression and classification problems, the dependent variables or labels y are treated as hidden variables that are inferred from the data.

If both hidden variables ϕ and hidden model parameters θ need to be inferred given I data examples, the training optimization problem is

$$\underset{\theta}{\text{maximize}} \quad \sum_{i=1}^{I} \log \int_{\phi_i} p\left(x_i, \phi_i | \theta\right). \tag{2.9}$$

The standard approach to solve this problem is the expectation-maximization (EM) algorithm [32], which alternates between computing the expected value of the log-likelihood function with respect to the conditional distribution of the hidden variables given the observed data and the current settings of the parameters (the expectation step) and estimating new values for the parameters that maximize this new expected log-likelihood function (the maximization step). Mathematically, for K alternating iterations, this procedure can be described as

$$\begin{aligned}
&\text{for } k = 1 : K, \\
&\quad q^{(k)}(\theta|\theta^{(k-1)}) = \quad \sum_i E_{\phi_i|x_i,\theta^{(k-1)}}\left\{ \log p(x_i, \phi_i|\theta^{(k-1)})\right\} && (2.10) \\
&\quad \theta^{(k)} = \underset{\theta}{\text{argmax}} \quad q^{(k)}(\theta|\theta^{(k-1)}) && (2.11)
\end{aligned}$$

The E-step (2.10) can be thought of "filling in" the missing values of the hidden variables given the observed data and current estimate of the model parameters. The EM algorithm produces monotonic increase of the log-likelihood function, but the alternating updates only converge to a local maximum instead of the global maximum.

2.3 Statistical Modeling of the Complex-Valued STFT

To process nonstationary signals, various statistical models of the STFT have been proposed. Here we focus on the most relevant models. The first model described is unsupervised, in the sense that its parameters are not trained from data. The second model, nonnegative matrix factorization, is actually a family of models and relies on supervised learning of patterns across frequency for signal spectra from data.

2.3.1 Conventional Model

The diagonal-covariance zero-mean circular Gaussian model of the STFT was one of the earliest successful models used for speech enhancement, and is still very widely used because of its simplicity, efficiency, and compatibility with modeling reverberation in the frequency domain. Furthermore, this model is based on the complex-valued STFT, which allows perfect reconstruction back to the time domain. Throughout the rest of this dissertation, this model will be referred to as the "conventional model." This model should not be confused with real-valued, diagonal-covariance, non-zero mean Gaussian mixture models of MFCCs, which were the most effective acoustic model for speech recognition before the advent of deep neural networks [132]. The conventional model has great utility for speech enhancement since it directly models linear complex-valued STFT coefficients, allowing inversion of the STFT to synthesize an enhanced audio signal. Figure 2.1 illustrates the various assumptions of the model.

An unsupervised method that only relies on estimation of the background noise spectrum was proposed by Ephraim and Malah [37, 38], who recommended modeling the STFT coefficients of speech and stationary background noise as independent zero-mean circular Gaussians across time and frequency. Under this model, they derived a minimum mean-square error (MMSE) estimator of both the magnitude [37] and the log-magnitude [38] of the speech STFT coefficients. The MMSE estimator for the log-magnitude is sometimes called the "log-spectral amplitude" (LSA) estimator. The stationary background noise spec-

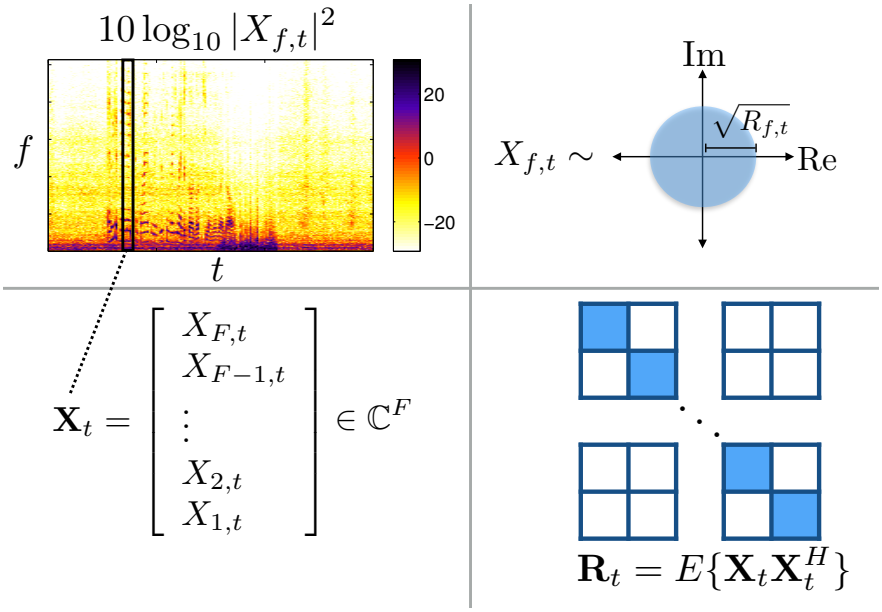

Figure 2.1: Illustration of diagonal-covariance, zero-mean, circular Gaussian model of STFT. Upper left: a spectrogram of a speech utterance embedded in car noise. Lower left: each column of the STFT is a F-dimensional complex-valued vector. Lower right: the covariance is assumed diagonal. Upper right: the Gaussian distribution for $X_{f,t}$ is zero-mean and circular.

trum is estimated from noise-only periods, as indicated by a voice activity detector (VAD). A popular choice for estimating the noise variance is Martin's minimum statistics [102]. The Ephraim and Malah algorithm was improved by Cohen [25] by incorporating a probability of speech presence, which he called the "optimal modified LSA" (OM-LSA) algorithm. The OM-LSA model was extended to also perform dereverberation by Habets [62], and Wisdom et al. [163] showed that combining chirp-based overcomplete dictionaries with the STFT further improved enhancement and dereverberation performance of Habets's algorithm.

2.3.2 Prior Work on Noncircular Extensions of the Conventional Model

Recently, noncircular models have been proposed for modeling speech and other nonstationary natural signals in time-frequency. Millioz and Martin [104] studied the circularity of the STFT coefficients of nonstationary signals in white Gaussian noise, and used noncircularity to segment synthetic and natural signals in time-frequency. Rivet et al. [123] and Clark [23] empirically observed that speech signals exhibit spectral noncircularity. Benesty et al. [11, 10] proposed a widely linear speech enhancement filter that trades off between noise reduction and distortion, and showed that exploiting the noncircularity of speech using widely linear processing resulted in improved output SNRs of up to 1dB.

For multichannel mixtures of nonstationary acoustic signals, the decorrelation ability of the circularity spectrum has an intriguing application: if the narrowband assumption is satisfied for all sources, the number of channels is equal to the number of sources, and if the mixing matrix at each time and frequency is non-singular, then the circularity spectrum $\boldsymbol{\kappa}$ at each time and frequency will consist of the circularity coefficients of the sources, sorted in decreasing order. Using this property, Okopal et al. [114, 113] demonstrated recovery of speech information in -25dB SNR for a synthetic anechoic two-channel mixture of speech and noise.

The contribution in this dissertation goes beyond this previous work by providing practical estimators of noncircularity from the STFT directly and studying their properties. Furthermore, these estimators are used to improve detection of nonstationary signals on realistic datasets.

2.3.3 Nonnegative matrix factorization

Speech enhancement and detection can be improved by using statistical models of speech that are trained from data. Furthermore, background noise is rarely perfectly stationary, and it is advantageous to learn noise models as well. A popular family of algorithms consists of learning nonnegative patterns across frequency using nonnegative matrix factoriza-

tion (NMF), originally proposed by Lee and Seung [89]. NMF is a popular method for modeling nonnegative functions of the STFT, such as magnitude, magnitude-squared and log-magnitude. NMF has successfully been applied to a wide variety of tasks, including voice activity detection [52], acoustic event detection [64, 27, 103, 13], source separation [135, 7, 40, 47, 46, 59, 1, 72, 83, 90, 91, 115, 133, 87], and marine mammal classification [110]. Since NMF is based on a family of statistical models [42], it is easy to extend in many ways, such as adding sparsity [34, 140, 87], convolution [112], dynamics across time, [43, 134], and phase awareness [75, 79, 98]. This section will review the basic formulation of NMF.

NMF decomposes an (element-wise) nonnegative data matrix $\mathbf{X} \in \mathbb{R}_+^{F \times T}$ into the product of a nonnegative dictionary $\mathbf{W} \in \mathbb{R}_+^{F \times N}$ and nonnegative coefficients (sometimes called "activations") $\mathbf{H} \in \mathbb{R}_+^{N \times T}$:

$$\mathbf{X} = \mathbf{WH} = \sum_{n=0}^{N-1} \mathbf{W}_{:,n}\mathbf{H}_{n,:}. \tag{2.12}$$

To estimate $\mathbf{W}$ and $\mathbf{H}$ from data $\mathbf{X}$, various cost functions have been proposed. Many popular cost functions, especially for audio source separation where $\mathbf{X}$ is a spectrogram, can be seen as special cases of a beta divergence [42], given by

$$D_\beta\left(\mathbf{X}||\mathbf{WH}\right) = \sum_{f,t} d_\beta\left(\mathbf{X}_{f,t}||\mathbf{W}_{f,:}\mathbf{H}_{:,t}\right) \tag{2.13}$$

where the scalar β-divergence is

$$d_\beta\left(x||y\right) \triangleq \begin{cases} \frac{1}{\beta(\beta-1)}\left(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}\right) & \beta \in \mathbb{R} \setminus \{0,1\} \\ x(\log x - \log y) + (y-x) & \beta = 1 \text{ (KL-div)} \\ \frac{x}{y} - \log\frac{x}{y} - 1 & \beta = 0 \text{ (IS-div).} \end{cases} \tag{2.14}$$

For $\beta = 2$, $d_\beta\left(x||y\right)$ is equivalent to Euclidean distance between x and y. For $\beta = 1$, $d_\beta\left(x||y\right)$ is the Kullback-Leibler divergence (KL-div) between x and y. For $\beta = 0$, $d_\beta\left(x||y\right)$ is the Itakura-Saito divergence (IS-div) between x and y.

Often, a regularizer on the activations $\mathbf{H}$ will be added to the cost function. A popular regularizer is the ℓ_1 norm, which promotes sparse activations $\mathbf{H}$. In the particular case of

an ℓ_1 regularizer, it is important to normalize the dictionary elements to be unit-norm [34], which must be taken into account during the optimization [87]. Thus, training the dictionary $\mathbf{W}$ consists of solving the optimization problem

$$\underset{\mathbf{W}\in\mathcal{W},\mathbf{H}\geq 0}{\text{minimize}} \quad D_\beta(\mathbf{X}||\mathbf{W}\mathbf{H}) + \lambda_1\|\mathbf{H}\|_1 \tag{2.15}$$

where the set $\mathcal{W}$ consists of all elementwise nonnegative matrices that have unit-norm columns.

Fevotte et al. [42] showed how NMF of a spectrogram using the Itakura-Saito divergence cost function ($\beta = 0$) corresponds to a generalization of the conventional statistical model described in section 2.3.1 where each time-frequency bin is a sum of independent, zero-mean, circular, complex-valued Gaussians. In this case, the NMF dictionary $\mathbf{W} \in \mathbb{R}_+^{F\times N}$ with unit-norm columns corresponds to unit-power PSDs of these different components, while the NMF activation matrix $\mathbf{H} \in \mathbb{R}_+^{N\times T}$ corresponds to time-dependent gains.

Ozerov et al. showed [117] that when the columns of the unnormalized dictionary $\mathbf{W}$ are composed of the variances for the N components in a zero-mean GMM and the activation matrix $\mathbf{H}$ constrained such that only one element is non-zero, IS-NMF is equivalent to a scale Gaussian mixture model (SGMM). When the single non-zero element in each column of $\mathbf{H}$ is further constrained to be equal to 1, IS-NMF is equivalent to a GMM.

The conventional model is a natural fit for multichannel (i.e., multi-microphone) data. Exploiting multiple microphones can greatly improve speech enhancement and recognition performance in the presence of noise, other speakers, and reverberation. One of the first explicit multichannel statistical models was proposed by Attias [6], who used GMM source models combined with either a narrowband or subband-filtering channel model. Attias derived a variational expectation-maximization (EM) algorithm for this model. We will refer to this model as the "multichannel GMM" (MCGMM), and will be unfolded in chapter 6. Later, Ozerov and Vincent [116] combined NMF with a narrowband channel model and derived a variational EM algorithm, which they called, "multichannel NMF." Sawada et al. [128] proposed an alternative form of multichannel NMF where instead of of time-frequency bins

being nonnegative scalars, each time-frequency bin is represented as a positive semidefinite complex-valued matrix. If a multichannel Itakura-Saito divergence is used as the cost function for this model, then each multichannel observation at a particular time-frequency point (composed by stacking the STFTs of all the sources together at each time and frequency) is modeled as a full-covariance (across the microphones), zero-mean complex Gaussian vector, where just as in single-channel IS-NMF, the Gaussian vectors are independent across time and frequency.

2.4 Statistical Modeling with Sparsity

Several of the statistical models that are used and unfolded in this dissertation rely on sparsity, so it is important to understand algorithms that are used to perform inference in these models and recover sparse coefficients.

We will review the problem of nonsequential sparse recovery from a single, static observation vector $\mathbf{x}$. The notation $\mathbb{F}$ represents the set of real numbers $\mathbb{R}$ or of complex numbers $\mathbb{C}$, and $\mathbf{D} \in \mathbb{F}^{N \times N}$ is a dictionary whose columns correspond to basis vectors. We make noisy observations of a signal $\mathbf{s} = \mathbf{D}\mathbf{h}$ through a measurement matrix $\mathbf{A} \in \mathbb{F}^{M \times N}$:

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \boldsymbol{\epsilon}, \tag{2.16}$$

where $M < N$ for a compressed sensing problem. Sparse recovery solves an optimization problem to find $\hat{\mathbf{h}} \in \mathbb{F}^N$ such that the reconstruction $\mathbf{A}\mathbf{D}\hat{\mathbf{h}}$ is as close as possible to $\mathbf{x}$ in terms of squared error, subject to a ℓ_1 penalty:

$$\min_{\mathbf{h}} \quad \frac{1}{2}\|\mathbf{x} - \mathbf{A}\mathbf{D}\mathbf{h}\|_2^2 + \lambda\|\mathbf{h}\|_1. \tag{2.17}$$

Problem (2.17) is known as basis pursuit denoising (BPDN) [19], which is also equivalent to minimizing the Lagrangian of the least absolute shrinkage and selection operator (LASSO) method for sparse recovery [146]. The ℓ_1-norm regularization on $\mathbf{h}$ promotes sparse coefficients, which explain the signal $\mathbf{s}$ with only a few basis vectors, which are columns of $\mathbf{D}$.

The LASSO corresponds to a probabilistic model where the observations $\mathbf{x}$ consist of a deterministic component $\mathbf{As} = \mathbf{ADh}$ plus zero-mean circular Gaussian noise with covariance $\sigma^2\mathbf{I}$, and each element of $\mathbf{h}$ has a centered circular Laplacian prior with scale β:

$$\begin{aligned} \mathbf{x} \sim& \mathcal{N}(\mathbf{ADh}, \sigma^2\mathbf{I}), \\ h_n \sim& \text{Laplace}(0, \beta) \text{ for } n = 1..N. \end{aligned} \tag{2.18}$$

Minimizing the joint negative log-likelihood of $\mathbf{x}$ and $\mathbf{h}$ under this model is equivalent to solving the problem (2.17) with $\lambda = 2\sigma^2/\beta$. This derivation is reviewed in appendix A.

Many algorithms have been proposed [17, 44, 33, 28] for solving the LASSO problem (2.17). Here we will focus on ISTA [17, 28], which is a proximal gradient method that consists of K iterations of soft-thresholding. The basic ISTA algorithm is described in algorithm 1, where $1/\alpha$ is a step size and $\text{soft}_b(\mathbf{z})$ of a vector $\mathbf{z}$ denotes application of the following soft-thresholding operation with real-valued threshold b to each element z_n of $\mathbf{z}$:

$$\text{soft}_b(z_n) = \frac{z_n}{|z_n|}\max(|z_n| - b, 0). \tag{2.19}$$

More generally, ISTA is an accelerated first-order gradient descent algorithm to solve the problem

$$\underset{\mathbf{h}}{\text{minimize}} \quad f(\mathbf{x}, \mathbf{h}) + \lambda_1 g(\mathbf{h}), \tag{2.20}$$

where f is a smooth function and g is a nonsmooth function [17, 28]. ISTA enjoys a $1/K$ rate of convergence, which improves over the $1/\sqrt{K}$ convergence of simple first-order gradient descent on (2.20) [9], where K is the number of iterations. Appendix A gives a derivation for ISTA. Note that when $\mathbf{h}$ has a nonnegativity constraint, as in problem (2.15), the soft-thresholding operation is one-sided: $\text{soft}_b(z_n | h_n \geq 0) = \max(z_n - b, 0)$, which is a ReLU (2.29).

More efficient variants of ISTA have been proposed that use an adaptive step size $\alpha^{(k)}$ and iteration-dependent sparse regularization weight $\lambda^{(k)}$, such as fast ISTA (FISTA) [9], which combines Nesterov's adaptive step size method [109] with a gradual decrease in $\lambda^{(k)}$, and sparse reconstruction by separable approximation (SparSA) [167], which combines Barzilai-Borwein gradient step size adjustment [8] with a gradual decrease in $\lambda^{(k)}$.

Algorithm 1 Iterative soft-thresholding algorithm (ISTA)

Input: observations $\mathbf{x}$, measurement matrix $\mathbf{A}$, dictionary $\mathbf{D}$, initial coefficients $\mathbf{h}^{(0)}$

1: **for** $k = 1$ to K **do**

2: $\quad \mathbf{z} \leftarrow (\mathbf{I} - \frac{1}{\alpha}\mathbf{D}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{D})\mathbf{h}^{(k-1)} + \frac{1}{\alpha}\mathbf{D}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{x}$

3: $\quad \mathbf{h}^{(k)} \leftarrow \text{soft}_{\lambda/\alpha}(\mathbf{z})$

4: **return** $\mathbf{h}^{(K)}$

For nonstationary signals, we will be particularly interested in the sequential extension of sparse coding, where a sequence of data vectors $\mathbf{x}_t$ with $t = 0..T-1$ are modeled using sparse coefficients $\mathbf{h}_t$ using a dictionary $\mathbf{D}$ and observation matrix $\mathbf{A}$:

$$\mathbf{x}_t = \mathbf{A}\mathbf{D}\mathbf{h}_t + \boldsymbol{\epsilon}_t, \text{ for } t = 0..T-1, \tag{2.21}$$

where $\boldsymbol{\epsilon}_t$ is additive noise. In chapter 6, we will derive a sequential version of ISTA to solve problem (2.21). Other algorithms besides sequential ISTA have been proposed for this sequential sparse coding problem [51, 149, 101, 4, 5]. In particular, Asif and Romberg proposed homotopy algorithms for estimation of linear dynamical systems with sparse innovation noise [4] and for sparse time-varying state coefficients [5]. Garrigues and El Ghaoui [51] proposed a homotopy algorithm for updating the sparse recovery solution given online observations. Malioutov et al. [101] studied compressed sensing from online sequences of observations. Vaswani [149] proposed a Kalman filtering-like algorithm for linear dynamical systems with sparse state coefficients whose support changes slowly over time. Adaptive filtering approaches that assume sparsity of the system response, e.g. [20, 105], solve a similar problem if the time-varying input signal taps are instead the static signal dictionary and the outputs are the noisy observations.

2.5 Deep Neural Networks

A deep neural network (DNN) is composed of differentiable computational layers, where each layer has a "forward pass" and a "backward pass." The forward pass q of a layer performs some computations on the input x and produces an output $y = q(x)$.

Deep network layers also have parameters, also called "weights," which can either be nontrainable or trainable. Nontrainable weights are constant, while trainable weights are optimized with respect to a training loss function during the neural network's training phase. In this dissertation, a combination of layers will be referred to as a "neural network architecture."

2.5.1 Training with backpropagation

Once a network architecture has been decided on, training a deep neural network consists of solving the following optimization problem using the supervised training set $\mathcal{D} = \{x_i, y_i\}_{i=0:I-1}$, which consists of I input examples x_i, each with a corresponding target example y_i:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_{i=0}^{I-1} f(\hat{y}_i, y_i) \\ \text{subject to} \quad & \hat{y}_i = q_\theta(x_i),\ i = 0..I-1, \end{aligned} \tag{2.22}$$

where q_θ is the forward pass of the network and θ are the network's parameters.

Usually, training uses first-order mini-batch stochastic gradient descent as the optimization algorithm. The gradients of the training loss with respect to the trainable weights are computed using the backpropagation gradient that comes from the backward pass of the network's layers. The backward pass of a layer is only used during the DNN's training phase and implements the chain rule that computes the gradient of the layer's output with respect to the layer's parameters and input. Usually, the input to the backward pass is the gradient of the scalar training loss with respect to the layer's output.

Often the training optimization algorithm incorporates various additions to improve con-

vergence such as momentum [141, 80] and gradient regularization [147, 80]. As a notational convenience, the trainable weights of a network will be referred to as θ in this dissertation.

Popular loss functions include mean-squared error (MSE) for regression tasks and cross-entropy (CE) for classification tasks. The MSE function for N-dimensional targets is

$$f_{MSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{N} \sum_{n=0}^{N-1} (\hat{y}_n - y_n)^2. \tag{2.23}$$

To describe the CE function, we need to first discuss the representation of classification targets. In classification with C possible labels, the targets y are usually one-hot C-dimensional indicator vectors, while the final layer in the neural network is a softmax function, which converts real-valued C-length vectors into a discrete probability mass function with C possible outcomes. The softmax considers the real-valued input vector to be log-probabilities of different class labels, and the ith element of its output is given by

$$[\text{softmax}(\mathbf{z})]_i = \frac{e^{z_i}}{\sum_j e^{z_j}}. \tag{2.24}$$

Using this representation, the CE function is given by

$$f_{CE}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{c=0}^{C-1} y_c \log \hat{y}_c, \tag{2.25}$$

where $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$, where $\mathbf{z} \in \mathbb{R}^C$ are the potentially unnormalized estimated log-likelihoods of the C classes.

2.5.2 Feedforward neural networks

Feedforward neural networks (FFNNs) are considered the simplest type of deep neural network. A FFNN has K layers, where each layer is a linear affine transform followed by an elementwise nonlinearity. Mathematically, the forward pass of the kth layer of a FFNN for an input vector $\mathbf{x}$ is

$$\mathbf{q}^{(k)}(\mathbf{x}) = \sigma\big(\mathbf{W}^{(k)}\mathbf{q}^{(k-1)} + \mathbf{b}^{(k)}\big), \tag{2.26}$$

with $\mathbf{q}^{(0)} = \mathbf{x}$. The elementwise nonlinearity σ is often referred to as an "activation" function. Popular activation functions include the logistic, or sigmoid, function given by

$$\text{sigm}(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x}. \tag{2.27}$$

The output of the sigmoid function is bounded between 0 and 1, and is equivalent to the second element of the output of the softmax function (2.24) on a 2-dimensional vector $\mathbf{x}$ representing the log probabilities of a binary random variable b being either 0 or 1. That is, the log probability of $b = 0$ is constrained to be 0: $x_0 = \log p(b = 0) = 0$, and $x_1 = \log p(b = 1)$. Another related activation function is the hyperbolic tangent, which is a similar "squashing" type function to the sigmoid function, except that it ranges between -1 and 1:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2.28}$$

We shall see in section 6.3.2 that the sigmoid and hyberbolic tangent functions arise from mean-field inference in Markov random fields (MRFs), a connection that was originally described by Hershey et al. [65].

Another activation function that perhaps the most popular in recent networks is the rectified linear unit (ReLU) [57]:

$$\text{ReLU}(x) = \max(0, x), \tag{2.29}$$

which was empirically observed to produce sparse intermediate outputs. We shall see later in section 6.3.2 that the ReLU arises from an accelerated gradient descent algorithm for nonnegative sparse coding, which provides a principled explanation for why ReLU outputs tend to be sparse.

2.5.3 Recurrent neural networks

A recurrent neural network computes output sequences $\hat{\mathbf{y}}_{1:T}$ from input sequences of data $\mathbf{x}_{1:T}$ using the following nonlinear model:

$$\mathbf{h}_t = \sigma\left(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{V}\mathbf{x}_t + \mathbf{b}\right) \tag{2.30}$$

$$\hat{\mathbf{y}}_t = \mathbf{Z}\mathbf{h}_t + \mathbf{c}, \tag{2.31}$$

where σ is a nonlinear function such as a sigmoid, tanh, or ReLU function. The vector $\mathbf{b}$ denotes optional parameters of the nonlinearity, such as the ReLU threshold. The parameters of the RNN are trained by minimizing a cost function using backpropagation on a supervised dataset consisting of I pairs of input sequences $\mathbf{x}_{1:T_i,i}$ and targets $\mathbf{y}_i$. The targets $\mathbf{y}_i$ may be sequences of vectors, single vectors, or scalars. Notice that the input sequences can have different lengths, denoted by T_i. To handle different sequence lengths, the RNN is made to have $\max_i\{T_i\}$ time steps and a mask is usually used on the training loss function to only backpropagate the loss on valid time steps. Trainable RNN parameters $\{\mathbf{h}_0, \mathbf{b}, \mathbf{U}, \mathbf{V}, \mathbf{Z}, \mathbf{c}\}$ consist of the initial hidden state $\mathbf{h}_0$, the optional parameters of the nonlinearity $\mathbf{b}$, the recurrence matrix $\mathbf{U}$, the input transform $\mathbf{V}$, and the affine output transform with matrix $\mathbf{Z}$ and vector $\mathbf{c}$.

One problem with RNNs is that they are difficult to train via backpropagation because they suffer from the vanishing or exploding gradients problem [12], which occurs when gradients backpropagated through time diminish or grow without bound because of repeated applications of a nonlinear function with non-unity gain. Using a regularized or structured RNN can alleviate this problem. Examples of such regularization include using long-short term memory (LSTM) units [67] that contain explicit memory, batch normalization [70, 26], initializing the recurrence matrix to identity [85], or using orthogonal or unitary recurrence matrices [2, 164].

RNNs are often stacked into multiple layers to create more expressive networks [129, 36, 119]. To stack RNNs, in layer $k > 1$ the hidden state $\mathbf{h}_t^{(k)}$ is connected to the hidden state $\mathbf{h}_t^{(k-1)}$ in layer $k-1$ by a linear transformation $\mathbf{S}^{(k)}$ that is added to the preactivation of

the nonlinearity, as shown in equation (2.32). The output of the network is taken from the hidden states $\mathbf{h}_{1:T}^{(K)}$ in the last layer, layer K, as in equation (2.33).

$$\mathbf{h}_t^{(k)} = \begin{cases} \sigma\left(\mathbf{U}^{(1)}\mathbf{h}_{t-1}^{(1)} + \mathbf{V}\mathbf{x}_t + \mathbf{b}^{(1)}\right), & k = 1, \\ \sigma\left(\mathbf{U}^{(k)}\mathbf{h}_{t-1}^{(k)} + \mathbf{S}^{(k)}\mathbf{h}_t^{(k-1)} + \mathbf{b}^{(k)}\right), & k = 2..K, \end{cases} \tag{2.32}$$

$$\hat{\mathbf{y}}_t = \mathbf{Z}\mathbf{h}_t^{(K)} + \mathbf{c}. \tag{2.33}$$

A diagram of this stacked RNN is shown in the left panel of figure 2.2.

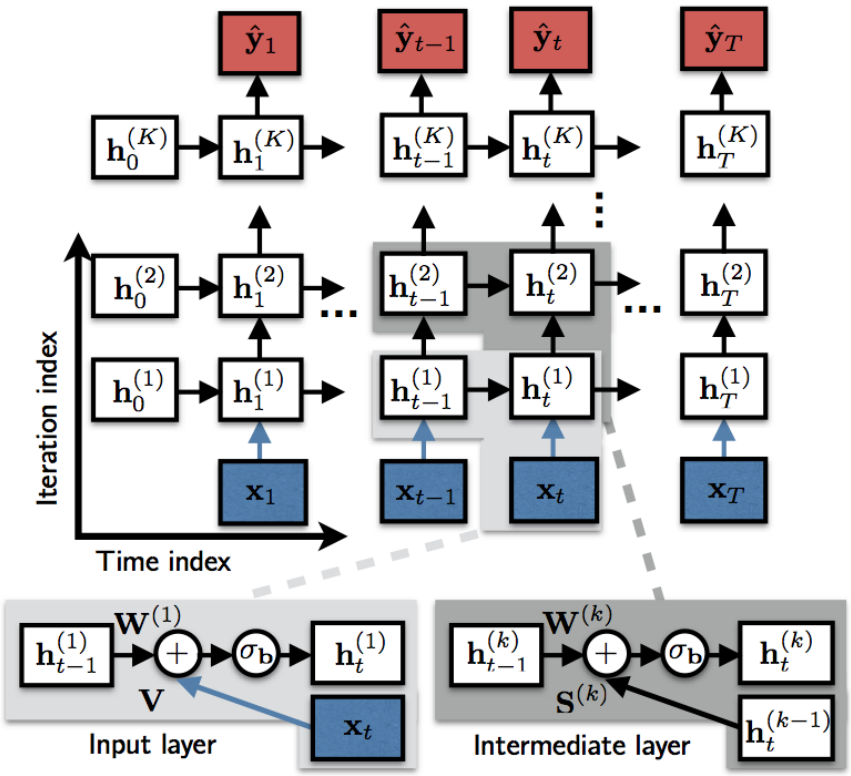

Figure 2.2: Diagram of stacked RNN.

More sophisticated versions of this simple RNN have been proposed. The next three sections cover the most popular RNN architectures.

2.5.4 Long short-term memory (LSTM) networks

The long short-term memory (LSTM) network was originall proposed by Hochreiter and Schmidhuber [67]. The LSTM was proposed to improve the long term memory ability of a RNN using local, or "short-term" computations at each time step. LSTMs were also designed to ameliorate the vanishing/exploding gradient problem that plagues RNN training using backpropagation through time. To address this problem, the LSTM uses a memory cell at each time step, which Hochreiter and Schmidhuber refer to as a "constant error carousel", since the memory cells' goal is to have constant error flow. As long as the gradients are truncated (that is, clipped to a fixed maximum magnitude), the Hochreiter and Schmidhuber prove that the LSTM achieves constant error flow. The state of the memory cell is controlled by "gates", which regulate the input, output, and modification (or 'forgetting") of the memory cell's content. A LSTM produces its hidden states as follows:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t) \tag{2.34}$$

$$\mathbf{C}_t = \mathbf{i}_t \odot \widetilde{\mathbf{C}}_t + \mathbf{f}_t \odot \mathbf{C}_{t-1} \tag{2.35}$$

$$\widetilde{\mathbf{C}}_t = \tanh\big(\mathbf{V}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c\big) \tag{2.36}$$

$$\mathbf{i}_t = \mathrm{sigm}\big(\mathbf{V}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i\big) \tag{2.37}$$

$$\mathbf{f}_t = \mathrm{sigm}\big(\mathbf{V}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f\big) \tag{2.38}$$

$$\mathbf{f}_t = \mathrm{sigm}\big(\mathbf{V}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o\big) \tag{2.39}$$

where for time step t, $\mathbf{C}_t$ is the content of the memory cell, $\mathbf{i}_t$ is the input gate, $\mathbf{f}_t$ is the forget gate, $\widetilde{\mathbf{C}}_t$ is the candidate for the value of the memory cell, and $\mathbf{o}_t$ is the output gate.

2.5.5 Gated recurrent unit (GRU) networks

The GRU architecture was developed by Cho et al. [21], and was inspired by the LSTM described in section 2.5.4. In fact, the GRU was designed to be a simpler replacement for

the LSTM. A GRU produces its hidden states as follows:

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \phi\Big(\mathbf{V}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}\Big) \tag{2.40}$$

$$\mathbf{r}_t = \sigma\Big(\mathbf{V}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r\Big) \tag{2.41}$$

$$\mathbf{z}_t = \sigma\Big(\mathbf{V}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z\Big) \tag{2.42}$$

where ϕ is a elementwise nonlinearity and σ is a sigmoid, or logistic, function. The GRU takes its name from the "gating" by the the vector $\mathbf{r}_t$, which is a "reset" gate, and the vector $\mathbf{z}_t$, which is an "update" gate.

2.5.6 Unitary recurrent neural networks (uRNNs)

The uRNN proposed by Arjovsky et al. [3] consists of the simple RNN described by (2.30) and (2.31), where the inputs $\mathbf{x}_t$ are either real- or complex-valued, the hidden states $\mathbf{h}_t$ of dimension N are complex-valued, and the outputs $\mathbf{y}_t$ are real- or complex-valued. Also, a unitary constraint is placed on the recurrence matrix $\mathbf{U}$. Arjovsky et al. propose using the element-wise "modified ReLU" nonlinearity

$$[\sigma_{\mathbf{b}}(\mathbf{z})]_i = \begin{cases} (|z_i| + b_i)\frac{z_i}{|z_i|}, & \text{if } |z_i| + b_i > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2.43}$$

Note that this non-linearity consists in a soft-thresholding (2.19) of the magnitude using the bias vector $\mathbf{b}$. Hard-thresholding would set the output of σ to z_i if $|z_i| + b_i > 0$.

Arjovsky et al. [3] propose the following parameterization of the unitary matrix $\mathbf{U}$:

$$\mathbf{U}_u(\theta_u) = \mathbf{D}_3\mathbf{R}_2\mathcal{F}^{-1}\mathbf{D}_2\mathbf{P}\mathbf{R}_1\mathcal{F}\mathbf{D}_1, \tag{2.44}$$

where $\mathbf{D}$ are diagonal unitary matrices, $\mathbf{R}$ are Householder reflection matrices [69], $\mathcal{F}$ is a DFT matrix, and $\mathbf{P}$ is a permutation matrix. The resulting matrix $\mathbf{U}_u$ is unitary because all its component matrices are unitary. This decomposition is efficient because diagonal, reflection, and permutation matrices are $\mathcal{O}(N)$ to compute, and DFTs can be computed

efficiently in $\mathcal{O}(N \log N)$ time using the fast Fourier transform (FFT). The parameter vector θ_u consists of $7N$ real-valued parameters: N parameters for each of the 3 diagonal matrices where $D_{i,i} = e^{j\theta_i}$ and $2N$ parameters for each of the 2 Householder reflection matrices, which are real and imaginary values of the complex reflection vectors $\mathbf{u}_i$: $\mathbf{R}_i = \mathbf{I} - 2\frac{\mathbf{u}_i \mathbf{u}_i^{\mathsf{H}}}{\langle \mathbf{u}_i, \mathbf{u}_i \rangle}$.

2.6 Deep Unfolding

Effective algorithms for processing data can be derived by defining a statistical model. Such a model defines a likelihood function and encodes simplifying assumptions and physical domain knowledge about the data generation process. Often something is known about the data of interest, which is often referred to as *domain knowledge*. Over decades and even centuries, much effort has been made to understand all types of physical processes, and statistical models are an effective way to express and exploit this domain knowledge. As described in section 2.2, algorithms that maximize the likelihood function with respect to the statistical model's parameters and/or the settings of hidden random variables in the model can then be derived.

Designing statistical models and deriving efficient and optimal algorithms for them can be very labor-intensive. Often, the exact and optimal inference algorithm is computationally intractable, and so approximate inference algorithms need to be used. Deep learning attempts to reduce this labor by combining general purpose algorithmic components (layers) and directly training the parameters of these components on a large amount of data to minimize some loss function of interest. The hope is that by using enough training data, the general-purpose model will learn weights for its layers such that the algorithm given by the composition of these layers yields useful outputs. This training procedure shifts the labor of designing a model and deriving algorithms to the work of gathering a large amount of data and using much more computation. However, the choice of deep network layers is not trivial, and can have a large impact on the trained network's performance. The best-performing architectures have been established through a long process of trial-and-error, which started with stacks of restricted Boltzmann machines [66]. For example, more activation functions,

such as rectified linear units [57] were formulated, as well as regularization techniques such as dropout [138] and batch normalization [70].

Recently, the deep unfolding framework was proposed by Hershey et al. [65] to combine the advantages of model-based methods and deep networks by treating the inference iterations of a model-based approach as layers in a deep network. Rather than optimizing the original model parameters, these parameters can be untied across layers to create a more powerful and novel deep architecture that retains some of the model-based constraints, yet allows a broader class of inference algorithms. The resulting network can be trained discriminatively with backpropagation as described in section 2.5.1 to perform accurate inference within a fixed network size.

This dissertation takes this perspective that every algorithm for processing data, whether it is an inference algorithm for a statistical model or a deep network, consists of a sequence of computational operations. By viewing both conventional statistical inference algorithms and recently-successful deep networks as being equivalent, this dissertation will argue that only a limited subset of the entire class of computational networks has been utilized and explored through trial and error. Adopting a general perspective provides connections between inference in statistical models and deep networks and provides a principled means of creating and explaining computational network architectures. In this dissertation, we will explore several instances of deep unfolding specifically for processing nonstationary signals.

2.7 Summary, Problem Statement, and Contributions

Now, let us summarize the main background concepts from this chapter and formulate a problem statement. Section 2.1 described the STFT, which will be the primary representation of nonstationary signals used in this dissertation. Section 2.2 described the general framework of statistical modeling, estimation of model parameters, and inference of hidden variables. Since the idea of statistical modeling is very general, sections 2.3 and 2.4 cover specific statistical models relevant to the contributions presented in this dissertation. First, section 2.1 describes conventional models the STFT, including the unsupervised circular

complex-valued Gaussian model and the family of supervised nonnegative matrix factorization models. Next section 2.5 gave an overview of deep learning, including training and the conventional layers used in deep networks. Finally, section 2.6 provided an overview of deep unfolding, which considers both statistical model inference algorithms and deep networks to be equivalent in the sense that they are both computational graphs.

The main question addressed in this dissertation is this: how can fundamental statistical tasks such as enhancement, detection, and classification be improved for nonstationary signals? To answer this question, this dissertation provides three main contributions:

1. Statistical models of nonstationary signals are improved using noncircular distributions of the complex-valued STFT. These models improve performance on detection of nonstationary audio signals. Specifically, practical estimators of noncircularity from the STFT of a single instance of a nonstationary signal are provided, and these estimators are shown to improve performance on realistic detection tasks. Chapters 3 and 4 describe this contribution.

2. A recently-proposed recurrent neural network, the unitary RNN (uRNN), is examined and improved. Specifically, the representational capacity of the set of unitary matrices that the uRNN relies on is quantified. As a result, the conventional unitary parameterization is found to have restricted capacity, and a method of optimization is proposed that ensures the full set of unitary matrices is used. This full-capacity uRNN achieves superior performance compared to conventional RNNs on a variety of benchmark tasks for nonstationary input sequences. This contribution is covered by chapter 5.

3. Existing statistical models of the STFT are unfolded to deep networks that are used for enhancement and classification of nonstationary audio signals. The unfolded networks have distinct advantages compared to both existing statistical models and deep networks. Chapters 6 and 7 describe this contribution.

Chapter 3

IMPROVING STATISTICAL MODELS OF NONSTATIONARY SIGNALS WITH NONCIRCULAR DISTRIBUTIONS

This chapter will describe the first main contribution of this dissertation, which is improving the conventional statistical model for the STFT of nonstationary signals. In the conventional model, each complex-valued time-frequency bin of the STFT is modeled as a scalar, circular, complex-valued Gaussian. This conventional model can be improved by relaxing the conventional circular assumption and using noncircular Gaussians instead. Using noncircular distributions exploits additional information that arises from certain classes of nonstationary signals exhibiting significant noncircularity in the STFT domain. By exploiting this additional information, the performance of statistical signal processing algorithms can be improved. In chapter 4, these noncircular models are applied to the task of detecting speech and transient acoustic signals.

An outline of this chapter is as follows. First, distributions of circular and noncircular complex-valued data will be reviewed. Then, specific examples of nonstationary signals that exhibit noncircularity in the STFT domain will be presented. Finally, maximum-likelihood estimators will then be given for the parameters of these noncircular distributions, as well as practical application of these estimators to the STFT.

3.1 Random Complex-Valued Data

This section describes the second-order statistics of complex-valued Gaussian random vectors. Recall that the conventional Gaussian model of the STFT uses a scalar Gaussian to model each time-frequency bin. Models for multi-channel nonstationary signals will also be considered in this dissertation, which model each time-frequency bin using complex-valued

Gaussian random vectors, where the dimension of the vector is equal to the number of channels (e.g., microphones). The notation and presentation follows Schreier and Scharf [130].

A complex-valued Gaussian random vector $\mathbf{x} \in \mathbb{C}^D$, $\mathbf{x} \sim \mathcal{CN}\left(\boldsymbol{\mu}_x, \widetilde{\mathbf{R}}_{xx}, \mathbf{R}_{xx}\right)$, has pdf

$$p(\mathbf{x}; \boldsymbol{\mu}_x, \widetilde{\mathbf{R}}_{xx}, \mathbf{R}_{xx}) = \frac{1}{\pi^D \det^{1/2} \underline{\mathbf{R}}_{xx}} \exp\left\{-\frac{1}{2}(\underline{\mathbf{x}} - \underline{\boldsymbol{\mu}}_x)^{\mathsf{H}} \underline{\mathbf{R}}_x^{-1} (\underline{\mathbf{x}} - \underline{\boldsymbol{\mu}}_x)\right\} \tag{3.1}$$

where $\boldsymbol{\mu}_x$ is the complex-valued mean, $\underline{\boldsymbol{\mu}}_x = [\boldsymbol{\mu}_x; \boldsymbol{\mu}_x^*]$ is the augmented mean that stacks $\boldsymbol{\mu}_x$ on top of its elementwise conjugate $\boldsymbol{\mu}_x^*$, $\widetilde{\mathbf{R}}_{xx} := E[\mathbf{x}\mathbf{x}^{\mathsf{T}}] \in \mathbb{C}^{D \times D}$ is the complementary covariance, and $\mathbf{R}_{xx} := E[\mathbf{x}\mathbf{x}^{\mathsf{H}}] \in \mathbb{C}^{d \times d}$, $\mathbf{R}_{xx} \succeq 0$, is the Hermitian covariance. The matrix $\underline{\mathbf{R}}_{xx} \in \mathbb{C}^{2d \times 2d}$ is the augmented covariance matrix, given by

$$\underline{\mathbf{R}}_{xx} = \begin{bmatrix} \mathbf{R}_{xx} & \widetilde{\mathbf{R}}_{xx} \\ \widetilde{\mathbf{R}}_{xx}^* & \mathbf{R}_{xx}^* \end{bmatrix}. \tag{3.2}$$

The augmented covariance can also be seen as the covariance of the augmented vector $\underline{\mathbf{x}} := [\mathbf{x}; \mathbf{x}^*]$: $\underline{\mathbf{R}}_{xx} = E\left[\underline{\mathbf{x}}\underline{\mathbf{x}}^{\mathsf{H}}\right]$. When the complementary covariance $\widetilde{\mathbf{R}}_{xx}$ is 0, $\mathbf{x}$ is said to be *second-order circular*, or *proper*. When $\widetilde{\mathbf{R}}_{xx}$ is not zero, $\mathbf{x}$ is said to be *second-order noncircular*, or *improper*. The terms second-order noncircular and improper are used interchangeably. Since this dissertation focuses on Gaussian distributions that do not exhibit higher-order noncircularity, the term noncircular will henceforth be synonomous with second-order noncircular and improper. Figure 3.1 shows a comparison between a realization of a circular scalar Gaussian (left panel) and a noncircular scalar Gaussian (right panel) in the complex plane.

The complex augmented form of $\mathbf{x}$ is unitarily equivalent within a factor of 2 to the real-composite form, given by stacking the real part of $\mathbf{x}$ on top of the imaginary part of $\mathbf{x}$ [130]. That is, the augmented vector $\underline{\mathbf{x}} = [\mathbf{x}; \mathbf{x}^*]$ can be related to the real composite vector $[\mathrm{Re}\{\mathbf{x}\}; \mathrm{Im}\{\mathbf{x}\}]$ using the unitary transform $\mathbf{T}$:

$$\begin{bmatrix} \mathrm{Re}\{\mathbf{x}\} \\ \mathrm{Im}\{\mathbf{x}\} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ -j\mathbf{I} & j\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}^* \end{bmatrix} = \frac{1}{2}\mathbf{T}\underline{\mathbf{x}}. \tag{3.3}$$

Using complex augmented form instead of the real composite form can lead to algebraic simplifications in derivations. Both forms are entirely equivalent representations.

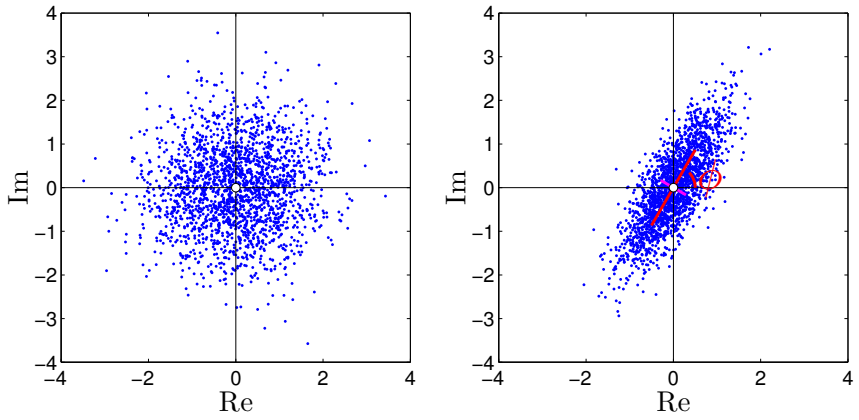

Figure 3.1: Realizations in the complex plane of a circular Gaussian (left) and a noncircular Gaussian (right). The angle ϕ of the noncircular Gaussian in the right panel is the angle of the complementary variance.

For a scalar complex Gaussian random variable x with scalar variances R_{xx} and $\widetilde{R}_{xx}$, an important statistic is $k_x = \frac{\widetilde{R}_{xx}}{R_{xx}}$, which is called the *circularity coefficient* (CC) of x. Using the CC, the distribution (3.1) can be written

$$p(x; \mu, \widetilde{R}_{xx}, R_{xx}) = \frac{1}{\pi R_{xx}\sqrt{1-|k_x|^2}} \exp\left(-\frac{|x|^2 - \mathrm{Re}\left\{k_x^* x\right\}}{R_{xx}\left(1-|k_x|^2\right)}\right). \tag{3.4}$$

When L samples of x are available, the pdf is

$$p(x_{1:L}; \mu, \widetilde{R}_{xx}, R_{xx}) = \left(\pi R_{xx}\sqrt{1-|k_x|^2}\right)^{-L} \exp\left(-\frac{\sum_\ell \left(|x_\ell|^2 - \mathrm{Re}\left\{k_x^* \sum_\ell x_\ell^2\right\}\right)}{R_{xx}\left(1-|k_x|^2\right)}\right). \tag{3.5}$$

Note that $\frac{R_{xx}}{R_{xx}^2 - |\widetilde{R}_{xx}|^2} = \frac{R_{xx}}{R_{xx}^2(1-|k_x|^2)} = \frac{1}{R_{xx}(1-|k_x|^2)}$, which allows the pdfs to be written in terms of the Hermitian variance R_{xx} and magnitude of the complementary variance $|\widetilde{R}_{xx}|$, instead of in terms of the Hermitian variance R_{xx} and magnitude of the circularity coefficient $|k_x|$. The magnitude of the circularity coefficient $|k_x|$ ranges from 0 to 1. When $|k_x| = 0$, the random variable x is said to be circular. When $|k_x| = 1$, x is maximally noncircular, or rectilinear, and samples of x fall along a line in the complex plane.

An alternative way to write the complex normal pdf of a scalar random variable x is as a joint distribution over the magnitude r and phase θ of x [130, (2.71)]:

$$p(r,\theta;R_{xx},\widetilde{R}_{xx}) = \frac{r}{\pi R_{xx}\sqrt{1-|k_x|^2}} \exp\left\{-\frac{r^2}{R_{xx}\left(1-|k_x|^2\right)}+\frac{r^2|k_x|}{R_{xx}\left(1-|k_x|^2\right)}\cos(2\theta-\angle\widetilde{R}_{xx})\right\}. \tag{3.6}$$

If we marginalize over $\theta = \angle x$, using the fact that $\frac{1}{\pi}\int_0^\pi e^{z\cos(\theta+\psi)}d\theta = I_0(z)$ for any ψ, where $I_0(z)$ is a modified Bessel function of the first kind with order 0, the result is

$$p(r;R_{xx},|\widetilde{R}_x|) = \frac{2r}{R_{xx}\sqrt{1-|k_x|^2}} \exp\left\{-\frac{r^2}{R_{xx}\left(1-|k_x|^2\right)}\right\} I_0\left(\frac{r^2|k_x|}{R_{xx}\left(1-|k_x|^2\right)}\right), \tag{3.7}$$

which is no longer dependent on the phase θ of x nor on the phase ϕ of $\widetilde{R}_x$.

The conditional distribution of $\phi = 2\theta = 2\angle x$ given the magnitude $r = |x|$, $p(\phi|r;R_{xx},\widetilde{R}_x)$, is a von Mises distribution with mean direction $\tau = \angle\widetilde{R}_x$ and concentration parameter $\nu = \frac{r^2|k_x|}{R_{xx}(1-|k_x|^2)}$. The pdf of a von Mises random variable ϕ with mean direction τ and concentration parameter ν is

$$p(\phi;\tau,\nu) = \frac{1}{2\pi I_0(\nu)} \exp\left\{\nu\cos(\phi-\tau)\right\}. \tag{3.8}$$

The variance of ϕ is $1-\frac{I_1(\nu)}{I_0(\nu)}$, where I_1 is a Bessel function of the first kind with order 1. Note that as $|k_x| \to 1$, $\nu \to \infty$, which means that $\mathrm{var}\{\phi\} \to 0$.

When the complex data is a random vector, $\mathbf{x} \in \mathbb{C}^N$, the *circularity spectrum* (CS) is the extension of the CC from scalar random variables to random vectors. Like the circularity coefficient for scalar random variables, the CS characterizes the noncircularity of the random vector $\mathbf{x}$. The CS is given by the singular values $\{k_i\}_{i\in[1,N]}$ of the coherence matrix $\mathbf{C_{xx}}$, given by [130, Section 3.2]

$$\mathbf{C_{xx}} = \mathbf{R_{xx}}^{-1/2}\widetilde{\mathbf{R}}_{\mathbf{xx}}\mathbf{R_{xx}}^{-T/2} = \mathbf{UKV}^{\mathsf{H}}, \tag{3.9}$$

where $\mathbf{K} = \operatorname{diag}\mathbf{k}$, $\mathbf{U}$ are the left singular vectors of $\mathbf{C}_{xx}$, $\mathbf{V}$ the right singular vectors of $\mathbf{C}_{xx}$, $\widetilde{\mathbf{R}}_{\mathbf{xx}} = E\left[\mathbf{xx}^{\mathsf{T}}\right]$, and $\mathbf{R_{xx}} = E\left[\mathbf{xx}^{\mathsf{H}}\right]$. The CS has two compelling properties: it is invariant to nonsingular linear transformations of $\mathbf{x}$, and its elements are the magnitude

of the circularity coefficients of the maximally-decorrelated components of $\mathbf{x}$. That is, if $\mathbf{x} = \mathbf{G}\mathbf{y}$, where $\mathbf{G} \in \mathbb{C}^{N\times N}$ is an nonsingular linear transform and the elements of $\mathbf{y}$ are uncorrelated, then $k_i = |\mathrm{CC}\{y_i\}| = \left|\widetilde{R}_{y_iy_i}\right| / R_{y_iy_i}$. Because of this decorrelation property, the CS is an essential component in complex-valued blind source separation [39].

3.2 Theoretical STFT Noncircularity for Nonstationary Signals

This section presents the theoretical spectral noncircularity and spectral correlations of several models of natural signals. To study the theoretical spectral noncircularity of random signals, it is important to first understand the Cramér-Loève spectral representation [96] of a random process $x(t)$, given by

$$x(t) = \int_{-\infty}^{\infty} e^{j2\pi ft} dX(\xi), \tag{3.10}$$

where $dX(\xi)$ is a complex-valued spectral increment process and ξ is continuous frequency. Since $dX(\xi)$ is complex valued, it requires two second-order statistics. Thus, these increment processes have bifrequency Hermitian spectral correlation (HSC) and complementary spectral correlation (CSC) which are, respectively,

$$S_{xx}^{(bf)}(\xi_1, \xi_2) d\xi_1 d\xi_2 := E[dX(\xi_1) dX^*(\xi_2)], \tag{3.11}$$

$$\widetilde{S}_{xx}^{(bf)}(\xi_1, \xi_2) d\xi_1 d\xi_2 := E[dX(\xi_1) dX(\xi_2)]. \tag{3.12}$$

An alternative and sufficient representation to the bi-frequency SCs presented above are global-local SCs, with global frequency ξ and local frequency ν:

$$S_{xx}^{(gl)}(\xi, \nu) d\xi d\nu \triangleq E[dX(\xi) dX^*(\xi - \nu)] \tag{3.13}$$

$$\widetilde{S}_{xx}^{(gl)}(\xi, \nu) d\xi d\nu \triangleq E[dX(\xi) dX(\xi - \nu)]. \tag{3.14}$$

Note that global-local SC can be transformed to bifrequency SC by using the transformation $\xi_1 = \xi$ and $\xi_2 = \xi - \nu$, which is a skewing of the (ξ_1, ξ_2) plane to the (ξ, ν) plane. In this section, expressions for cross-frequency correlations will be given in the most convenient representation.

3.2.1 Wide-sense stationary random processes

A wide-sense stationary (WSS) random process has orthogonal increments [121], which means that a WSS process exhibits no spectral correlation between different frequencies. That is,

$$S_{xx}^{(bf)}(\xi_1, \xi_2) = 0, \text{ for } \xi_1 \neq \xi_2 \tag{3.15}$$

Furthermore, a WSS process's spectral increments are circular [130], which means that the process has zero complementary spectral correlation everywhere. That is,

$$\widetilde{S}_{x}^{(bf)}(\xi_1, \xi_2) = 0, \text{ for all } \xi_1, \xi_2. \tag{3.16}$$

Thus, a WSS process is completely characterized by its power spectral density (PSD) along the "stationary manifold," which consists of all pairs $\{(\xi_1, \xi_2) : \xi_1 = \xi_2\}$ in the bifrequency representation, or all pairs $\{(\xi, \nu) : \nu = 0 \ \forall \xi\}$ in the global-local representation [131]. The PSD is thus

$$S_x(\xi) \stackrel{\Delta}{=} S_{xx}^{(bf)}(\xi_1 = \xi, \xi_2 = \xi). \tag{3.17}$$

Figure 3.2 illustrates these concepts. The theoretical bifrequency HSC (left panel), global-local frequency HSC (center panel), and conventional PSD (right panel) are plotted for an example WSS second-order autoregressive process. Note that for a WSS process, the HSC is zero off of the stationary manifold.

3.2.2 Deterministic signal in WSS noise

Now let us consider the case of a deterministic signal $p(t)$ in the presence of additive WSS noise:

$$x(t) = p(t - t_0) + v(t), \tag{3.18}$$

where t_0 is a deterministic delay. In this case, the spectral increments are

$$dX(\xi) = P(\xi)e^{-\mathrm{j}2\pi\xi t_0}d\xi + dV(\xi), \tag{3.19}$$

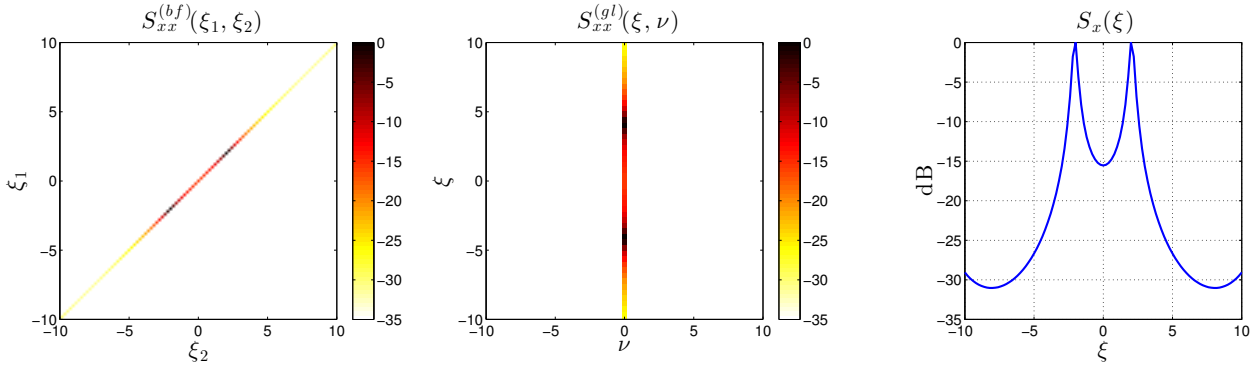

Figure 3.2: Theoretical bifrequency Hermitian spectral correlation (HSC) and power spectral density (PSD) for a real-valued wide-sense stationary (WSS) random process. Left panel: bifrequency HSC; center panel: global-local frequency HSC; right panel: conventional PSD $S_x(\xi)$, equivalent to $S_{xx}^{(bf)}(\xi,\xi)$. Reprinted from [159], ©2016 IEEE.

and the CSC and HSC along the stationary manifold are, respectively,

$$\widetilde{S}_x(\xi) = \widetilde{S}_{xx}^{(bf)}(\xi,\xi) = P^2(\xi)\exp(-\mathrm{j}4\pi\xi t_0), \tag{3.20}$$

$$S_x(\xi) = S_{xx}^{(bf)}(\xi,\xi) = |P(\xi)|^2 + S_v(\xi). \tag{3.21}$$

Thus, along the stationary manifold, $x(t)$ exhibits spectral noncircularity at all frequencies where $P(\xi)$ has non-zero power, with degree of noncircularity

$$|k_x(\xi)| = \frac{|\widetilde{S}_{xx}(\xi)|}{S_{xx}(\xi)} = \frac{|P(\xi)|^2}{|P(\xi)|^2 + S_v(\xi)} = \frac{\mathrm{SNR}(\xi)}{\mathrm{SNR}(\xi)+1}, \tag{3.22}$$

where $\mathrm{SNR}(\xi) := \frac{|P(\xi)|^2}{S_v(\xi)}$ is the frequency-dependent power ratio between the deterministic signal and the random noise.

Off the stationary manifold where $\xi_1 \neq \xi_2$, the HSC is only nonzero at all combinations of frequencies where $P(\xi_1)$ and $P(\xi_2)$ are nonzero. The HSC and CSC are

$$S_{xx}^{(bf)}(\xi_1,\xi_2) = P(\xi_1)P^*(\xi_2), \tag{3.23}$$

$$\widetilde{S}_{xx}^{(bf)}(\xi_1,\xi_2) = P(\xi_1)P(\xi_2). \tag{3.24}$$

3.2.3 Modulated WSS noise

Consider a real-valued signal of the form

$$x(t) = m(t)w(t), \tag{3.25}$$

where $m(t)$ is a deterministic sum of sinusoids:

$$\begin{aligned} m(t) &= \sum_{k=1}^{K} A_k \cos(2\pi\xi_k t + \theta_k) \\ &= \sum_{k=-K}^{K} C_k \exp(\mathrm{j}2\pi\xi_k t) + v(t), \end{aligned} \tag{3.26}$$

where A_k are deterministic amplitudes, θ_k are deterministic phases on $[-\pi, \pi]$, and $v(t)$ is WSS additive noise with PSD $S_v(\xi)$. The sinusoidal parameters can be equivalently expressed [121] using complex-valued quantities $C_k = \frac{A_k}{2} e^{j\theta_k}$ with $C_0 = 0$ and $C_{-k} = C_k^*$. Also, $\xi_{-k} = -\xi_k$ and $\xi_0 = 0$. If the ξ_k have a least common multiple ξ_0, this is a cyclostationary signal with cycle period $T_0 = 1/\xi_0$; otherwise, it is an almost-periodic cyclostationary signal [55, 48, 49, 108].

The simplest case to consider is when $m(t)$ has one harmonic ($K = 1$) with $A_1 = A$. In this case, the spectrum of $m(t)$ with $C = C_1$ is

$$M(\xi) = C^*\delta(\xi + \xi_1) + C\delta(\xi - \xi_1). \tag{3.27}$$

Using the Fourier transform identity $m(t)w(t) \Leftrightarrow M(\xi) * W(\xi)$, where $*$ is convolution, and the identity $W(= \xi) * \delta(\xi - \xi_1) = W(\xi - \xi_1)$, the spectral increments of $x(t)$ are

$$dX(\xi) = C^* dW(\xi + \xi_1) + C dW(\xi - \xi_1). \tag{3.28}$$

Since $w(t)$ is real-valued, then $dW^*(\xi) = dW(-\xi)$, and the HSC of $x(t)$ is

$$\begin{aligned} S_{xx}^{(gl)}(\xi, \nu) d\xi d\nu &= E\left\{ \Big[dX(\xi) dX^*(\xi - \nu) \Big] \right\} \\ &= E\Big\{ \Big[C^* dW(\xi + \xi_1) + C dW(\xi - \xi_1) \Big] ... \\ &\qquad ... \Big[C dW^*(\xi + \xi_1 - \nu) + C^* dW^*(\xi - \xi_1 - \nu) \Big] \Big\}. \end{aligned} \tag{3.29}$$

The terms in the HSC will only be non-zero when the arguments of $dW(\cdot)$ and $dW^*(\cdot)$ are equal. Thus, along the stationary manifold ($\nu = 0$), the HSC is

$$S_{xx}^{(gl)}(\xi, \nu = 0) = |C|^2 \Big[S_w(\xi + \xi_1) + S_w(\xi - \xi_1) \Big]. \tag{3.30}$$

Using the fact that $x(t)$ is real-valued, $dX(\xi) = dX^*(-\xi)$, and the CSC is

$$\begin{aligned} \widetilde{S}_{xx}^{(gl)}(\xi, \nu) df d\nu &= E\left\{ \Big[dX(\xi) dX(\xi - \nu) \Big] \right\} \\ &= E\left\{ \Big[dX(\xi) dX^*(-\xi + \nu) \Big] \right\} \\ &= E\Big\{ \Big[C^* dW(\xi + \xi_1) + C dW(\xi - \xi_1) \Big] ... \\ &\quad ... \Big[C dW^*(-\xi - \xi_1 + \nu) + C^* dW^*(-\xi + \xi_1 + \nu) \Big] \Big\}. \end{aligned} \tag{3.31}$$

Now, we have that along the stationary manifold $\nu = 0$, the CSC is

$$\widetilde{S}_{xx}^{(gl)}(\xi, \nu = 0) = \begin{cases} 2\Re\{C^2\} S_w(\xi_1) & \text{if } \xi = 0 \\ |C|^2 S_w(0) & \text{if } \xi = \pm \xi_1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.32}$$

3.3 Estimating STFT Noncircularity

Now that we have seen that various classes of nonstationary signals can exhibit noncircularity in the STFT, this section proposes methods to estimate spectral correlations and spectral noncircularity in practice from the STFT.

Given L samples of a scalar complex Gaussian random variable x, the maximum likelihood estimate for the circularity coefficient k_x is

$$\hat{k}_x = \frac{\hat{\widetilde{R}}_{xx}}{\hat{R}_{xx}} = \frac{\frac{1}{L} \sum_{\ell=0}^{L-1} x_\ell x_\ell}{\frac{1}{L} \sum_{\ell=0}^{L-1} |x_\ell|^2}. \tag{3.33}$$

To determine statistically significant noncircularity, the circularity coefficient plays an important role. It can be shown that the degree of impropriety (DOI) $|\hat{k}_x|^2$ is a generalized likelihood ratio test (GLRT) statistic for impropriety [130, Result 3.8].

Now assume that M samples, x_m, $m = 0..M-1$ of a random process $x(t)$ are measured. In this case, applying an analysis window $h \in \mathbb{R}^M$ to x, computing the discrete spectrum X_f using a M-length FFT, and taking the magnitude-squared corresponds to a direct spectral estimate $\hat{S}_x^{(D)}(\xi_f)$ [121], which is equivalent to an estimate of the Hermitian spectral correlation $S_{xx}^{(bf)}(\xi_f, \xi_f)$ along the stationary manifold at the discrete Fourier frequencies $\xi_f = \frac{f}{M} \cdot \xi_s$, where ξ_s is the sampling frequency.

Unfortunately, estimating the complementary spectral correlation is not as simple as just squaring X_f (i.e., multiplying X_f by itself), since this would produce an estimator that is maximally noncircular at every frequency. That is, if $L = 1$,

$$|\hat{k_x}| = \frac{|\sum_{\ell=0}^{0} (X_f^\ell)^2|}{\sum_{\ell=0}^{0} |X_f^\ell|^2} = 1. \tag{3.34}$$

Thus, an effective estimator requires some amount of averaging over multiple realizations (i.e., L must be greater than 1).

Since only one realization of the random process samples $x_{0:M-1}$ is available, ensemble averaging is impossible. Instead, multiple realizations can be acquired in one of two ways. First, x can be segmented into L shorter chunks and the estimator can average in time over the spectra of the chunks. To ensure the successive chunks have the same phase reference, the deterministic phase progressions caused by the STFT window hops needs to be removed by multiplying X_f^ℓ by $\exp(\mathrm{j}2\pi\xi_f \ell M_{hop})$. Averaging the spectra of time-domain chunks is akin to Welch's method [155] for spectral estimation. Note that by averaging over time, shorter windows are used, which results in reduced spectral resolution.

Alternatively, by using L mutually orthogonal multitapers $\mathbf{h}^\ell \in \mathbb{R}^M$ such that $\langle \mathbf{h}^{\ell_1}, \mathbf{h}^{\ell_2} \rangle = 0$ for $\ell_1 \neq \ell_2$, we can acquire L independent observations of the spectra: $X_f^\ell \triangleq \mathrm{DFT}_f \left\{ \mathbf{h}^\ell \odot \mathbf{x} \right\}$, where f is discrete frequency index and ℓ is taper index. A multitaper estimator (a method originally proposed by Thomson [145]) averages across frequency, with the bandwidth increasing as the number of tapers increases. Similar to Welch's method, averaging across frequency reduces spectral resolution. The standard choice for the multitapers are the discrete prolate spheroidal sequences (DPSS), which maximize the spectral concentration of the

windows $\mathbf{h}_\ell$ for a given bandwidth W [145].

Using either Welch's or Thomson's multitaper method, the estimators for HSC and CSC at frequencies ξ_{f_1}, ξ_{f_2} are

$$\hat{S}_{xx}^{(bf)}(\xi_{f_1}, \xi_{f_2}) = \frac{1}{L} \sum_{\ell=0}^{L-1} c_\ell X_{f_1}^\ell (X_{f_2}^\ell)^*, \tag{3.35}$$

$$\hat{\widetilde{S}}_{xx}^{(bf)}(\xi_{f_1}, \xi_{f_2}) = \frac{1}{L} \sum_{\ell=0}^{L-1} c_\ell X_{f_1}^\ell X_{f_2}^\ell \tag{3.36}$$

where $c_\ell = 1$ for all ℓ when using Welch's method and when using the multitaper method, c_ℓ are weights corresponding to the eigenvalues of the tapers. When $f_1 = f_2 = f$, by the triangle inequality these estimators satisfy the property $\left|\hat{\widetilde{S}}_{xx}^{(bf)}(\xi_f, \xi_f)\right| \leq \hat{S}_{xx}^{(bf)}(\xi_f, \xi_f)$ for $f = 0..F-1$. A version of the multitaper CSC estimator along the stationary manifold was originally proposed by Clark et al. [24].

Using (3.35) and (3.36), the spectral noncircularity estimator at the fth frequency ξ_f is

$$\hat{k}_x(\xi_f) = \frac{\hat{\widetilde{S}}_{xx}^{(bf)}(\xi_f, \xi_f)}{\hat{S}_{xx}^{(bf)}(\xi_f, \xi_f)}. \tag{3.37}$$

If the samples $X_f^\ell / \sqrt{c^\ell}$ are assumed to be distributed as i.i.d. complex-valued Gaussians, Delmas et al. [31, Remark 5] provide an approximate finite-sample distribution for the estimator of the degree of noncircularity $\hat{k}$ in (3.37). According to this approximation, the distribution of $|\hat{k}|$ converges as $L \to \infty$ to a Rayleigh distribution with scale $L^{-1/2}$ when $|k| = 0$ and to a normal distribution with mean $|k|$ and variance $\sigma_{|k|}^2 = L^{-1/2}(1 - |k|^2)^2$ when $|k| > 0$. In practice, this approximation seems to hold for L as low as 10 samples [23, Appendix C]. Using this approximate finite-sample distribution, we can define a threshold T for deciding whether a frequency bin is noncircular for a given probability of false alarm P_{FA}.

3.3.1 Synthetic data

Two simple test signals to verify these estimators are a sinusoid at frequency $\xi_0 = 125$ Hz of duration 1 second and a Kronecker delta function. Both signals are embedded in

white noise at a SNR of 10dB. The sampling frequency is 16kHz and frames of length 64 milliseconds are used for both the Welch's and multitaper estimators. Statistics are computed in 8 millisecond increments using $L = 16$ frames at a time. The Welch's estimator uses a Hamming window with a hop of 32 milliseconds, and the multitaper estimator uses 16 DPSS tapers with duration equal to the total duration of the Welch's averaging region, which is $L \cdot (\text{hop}) = 16(32\text{ms}) = 512\text{ms}$. Figures 3.3 and 3.4 shows the results.

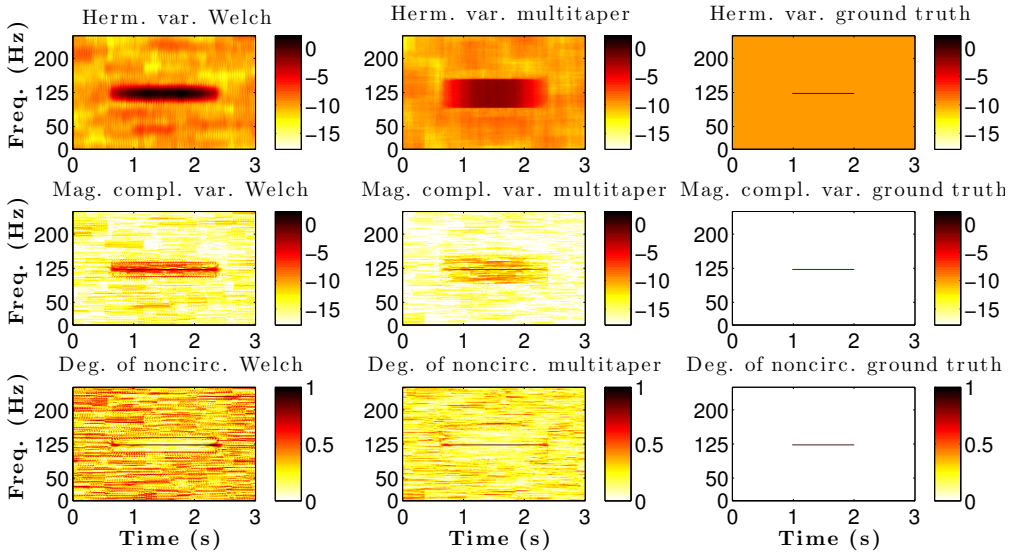

Figure 3.3: Outputs of Welch and multitaper estimators of spectral Hermitian variance (top row), complementary variance (middle row), and degree of noncircularity (bottom row) for a sinusoid in 10dB SNR white noise. The ideal ground truth is given in the right columns of the panels. The estimator uses a bandwidth of $2/(64\text{ms}) = 31.25\text{Hz}$. Reprinted from [159], ©2016 IEEE.

For reference, we give ground truth time-frequency plots that are the output of an ideal estimator using our results from §III.B. Aside from smoothing, both estimators appear to work relatively well, in the sense that the estimates appear close to the ground truth.

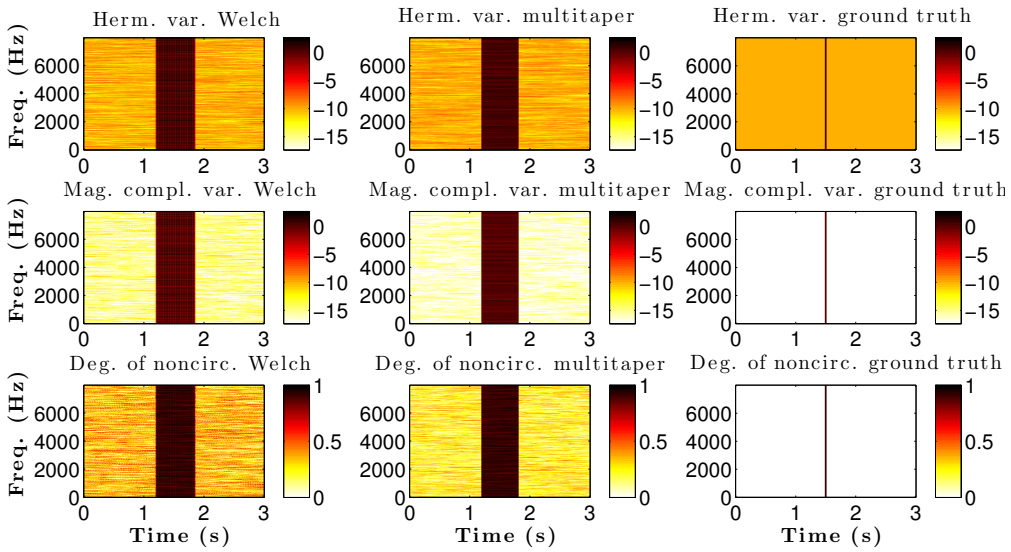

Figure 3.4: Outputs of Welch and multitaper estimators of spectral Hermitian variance (top row), complementary variance (middle row), and degree of noncircularity (bottom row) for a Kronecker delta function in 10dB SNR white noise. The ideal ground truth is given in the right columns of the panels. The estimator uses a bandwidth of $2/(64\text{ms}) = 31.25\text{Hz}$. Reprinted from [159], ©2016 IEEE.

3.3.2 Effect of STFT parameters

STFT parameters have a substantial effect on estimates of noncircularity, especially for narrowband signals. In this section, some of these effects are explored. Consider a discrete-time noisy signal

$$y_n = s_n + v_n, \tag{3.38}$$

where s_n is a narrowband tone and v_n is additive noise. We consider a complex-valued subband with center frequency $\xi_f = 2\pi\frac{f}{M_{DFT}}$, $f = 0..F-1$ with $F = \frac{M_{DFT}}{2} + 1$, to be

$$Y_{f,t} = h_n * \left(y_n e^{-\mathrm{j}2\pi\xi_m n}\right), \tag{3.39}$$

where $\mathbf{h} \in \mathbb{R}^M$ is a real-valued subband filter. If $\mathbf{h}$ is symmetric, (3.39) can be written as

$$Y_{f,t}^S = \sum_m h_{(m-tM_{hop})} y_m e^{-\mathrm{j}2\pi\xi_f m}, \tag{3.40}$$

with $M_{hop} = 1$, which is easily recognized as a maximally-oversampled STFT where $\mathbf{h}$ is the analysis window.

If a subband contains both a tone and noise, it contains two complex-valued components, $S_{f,t}$ and $V_{f,t}$. We consider the estimated impropriety of these two components over a short period $t_0 \leq n < t_0 + L$ under the following assumptions:

1. $\mathbf{h}$ is a real-valued subband filter of duration M.

2. s_n is steady during the period $t_0 \leq n < t_0 + L$, so it can be approximated within the narrow subband as a complex-valued tone, given by

$$S_{f,t} \approx Ae^{\mathrm{j}2\pi(\xi_0 - \xi_f)n + \mathrm{j}\theta}. \tag{3.41}$$

3. v_n is zero-mean, real-valued, white Gaussian noise (WGN). Within the subband, $V_{f,t}$ is complex-valued, narrowband Gaussian noise.

When the center frequency ξ_f of a subband matches the frequency ξ_0 of the tone, $S_{f,t}$ is maximally improper, because according to (3.41), $S_{f,t} \approx Ae^{\mathrm{j}\theta}$, a constant complex value. This constant is shown as the single fixed point in the left panel of figure 3.5. However, when the subband is not aligned, i.e. $\xi_0 \neq \xi_t$, then $S_{f,t}$ will slowly rotate over time, appearing more and more circular. This is shown in the second panel of figure 3.5.

Demodulating WGN, even real-valued WGN, does not affect its whiteness; an example is shown in the third panel of figure 3.5. Thus, since v_n is real-valued WGN, $v_{\xi_f,n} = v_n e^{-\mathrm{j}2\pi\xi_f n}$ is complex-valued WGN. When $v_{\xi_f,n}$ is narrowband-filtered, it appears in the complex plane as a slowly wandering trajectory. An example of narrowband-filtered, demodulated WGN is shown in the right panel of figure 3.5.

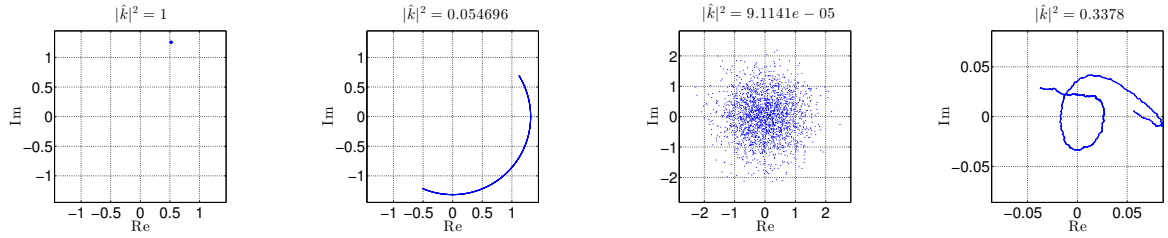

Figure 3.5: Constellations of complex-valued signals in the complex plane, labelled with their estimated degree of impropriety. From left to right: tone in subband with $\omega_0 = \omega_m$, tone in subband with $\omega_0 \neq \omega_m$, demodulated fullband WGN, and narrowband-filtered WGN. Reprinted from [162], ©2015 IEEE.

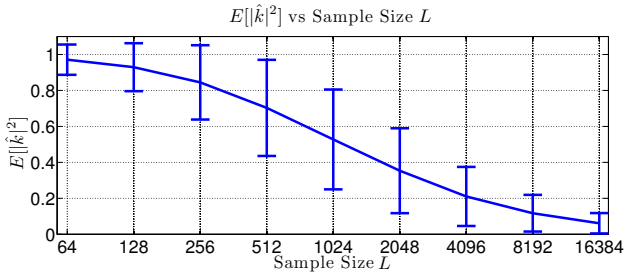

Figure 3.6: Results of Monte Carlo experiment (10000 trials) showing the expected sample impropriety of narrowband-filtered, complex-valued WGN versus sample size L. Reprinted from [162], ©2015 IEEE.

The sample impropriety of filtered WGN depends on the bandwidth of the filter $\mathbf{h}$ and the sample size L. Using a 1024-length Hamming window for the subband filter $\mathbf{h}$, figure 3.6 shows the results of a Monte Carlo experiment measuring the mean and standard deviation of the squared sample circularity coefficient $|\hat{k}|^2$ from (3.33) versus sample size L. Notice that $E[|\hat{k}|^2]$ decreases with increasing L.

3.3.3 Finite sample distribution of the sample circularity coefficient

To determine the suitability of a noncircular model for complex-valued random data, we can perform a hypothesis test:

$$\begin{aligned} \mathcal{H}_0 &: \ |k_x| = 0 \quad (x \text{ is circular}) \\ \mathcal{H}_1 &: \ |k_x| > 0 \quad (x \text{ is noncircular}) \end{aligned} \tag{3.42}$$

Assuming we have L i.i.d. samples $x_{0:L-1}$ of the random variable x, Delmas et al. [31, Remark 5] provide a finite-sample approximation for the sample circularity coefficient $\hat{k}_x$ under these two hypotheses, given by

$$\begin{aligned} \mathcal{H}_0 &: \ |\hat{k}_x| \sim \mathcal{R}\left(\frac{1}{\sqrt{L}}\right), \\ \mathcal{H}_1 &: \ |\hat{k}_x| \sim \mathcal{N}\left(|k_x|, \frac{(1-|k_x|^2)^2}{\sqrt{L}}\right), \end{aligned} \tag{3.43}$$

where $\mathcal{R}$ is a Rayleigh distribution. Using the distribution under the null hypothesis, we can choose a constant probability of false alarm (PFA) and compute a threshold T that we can test $|\hat{k}|$ against to determine statistically significant noncircularity. These pdfs are illustrated in figure 3.7 for $L = 10$ samples, a constant PFA of 0.05, and true circularity coefficient $|k|$ equal to the threshold that achieves PFA = 0.05, which yields a threshold of $T = 0.77$. In practice, this finite-sample approximation has been empirically observed to be accurate for L as low as 10 [23, Appendix C].

Next, we use the finite-sample approximation to demonstrate that a particular nonstationary speech signal exhibits substantial, statistically significant noncircularity. The left panel of figure 3.8 shows a spectrogram of a speech signal from a male speaker, which is the log magnitude of the complex-valued STFT. The sampling rate of the speech signal is 16kHz, and the STFT uses a window length of $M = 512$, a hop of $M_{hop} = 256$, and a DFT length of $M_{DFT} = 512$. In the center panel, we plot the estimated circularity coefficient versus time and frequency bin computed using the Welch's method version of the estimators (3.35) and (3.36) with $L = 10$. Using the finite-sample approximation (3.43) and a PFA of 0.05, we determine the threshold for the hypothesis test (3.42) to be $T = 0.77$. In the right panel

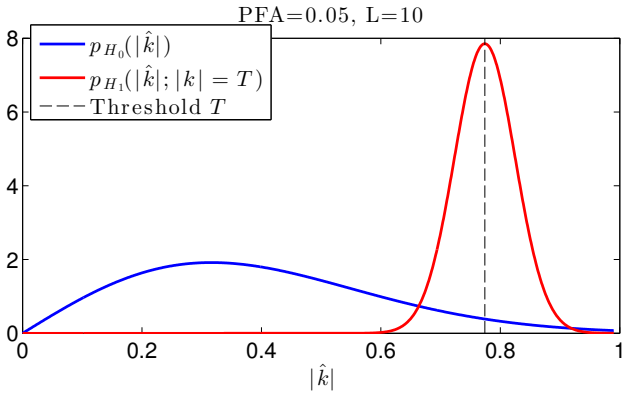

Figure 3.7: Illustration of pdfs for finite-sample approximation of circularity coefficient estimator, for $L = 10$ samples, probability of false alarm (PFA) of 0.05. For illustration purposes, we choose a true circularity coefficient $|k|$ equal to the theshold T, where $T = 0.77$ is chosen to achieve PFA $= 0.05$. Reprinted from [157], ©2016 IEEE.

of figure 3.8, we apply this threshold to the estimated circularity coefficient shown in the center panel. Notice that the speech signal exhibits statistically significant noncircularity, especially at onsets and offsets.

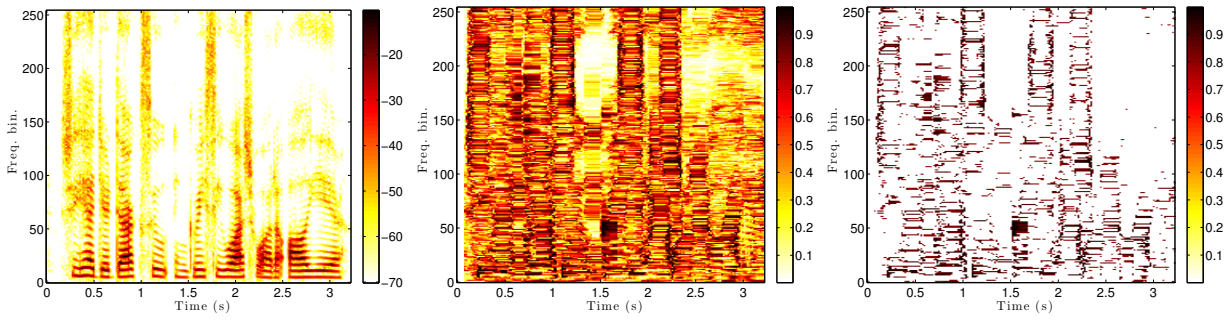

Figure 3.8: Testing for noncircularity of a nonstationary speech signal. Left panel: spectrogram of speech signal. Center panel: estimated circularity coefficient versus time and frequency bin. Right panel: thresholded estimated circularity coefficient with PFA $= 0.05$. Reprinted from [157], ©2016 IEEE.

3.4 Conclusion

In this chapter, we saw that certain classes of real-valued nonstationary signals tend to exhibit second-order noncircularity in the complex-valued STFT domain. First, we described maximum likelihood estimators of the second-order statistics of a noncircular Gaussian model, which can be computed efficiently from the STFT of a single realization of a nonstationary signal. These estimators have a simple finite sample approximation, which allowed us to empirically test the statistical significance of noncircularity in the STFT. In the next chapter, these estimators are applied to the practical problem of detecting real-world nonstationary signals of interest.

Chapter 4

APPLICATIONS OF NONCIRCULAR DISTRIBUTIONS FOR NONSTATIONARY SIGNALS

In this chapter, we will leverage the estimators proposed in the last chapter to build detectors for nonstationary signals. First, section 4.1 applies these estimators to the task of detecting speech in the presence of noise. Next, section 4.2 improves detection of acoustic transients using the proposed estimators of noncircularity.

4.1 Voice Activity Detection

Speech is primarily composed of both harmonically-related narrowband components because of voicing and transients caused by the onsets and offsets of various phonemes. We saw in chapter 3 that both tones and transients can exhibit significant noncircularity in the STFT domain. Thus, in this section, we will exploit the noncircularity of speech to improve voice activity detection.

This section proposes two impropriety-based VADs, one for single-channel and one for dual-channel audio. For both approaches, the complex filterbank in (3.39) of each channel can be efficiently computed using the STFT (3.40) with $\mathbf{h}$ of duration M, a window hop of M_{hop}, and a M_{DFT}-length DFT. To correct for the phase rotations induced by the STFT window hops, a phase modification is applied to each subband, which results in a complex-valued filterbank $Y_{f,t}$ with subbands downsampled by the factor M_{hop}:

$$Y_{f,t} = \left|Y^S_{f,t}\right| \exp \mathrm{j} \left(\angle Y^S_{f,t} - t2\pi\xi_f M_{hop}\right), \tag{4.1}$$

where $\angle Y^S_{f,t}$ is the unwrapped phase of the STFT $Y^S_{f,t}$.

4.1.1 Single-channel method

Given a single-channel audio mix y_n of speech and noise, the STFT observations $Y_{f,t}$ are used to estimate the instantaneous CC at each time/frequency point (f,t). These estimates use a sliding window of duration $L^d = L/M_{hop}$ and hop $L^d_{hop} = L_{hop}/M_{hop}$ in each subband to compute the CC estimate using (3.33):

$$\hat{k}_{Y,f,t} = \frac{\frac{1}{L^d}\sum_{\ell=0}^{L^d-1} Y^2_{f,tL^d_{hop}+\ell}}{\frac{1}{L^d}\sum_{\ell=0}^{L^d-1} \left|Y_{f,tL^d_{hop}+\ell}\right|^2}. \tag{4.2}$$

For each time/frequency point, the CC estimate will be between 0 and 1. Recall from §3.1 that the DOI $|\hat{k}^2_{Y,f,t}|$ corresponds to a GLRT for the impropriety of $Y_{f,t}$.

The test statistic for the VAD is chosen to be the estimated DOIs averaged across frequency, which will be referred to as the "summed degree of impropriety" (SDOI). The SDOI for frame t is given by

$$\text{SDOI}_t = \frac{1}{F}\sum_{f=0}^{F-1} |\hat{k}_{Y,f,t}|^2, \tag{4.3}$$

The SDOI will take on values between 0 and 1. To see the effectiveness of the SDOI for discriminating between speech-plus-noise and noise-only, we can examine its empirical distribution under the two hypotheses: noise-only or speech-plus-noise. Using $M_{hop} = 16$ and a sliding window length of $L = 2048$ with hop $L_{hop} = 80$, figure 4.2 shows the distributions of the SDOI for 50 minutes of audio with sampling rate $\xi_s = 8$ kHz, consisting of TIMIT utterances [45] embedded at 0 dB SNR in car noise (window down, highway driving) from the QUT-NOISE [29] database.

To get intuition about why these distributions are different, figure 4.1 compares the estimated DOIs of speech, noise, and the mix for a short clip of the audio used for figure 4.2. Notice that time/frequency points where speech is highly improper tend to dominate over more proper noise.

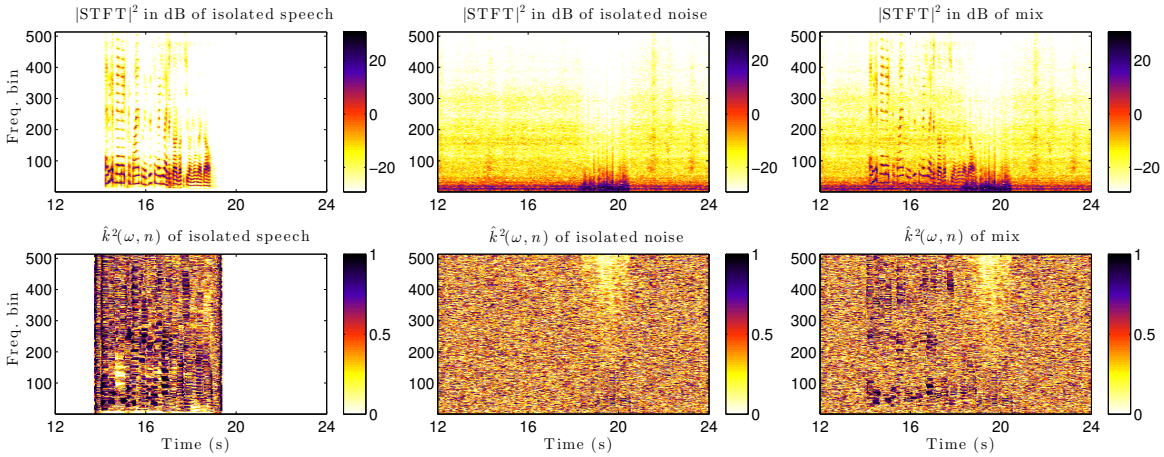

Figure 4.1: The high impropriety of speech (lower left, quiet shown as white) tends to dominate over more proper noise (lower middle) in the estimated impropriety of the mixed speech and noise (lower right). Spectrograms are shown in top panels. Reprinted from [162], ©2015 IEEE.

4.1.2 Two-channel method

For two-channel data, we use the L-length sliding window to estimate the 2×2 spatial covariance matrices $\widetilde{\mathbf{R}}_{YY,f,t}$ and $\mathbf{R}_{YY,f,t}$ at each time/frequency point (f, t). These estimated covariances are used to compute the two-dimensional circularity spectrum $\mathbf{k}_{f,t}$ for each time-frequency bin using (3.9).

To get a maximally discriminative test statistic, linear discriminant analysis (LDA) [106, Section 8.6.3] is used to train two F-length vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ using $\mathbf{k}_{f,t}$ and ground-truth labels. The resulting statistic will be referred to as "circularity spectrum with linear discriminant analysis" (CS-LDA), and it is given by

$$\text{CS-LDA}_t = \frac{1}{2}\sum_{i=1}^{2}\left(\frac{1}{F}\sum_{f=1}^{F} a_{i,f}k_{f,t,i}^2\right). \tag{4.4}$$

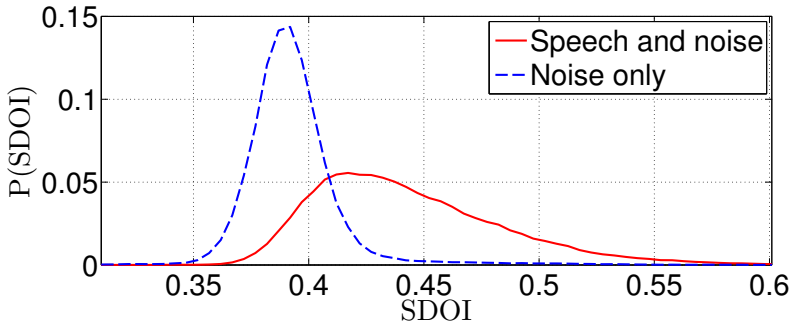

Figure 4.2: Empirical distributions of summed degree of impropriety (SDOI) under two hypotheses for 50 minutes of speech in 0 dB SNR car noise (windows down, highway driving). Reprinted from [162], ©2015 IEEE.

4.1.3 Experiments

To test the performance of VAD using the SDOI (4.3) and CS-LDA (4.4), we use the QUT-NOISE-TIMIT corpus [29], which consists of TIMIT utterances [45] embedded in two-channel recordings of five different types of real-world noise, with each noise type recorded at two different locations for two different sessions (total of 20 noise conditions). Two noise types, CAR and REVERB, also include exponentially-swept sines, which allows estimation of two-channel reverberation impulse responses (RIRs) using the technique of Farina [41]. The CAR RIRs are relatively short, while the REVERB RIRs are highly reverberant. The locations of the noise types given by Dean et al. [29] are listed in table 4.1.

There are six SNRs ranging from -10 dB to 15 dB. We use the `sA` subset of QUT-NOISE-TIMIT, which consists of 6000, 60-second files, for 100 total hours of audio. For each of the six SNRs, there are 50 files for each noise type, location, and session. 25% of the files contain 25% or less of speech, 50% have 25% to 75% speech, and the remainder have 75% or more speech. We use a sampling rate of $\xi_s = 8$ kHz. For noise types without RIRs provided, the second channel of speech is a copy of the first channel of speech.

We compare to two baseline VAD methods: Sohn et al.'s statistical model-based like-

Table 4.1: QUT-NOISE noise types and locations. Reprinted from [162], ©2015 IEEE.

Noise type	Location 1	Location 2
CAFE	Cafe	Food court
HOME	Kitchen	Living room
STREET	City	Suburb
CAR	Window down	Window up
REVERB	Car park	Pool

lihood ratio test [136] and Ramírez et al.'s long-term spectral divergence (LTSD) [122]. These methods are well-suited for comparison, because they use the magnitude (-squared) of complex STFTs to derive their detection statistics. Thus, we are comparing the proposed noncircularity-based VADs to state-of-the-art methods that use magnitude.

To try and ensure a fair comparison to the two-channel CS-LDA method, we apply a delay-and-sum beamformer (DSB) to two-channel data before applying single-channel baselines, which gives an average relative improvement of 4.57% in overall HTER versus the single-channel baselines alone.

The procedure for evaluating VAD performance is exactly the same as that used by Dean et al. [29] and Ghaemmaghami et al. [53], which is computing the half-total error rate (HTER) at an optimal detection threshold. The HTER is given by the average of the false-alarm rate (FAR) and the miss rate (MR). The FAR and MR are given by

$$\text{FAR} = \frac{\#\text{ false positives}}{\#\text{ negatives}} \quad \text{MR} = \frac{\#\text{ false negatives}}{\#\text{ positives}} \tag{4.5}$$

To choose the detection threshold for each method—and, in the case of CS-LDA, to train the LDA vectors $\mathbf{a}_i(\omega)$—we follow the same procedure as Ghaemmaghami et al. [53], which uses cross-validation between noise locations to choose detection thresholds for a particular noise type. For example, the detection threshold for STREET-City noise is given by the threshold that minimizes HTER on STREET-Suburb noise, and vice versa. This cross-

Figure 4.3: Overall VAD results averaged across noise types on the QUT-NOISE-TIMIT corpus. Reprinted from [162], ©2015 IEEE.

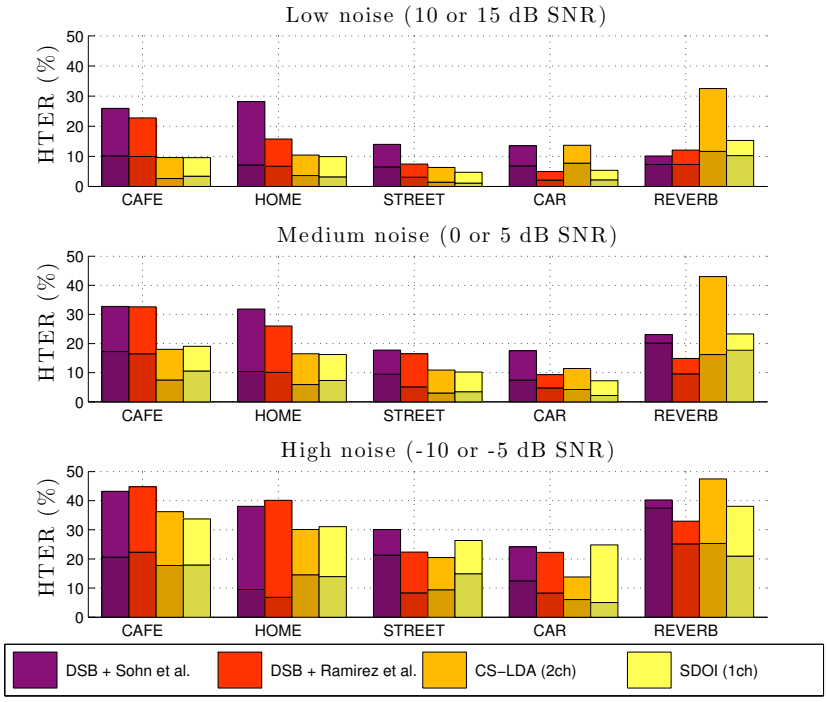

Figure 4.4: VAD results on the QUT-NOISE-TIMIT corpus. In each stack, top lighter-shaded bar corresponds to %FAR/2, and bottom darker-shaded bar corresponds to %MR/2. Reprinted from [162], ©2015 IEEE.

validation ensures that the methods are robust across different realizations of the same type of noise. Also, the six SNRs are grouped into three subsets (low, medium, and high noise), which evaluates the robustness of the methods to different levels of noise.

Decisions for the methods are made in 10 ms increments. For Sohn et al.'s method, we use the implementation from the VOICEBOX Matlab toolbox with default parameters [15]. For Ramírez et al., we use default parameters from their paper [122]. For SDOI and CS-LDA, we use a STFT window length equal to the DFT size: $M = M_{DFT} = 1024$, with a

Method	Low noise (10 or 15 dB SNR)			Medium noise (0 or 5 dB SNR)			High noise (−10 or −5 dB SNR)		
	%FAR	%MR	%HTER	%FAR	%MR	%HTER	%FAR	%MR	%HTER
DSB + Sohn et al.	15.17	21.49	18.33	25.91	23.24	24.58	40.55	**29.77**	35.16
DSB + Ramirez et al.	11.63	13.54	12.58	18.44	21.31	19.87	28.38	36.64	32.51
CS-LDA (2ch)	10.80	18.16	14.48	**14.75**	25.17	19.96	29.26	30.01	**29.63**
SDOI (1ch)	**8.03**	**9.86**	**8.95**	16.48	**13.94**	**15.21**	**29.13**	32.47	30.80

STFT hop of $M_{hop} = 16$, averaging window length of $L = 2048$, and averaging window hop of $L_{hop} = 80$. The resulting decision labels are median filtered with a window of 1 second.

The results are shown in table 4.1.3 and figure 4.4. Our new VADs achieve equivalent or superior performance to the baseline methods on all noise types except for the REVERB noise type. Interestingly, the unsupervised single-channel SDOI statistic performs better than the supervised two-channel CS-LDA at lower noise levels (0 to 15 dB SNR), while CS-LDA performs better at higher noise levels (−10 and −5 dB SNR). Vulnerability to high amounts of reverberation is expected, since adding many shifted copies of improper signals together in a subband tends to make them look more proper.

4.2 Acoustic Transient Detection

To further demonstrate the benefit of noncircularity for detection, we use Task 2 of the Detection and Classification of Acoustic Events 2016 (DCASE2016) dataset [139, 154]. This dataset consists of eleven different types of acoustic events embedded in stationary background noise typical of an office environment. These events include clearing throat, coughing, door knock, door slam, drawer, human laughter, keyboard, keys put on table, page turning, phone ringing, and speech. Our goal is to detect when at least one acoustic event is active in 8 millisecond increments. Acoustic events are mixed with the background noise at −6dB, 0dB, and 6dB signal to noise ratio (SNR).

We will assume that the background noise v_n is stationary, which means its STFT coefficients $V_{f,t}$ are circular with frequency-dependent Hermitian variance $R_{vv,f}$. We will also assume that nonstationary acoustic events exhibit noncircularity in the STFT domain. Thus,

the detection problem for a particular observed STFT frame $Y_{0:F-1,t}$ at time t is

$$\begin{aligned} \mathcal{H}_0 :\ & Y_{f,t} = V_{f,t}, && \forall f, \\ \mathcal{H}_1 :\ & Y_{f,t} = V_{f,t} + X_{f,t}, && \forall f, \end{aligned} \tag{4.6}$$

where the $X_{f,t}$ for $f = 0..F-1$ are the STFT coefficients of the acoustic event.

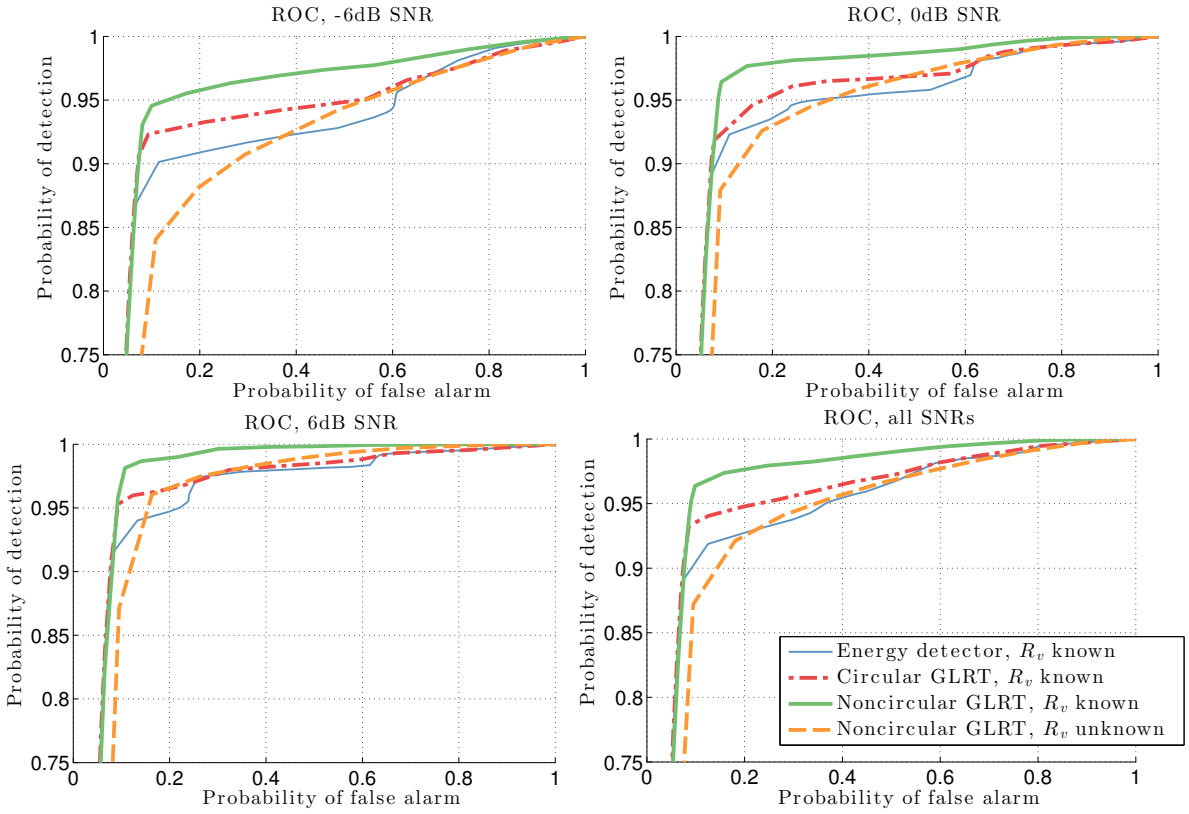

Figure 4.5: Receiver operating characteristic (ROC) curves for DCASE2016 event detection with FFT length $N = 1024$ and averaging window length $L = 64$ for various SNRs. Reprinted from [157], ©2016 IEEE.

The generalized likelihood ratio test (GLRT) is defined as

$$G(y) \triangleq \left[\max_{\theta_1} p_{\mathcal{H}_1}(y|\theta_1)\right] / \left[\max_{\theta_0} p_{\mathcal{H}_0}(y|\theta_0)\right], \tag{4.7}$$

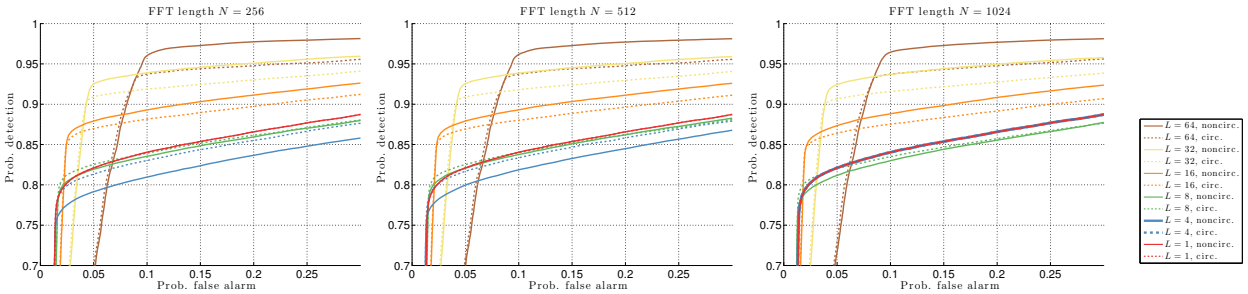

Figure 4.6: ROC curves for parameter sweep over FFT length N and averaging window length L. Notice that detection performance is relatively invariant to FFT length N for larger Ls, and that there is a tradeoff between probability of false alarm and probability of detection for larger Ls. Reprinted from [157], ©2016 IEEE.

where θ_0 and θ_1 are the unknown parameters under hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$, respectively. If we assume that $X_{f,t}$ are circular Gaussians, the log GLRT for the detection problem (4.6) given observed STFT coefficients $\mathbf{Y}_t := \mathbf{Y}_{0:F-1,t}$ is

$$\log G_C(\mathbf{Y}_t) = \sum_f \left(\frac{\hat{R}_{yy,f,t}}{R_{vv,f}} - \log \frac{\hat{R}_{yy,f,t}}{R_{vv,f}} \right). \tag{4.8}$$

When the $X_{f,t}$ are noncircular, the log GLRT is simply the circular log GLRT, $\log G_C(\mathbf{Y}_t)$ given by (4.8), plus an additional term:

$$\log G_{NC}(\mathbf{Y}_t) = \log G_C(\mathbf{Y}_t) - \frac{1}{2} \sum_f \log \left(1 - \left| \hat{k}_{y,f,t} \right|^2 \right). \tag{4.9}$$

This additional term can also be used by itself as a blind detector, that is

$$\log G_{NC}^{\text{blind}}(\mathbf{Y}_t) = \log G_{NC}(\mathbf{Y}_t) - \log G_C(\mathbf{Y}_t). \tag{4.10}$$

This detector is blind in the sense that it does not require knowledge of the Hermitian noise variance $R_{vv,f}$. In practice, we estimate the noise variance $R_{vv,f}$ from all observation samples before the first acoustic event, which we know *a priori* to only contain noise. The blind detector is useful in situations where we do not know the when noise-only durations occur.

Using a DFT length of $N = 1024$ and averaging window length of $L = 64$, receiver operating curves (ROCs) for DCASE2016 event detection for various SNRs and the overall ROC curves across all SNRs are shown in figure 4.5. Note that the noncircular GLRT (solid thick green line) achieves the best performance versus all other detectors at all SNRs. The blind noncircular GLRT (dashed orange line) performs the worst, but has the advantage that it does not require knowledge of the Hermitian background noise variance $R_{vv,f}$. Note that if $R_{vv,f}$ is not known or cannot be estimated, detection using only circular Hermitian statistics is impossible.

4.2.1 Effect of estimator parameters

In this section, we empirically explore the effect of the estimation parameters on DCASE2016 acoustic event detection. These parameters are the DFT length N, and the averaging window length L. Figure 4.6 shows ROC curves using the circular GLRT (4.8) and noncircular GLRT (4.9) for various settings of N and L. Notice that for larger L, the DFT length N does not have much effect on detection performance. Also, the noncircular detector does not start improving performance versus the circular detector until $L = 16$, which suggests that a certain minimum number of samples is required in order for a noncircular model to provide benefit.

To quantify the performance improvement and determine the best setting of estimator parameters, we use area under the curve (AUC), which is the integral of the ROC curve. For various FFT lengths N, figure 4.7 plots the AUCs for the ROCs in figure 4.6 versus averaging window length L. From the AUC plot, settings of $L = 32$ and $L = 64$ yield the best AUC, while FFT length N does not have much effect for larger L.

4.3 Conclusion

This chapter has shown that using noncircular Gaussian distributions to model the STFT provides distinct benefits for processing nonstationary signals. In particular, we presented two applications of STFT noncircularity for nonstationary signals: voice activity detection

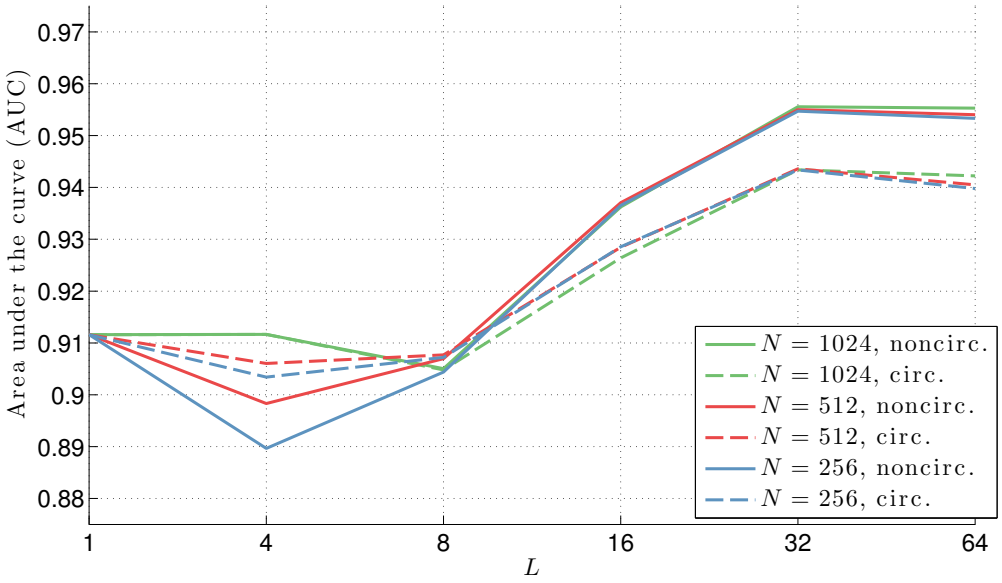

Figure 4.7: Areas under the curve (AUCs) for parameter sweep ROCs in figure 4.6. Notice that $L = 32$ and $L = 64$ appear to be the best settings of averaging window length, while the AUC is not very dependent on DFT length N. Reprinted from [157], ©2016 IEEE.

and acoustic transient detection. For both types of signal, exploiting the additional noncircular information models provided significant improvement in terms of total error rate, in the case of voice activity detection, and in detection of acoustic transients versus false alarm rate, as quantified by the ROC curves.

Chapter 5

IMPROVING UNITARY RECURRENT NEURAL NETWORKS

This chapter proposes an improvement to unitary recurrent neural networks, originally proposed by Arjovsky et al. [3] and described in section 2.5.6. The proposed improvement is motivated by a fundamental problem: the original uRNN proposed by Arjovsky et al. uses a parameterization of the unitary recurrence matrix, given by (2.44) that is limited. This restricted capacity limits the representational ability of the uRNN, which hinders its ability to process nonstationary signals. The two main components of this chapter's contribution can be summarized as follows:

1) A theoretical argument is provided to determine the smallest dimension N for which any parameterization of the unitary recurrence matrix does not cover the entire set of all unitary matrices. The argument relies on counting real-valued parameters and using Sard's theorem to show that the smooth map from these parameters to the unitary manifold is not onto. Using this argument, we show that a previously proposed parameterization [3] cannot represent all unitary matrices larger than 7×7. Thus, such a parameterization results in what we refer to as a *restricted-capacity* unitary recurrence matrix.

2) To overcome the limitations of restricted-capacity parameterizations, a new method is proposed for stochastic gradient descent for training the unitary recurrence matrix, which constrains the gradient to lie on the differentiable manifold of unitary matrices. This approach allows us to directly optimize a complete, or *full-capacity*, unitary matrix. Neither restricted-capacity nor full-capacity unitary matrix optimization require gradient clipping. Furthermore, full-capacity optimization still achieves good results without adaptation of the learning rate during training.

To test the limitations of a restricted-capacity representation and to confirm that our

full-capacity uRNN does have practical implications, we test restricted-capacity and full-capacity uRNNs on both synthetic and natural data tasks. These tasks include synthetic system identification, long-term memorization, frame-to-frame prediction of speech spectra, and pixel-by-pixel classification of handwritten digits. The proposed full-capacity uRNNs generally achieve equivalent or superior performance on synthetic and natural data compared to both LSTMs [67] and the original restricted-capacity uRNNs [3].

Section 5.1 presents the first component of this chapter's contribution: the theoretical argument to determine if any unitary parameterization has restricted-capacity. Section 5.2 describes the second component, where we show how to optimize a full-capacity unitary matrix. These results are confirmed with simulated and natural data in section 5.3. Section 5.4 provides a conclusion.

5.1 Estimating the Representation Capacity of Structured Unitary Matrices

In this section, we state and prove a theorem that can be used to determine when any particular unitary parameterization does not have capacity to represent all unitary matrices. As an application of this theorem, we show that the parameterization (2.44) does not have the capacity to cover all $N \times N$ unitary matrices for $N > 7$. First, we establish an upper bound on the number of real-valued parameters required to represent any $N \times N$ unitary matrix. Then, we state and prove our theorem.

Lemma 5.1.1 *The set of all unitary matrices is a manifold of dimension N^2.*

Proof: The set of all unitary matrices is the well-known unitary Lie group $U(N)$ [54, §3.4]. A Lie group identifies group elements with points on a differentiable manifold [54, §2.2]. The dimension of the manifold is equal to the dimension of the Lie algebra $\mathfrak{u}$, which is a vector space that is the tangent space at the identity element [54, §4.5]. For $U(N)$, the Lie algebra consists of all skew-Hermitian matrices $\mathbf{A}$ [54, §5.4]. A skew-Hermitian matrix is any $\mathbf{A} \in \mathbb{C}^{N \times N}$ such that $\mathbf{A} = -\mathbf{A}^{\mathsf{H}}$, where $(\cdot)^{\mathsf{H}}$ is the conjugate transpose. To determine the dimension of $U(N)$, we can determine the dimension of $\mathfrak{u}$. Because of the skew-Hermitian

constraint, the diagonal elements of $\mathbf{A}$ are purely imaginary, which corresponds to N real-valued parameters. Also, since $A_{i,j} = -A^*_{j,i}$, the upper and lower triangular parts of $\mathbf{A}$ are parameterized by $\frac{N(N-1)}{2}$ complex numbers, which corresponds to an additional $N^2 - N$ real parameters. Thus, $U(N)$ is a manifold of dimension N^2. □

Theorem 5.1.2 *If a family of $N \times N$ unitary matrices is parameterized by P real-valued parameters for $P < N^2$, then it cannot contain all $N \times N$ unitary matrices.*

Proof: We consider a family of unitary matrices that is parameterized by P real-valued parameters through a smooth map $g : \mathcal{P}(P) \to \mathcal{U}(N^2)$ from the space of parameters $\mathcal{P}(P)$ to the space of all unitary matrices $\mathcal{U}(N^2)$. The space $\mathcal{P}(P)$ of parameters is considered as a P-dimensional manifold, while the space $\mathcal{U}(N^2)$ of all unitary matrices is an N^2-dimensional manifold according to lemma 5.1.1. Then, if $P < N^2$, Sard's theorem [127] implies that the image $g(\mathcal{P})$ of g is of measure zero in $\mathcal{U}(N^2)$, and in particular g is not onto. Since g is not onto, there must exist a unitary matrix $\mathbf{U} \in \mathcal{U}(N^2)$ for which there is no corresponding input $\mathbf{P} \in \mathcal{P}(P)$ such that $\mathbf{U} = g(\mathbf{P})$. Thus, if P is such that $P < N^2$, the manifold $\mathcal{P}(P)$ cannot represent all unitary matrices in $\mathcal{U}(N^2)$. □

We now apply Theorem 5.1.2 to the parameterization (2.44). Note that the parameterization (2.44) has $P = 7N$ real-valued parameters. If we solve for N in $7N < N^2$, we get $N > 7$. Thus, the parameterization (2.44) cannot represent all unitary matrices for dimension $N > 7$.

5.2 Optimizing Full-Capacity Unitary Matrices

In this section, we show how to get around the limitations of restricted-capacity parameterizations and directly optimize a full-capacity unitary matrix. We consider the Stiefel manifold of all $N \times N$ complex-valued matrices whose columns are N orthonormal vectors in $\mathbb{C}^N$ [143]. Mathematically, the Stiefel manifold is defined as

$$\mathcal{V}_N(\mathbb{C}^N) = \left\{ \mathbf{U} \in \mathbb{C}^{N \times N} : \mathbf{U}^{\mathsf{H}}\mathbf{U} = \mathbf{I}_{N \times N} \right\}. \tag{5.1}$$

For any $\mathbf{U} \in \mathcal{V}_N(\mathbb{C}^N)$, any matrix $\mathbf{Z}$ in the tangent space $\mathcal{T}_{\mathbf{U}}\mathcal{V}_N(\mathbb{C}^N)$ of the Stiefel manifold satisfies $\mathbf{Z}\mathbf{U}^{\mathsf{H}} - \mathbf{U}\mathbf{Z}^{\mathsf{H}} = 0$ [143]. The Stiefel manifold becomes a Riemannian manifold when its tangent space is equipped with an inner product. Tagare [143] suggests using the canonical inner product, given by

$$\langle \mathbf{Z}_1, \mathbf{Z}_2 \rangle_c = \mathrm{tr}\left(\mathbf{Z}_1^{\mathsf{H}}(\mathbf{I} - \frac{1}{2}\mathbf{U}\mathbf{U}^{\mathsf{H}})\mathbf{Z}_2\right). \tag{5.2}$$

Under this canonical inner product on the tangent space, the gradient in the Stiefel manifold of the loss function f with respect to the matrix $\mathbf{U}$ is $\mathbf{A}\mathbf{U}$, where $\mathbf{A} = \mathbf{G}\mathbf{U}^{\mathsf{H}} - \mathbf{U}\mathbf{G}^{\mathsf{H}}$ is a skew-Hermitian matrix and $\mathbf{G}$ with $G_{i,j} = \frac{\delta f}{\delta W_{i,j}}$ is the usual gradient of the loss function f with respect to the matrix $\mathbf{U}$ [143]. Using these facts, Tagare [143] suggests a descent curve along the Stiefel manifold at training iteration k given by the matrix product of the Cayley transformation of $\mathbf{A}^{(k)}$ with the current solution $\mathbf{U}^{(k)}$:

$$\mathbf{Y}^{(k)}(\lambda) = \left(\mathbf{I} + \frac{\lambda}{2}\mathbf{A}^{(k)}\right)^{-1}\left(\mathbf{I} - \frac{\lambda}{2}\mathbf{A}^{(k)}\right)\mathbf{U}^{(k)}, \tag{5.3}$$

where λ is a learning rate and $\mathbf{A}^{(k)} = \mathbf{G}^{(k)}\mathbf{U}^{(k)^{\mathsf{H}}} - \mathbf{U}^{(k)}\mathbf{G}^{(k)^{\mathsf{H}}}$. Gradient descent proceeds by performing updates $\mathbf{U}^{(k+1)} = \mathbf{Y}^{(k)}(\lambda)$. Tagare [143] suggests an Armijo-Wolfe search along the curve to adapt λ, but such a procedure would be expensive for neural network optimization since it requires multiple evaluations of the forward model and gradients. We found that simply using a fixed learning rate λ often works well. Also, RMSprop-style scaling of the gradient $\mathbf{G}^{(k)}$ by a running average of the previous gradients' norms [147] before applying the multiplicative step (5.3) can improve convergence. The only additional substantial computation required beyond the forward and backward passes of the network is the $N \times N$ matrix inverse in (5.3).

5.3 Experiments

All models are implemented in Theano [144], based on the implementation of restricted-capacity uRNNs by [3], available from `https://github.com/amarshah/complex_RNN`. All code to replicate these results is available from `https://github.com/stwisdom/urnn`. All

models use RMSprop [147] for optimization, except that full-capacity uRNNs optimize their recurrence matrices with a fixed learning rate using the update step (5.3) and optional RMSprop-style gradient normalization.

5.3.1 Synthetic data

First, we compare the performance of full-capacity uRNNs to restricted-capacity uRNNs and LSTMs on two tasks with synthetic data. The first task is synthetic system identification, where a uRNN must learn the dynamics of a target uRNN given only samples of the target uRNN's inputs and outputs. The second task is the copy memory problem, in which the network must recall a sequence of data after a long period of time.

System identification

For the task of system identification, we consider the problem of learning the dynamics of a nonlinear dynamical system that has the form (2.30)-(2.31), given a dataset of inputs and outputs of the system. We will draw a true system $\mathbf{U}_{sys}$ randomly from either a constrained set $\mathcal{U}_u$ of restricted-capacity unitary matrices using the parameterization $\mathbf{U}_u(\theta_u)$ in (2.44) or from a wider set $\mathcal{U}_g$ of restricted-capacity unitary matrices that are guaranteed to lie outside $\mathcal{U}_u$. We sample from $\mathcal{U}_g$ by taking a matrix product of two unitary matrices drawn from $\mathcal{U}_u$.

We use a sequence length of $T = 150$, and we set the input dimension and output dimension both equal to the hidden state dimension N. The input-to-hidden transformation $\mathbf{V}$ and output-to-hidden transformation $\mathbf{Z}$ are both set to identity, the output bias $\mathbf{c}$ is set to $\mathbf{0}$, the initial state is set to $\mathbf{0}$, and the hidden bias $\mathbf{b}$ is drawn from a uniform distribution in the range $[-0.11, -0.09]$. The hidden bias has a mean of -0.1 to ensure stability of the system outputs. Inputs are generated by sampling T-length i.i.d. sequences of zero-mean, diagonal and unit covariance circular complex-valued Gaussians of dimension N. The outputs are created by running the system (2.30)-(2.31) forward on the inputs.

We compare a restricted-capacity uRNN using the parameterization from (2.44) and a full-capacity uRNN using Stiefel manifold optimization with no gradient normalization

as described in Section 5.2. We choose hidden state dimensions N to test critical points predicted by our arguments in Section 5.1 of $\mathbf{U}_u(\theta_u)$ in (2.44): $N \in \{4, 6, 7, 8, 16\}$. These dimensions are chosen to test below, at, and above the critical dimension of 7.

For all experiments, the number of training, validation, and test sequences are 20000, 1000, and 1000, respectively. Mean-squared error (MSE) is used as the loss function. The learning rate is 0.001 with a batch size of 50 for all experiments. Both models use the same matrix drawn from $\mathcal{U}_u$ as initialization. To isolate the effect of unitary recurrence matrix capacity, we only optimize $\mathbf{U}$, setting all other parameters to true oracle values. For each method, we report the best test loss over 100 epochs and over 6 random initializations for the optimization.

The results are shown in Table 5.1. "$\mathbf{U}_{sys}$ init." refers to the initialization of the true system unitary matrix $\mathbf{U}_{sys}$, which is sampled from either the restricted-capacity set $\mathcal{U}_u$ or the wider set $\mathcal{U}_g$.

Table 5.1: Results for system identification in terms of best normalized MSE. $\mathcal{U}_u$ is the set of restricted-capacity unitary matrices from (2.44), and $\mathcal{U}_g$ is a wider set of unitary matrices.

$\mathbf{U}_{sys}$ init.	Capacity	$N = 4$	$N = 6$	$N = 7$	$N = 8$	$N = 16$
$\mathcal{U}_u$	Restricted	4.81e−1	**6.75e−3**	3.53e−1	3.51e−1	7.30e−1
$\mathcal{U}_u$	Full	**1.28e−1**	3.03e−1	**2.16e−1**	**5.04e−2**	**1.28e−1**
$\mathcal{U}_g$	Restricted	**3.21e−4**	**3.36e−1**	3.36e−1	2.69e−1	7.60e−1
$\mathcal{U}_g$	Full	8.72e−2	3.86e−1	**2.62e−1**	**7.22e−2**	**1.00e−6**

Notice that for $N < 7$, the restricted-capacity uRNN achieves comparable or better performance than the full-capacity uRNN. At $N = 7$, the restricted-capacity and full-capacity uRNNs achieve relatively comparable performance, with the full-capacity uRNN achieving slightly lower error. For $N > 7$, the full-capacity uRNN always achieves better performance versus the restricted-capacity uRNN. This result confirms our theoretical arguments that

the restricted-capacity parameterization in (2.44) lacks the capacity to model all matrices in the unitary group for $N > 7$ and indicates the advantage of using a full-capacity unitary recurrence matrix.

Copy memory problem

The experimental setup follows the copy memory problem from [3], which itself was based on the experiment from [67]. We consider alternative hidden state dimensions and extend the sequence lengths to $T = 1000$ and $T = 2000$, which are longer than the maximum length of $T = 750$ considered in previous literature.

In this task, the data is a vector of length $T + 20$ and consists of elements from 10 categories. The vector begins with a sequence of 10 symbols sampled uniformly from categories 1 to 8. The next $T - 1$ elements of the vector are the ninth 'blank' category, followed by an element from the tenth category, the 'delimiter'. The remaining ten elements are 'blank'. The task is to output $T+10$ blank characters followed by the sequence from the beginning of the vector. We use average cross entropy as the training loss function. The baseline solution outputs the blank category for $T + 10$ time steps and then guesses a random symbol uniformly from the first eight categories. This baseline has an expected average cross entropy of $\frac{10 \log(8)}{T+20}$.

The full-capacity uRNN uses a hidden state size of $N = 128$ with no gradient normalization. To match the number of parameters (≈ 22k), we use $N = 470$ for the restricted-capacity uRNN, and $N = 68$ for the LSTM. The training set size is 100000 and the test set size is 10000. The results of the $T = 1000$ experiment can be found on the left half of Figure 5.1. The full-capacity uRNN converges to a solution with zero average cross entropy after about 2000 training iterations, whereas the restricted-capacity uRNN settles to the baseline solution of 0.020. The results of the $T = 2000$ experiment can be found on the right half of Figure 5.1. The full-capacity uRNN hovers around the baseline solution for about 5000 training iterations, after which it drops down to zero average cross entropy. The restricted-capacity again settles down to the baseline solution of 0.010. These results demonstrate that

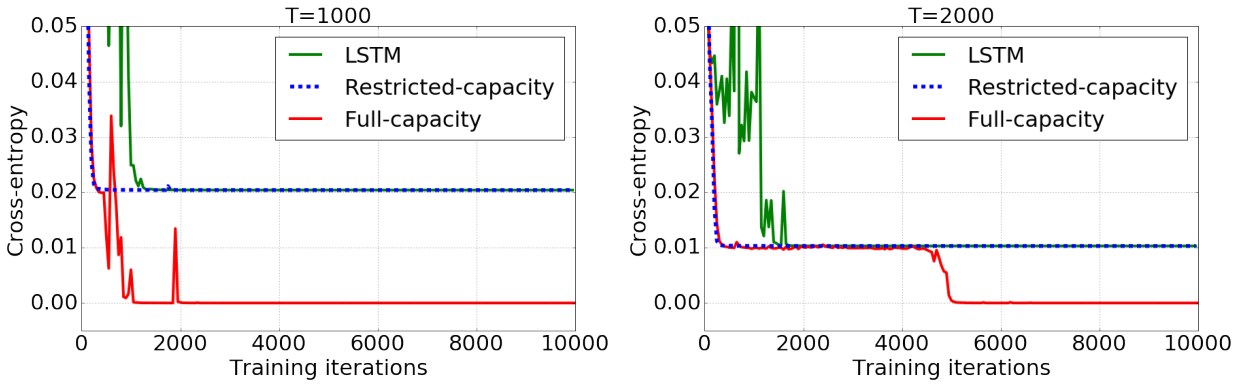

Figure 5.1: Results of the copy memory problem with sequence lengths of 1000 (left) and 2000 (right). The full-capacity uRNN converges quickly to a perfect solution, while the LSTM and restricted-capacity uRNN with approximately the same number of parameters are unable to improve past the baseline naive solution.

the full-capacity uRNN is very effective for problems requiring very long memory.

5.3.2 Speech data

We now apply restricted-capacity and full-capacity uRNNs to real-world speech data and compare their performance to LSTMs. The main task we consider is predicting the log-magnitude of future frames of a STFT.

The frame prediction task is as follows: given all the log-magnitudes of STFT frames up to time t, predict the log-magnitude of the STFT frame at time $t + 1$.We use the TIMIT dataset [50]. According to common practice [63], we use a training set with 3690 utterances from 462 speakers, a validation set of 400 utterances, an evaluation set of 192 utterances. Training, validation, and evaluation sets have distinct speakers. Results are reported on the evaluation set using the network parameters that perform best on the validation set in terms of the loss function over three training trials. All TIMIT audio is resampled to 8kHz. The STFT uses a Hann analysis window of 256 samples (32 milliseconds) and a window hop of 128 samples (16 milliseconds).

The LSTM requires gradient clipping during optimization, while the restricted-capacity and full-capacity uRNNs do not. The hidden state dimensions N of the LSTM are chosen to match the number of parameters of the full-capacity uRNN. For the restricted-capacity uRNN, we run models that match either N or number of parameters. For the LSTM and restricted-capacity uRNNs, we use RMSprop [147] with a learning rate of 0.001, momentum 0.9, and averaging parameter 0.1. For the full-capacity uRNN, we also use RMSprop to optimize all network parameters, except for the recurrence matrix, for which we use stochastic gradient descent along the Stiefel manifold using the update (5.3) with a fixed learning rate of 0.001 and no gradient normalization.

Table 5.2: Log-magnitude STFT prediction results on speech data, evaluated using objective and perceptual metrics (see text for description).

Model	N	# parameters	Valid. MSE	Eval. MSE	SegSNR (dB)	STOI	PESQ
LSTM	84	≈83k	18.02	18.32	1.95	0.77	1.99
Restricted-capacity uRNN	128	≈67k	15.03	15.78	3.30	0.83	2.36
Restricted-capacity uRNN	158	≈83k	15.06	**14.87**	3.32	0.83	2.33
Full-capacity uRNN	128	≈83k	**14.78**	15.24	**3.57**	**0.84**	**2.40**
LSTM	120	≈135k	16.59	16.98	2.32	0.79	2.14
Restricted-capacity uRNN	192	≈101k	15.20	15.17	3.31	0.83	2.35
Restricted-capacity uRNN	256	≈135k	15.27	15.63	3.31	0.83	2.36
Full-capacity uRNN	192	≈135k	**14.56**	**14.66**	**3.76**	**0.84**	**2.42**
LSTM	158	≈200k	15.49	15.80	2.92	0.81	2.24
Restricted-capacity uRNN	378	≈200k	15.78	16.14	3.16	0.83	2.35
Full-capacity uRNN	256	≈200k	**14.41**	**14.45**	**3.75**	**0.84**	**2.38**

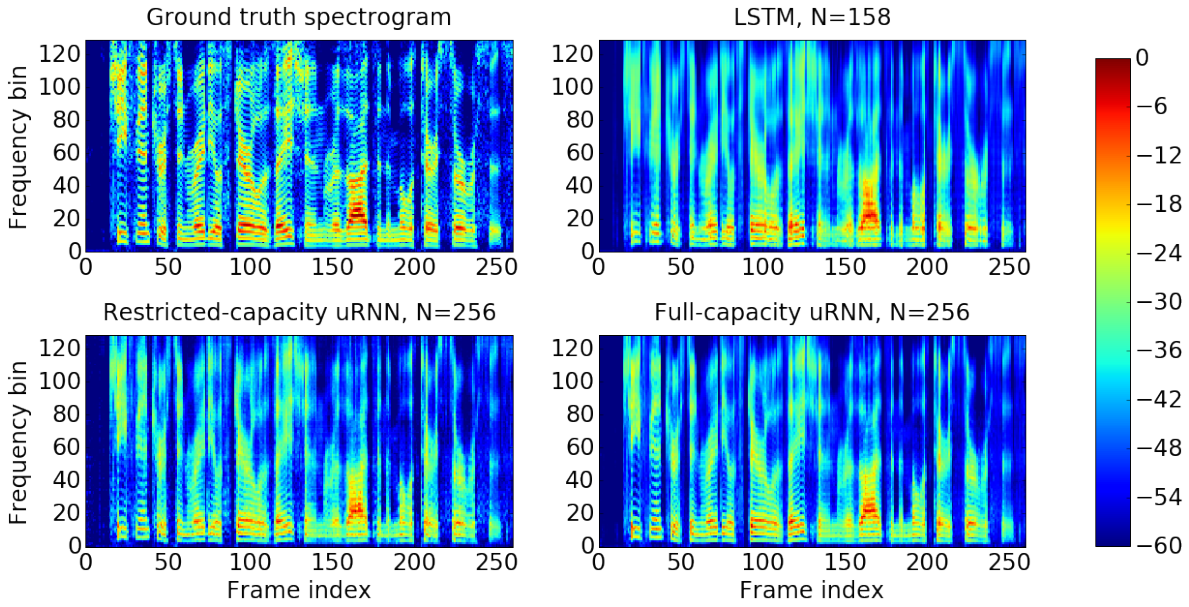

Figure 5.2: Ground truth and one-frame-ahead predictions of a spectrogram for an example utterance. For each model, hidden state dimension N is chosen for the best validation MSE. Notice that the full-capacity uRNN achieves the best detail in its predictions, especially in higher-frequency harmonics.

Results are shown in Table 5.2, and Figure 5.2 shows example predictions of the three types of networks. Results in Table 5.2 are given in terms of the mean-squared error (MSE) loss function and several metrics computed on the time-domain signals, which are reconstructed from the predicted log-magnitude and the original phase of the STFT. These time-domain metrics are segmental signal-to-noise ratio (SegSNR), short-time objective intelligibility (STOI), and perceptual evaluation of speech quality (PESQ). SegSNR, computed using [15], uses a voice activity detector to avoid measuring SNR in silent frames. STOI is designed to correlate well with human intelligibility of speech, and takes on values between 0 and 1, with a higher score indicating higher intelligibility [142]. PESQ is the ITU-T standard for telephone voice quality testing [124, 71], and is a popular perceptual quality metric for speech enhancement [97]. PESQ ranges from 1 (bad quality) to 4.5 (no distortion).

Note that full-capacity uRNNs generally perform better than restricted-capacity uRNNs with the same number of parameters, and both types of uRNN significantly outperform LSTMs.

5.3.3 Pixel-by-pixel MNIST

As another challenging long-term memory task with natural data, we test the performance of LSTMs and uRNNs on pixel-by-pixel MNIST and permuted pixel-by-pixel MNIST, first proposed by [85] and used by [3] to test restricted-capacity uRNNs. For permuted pixel-by-pixel MNIST, the pixels are shuffled, thereby creating some non-local dependencies between pixels in an image. Since the MNIST images are 28×28 pixels, resulting pixel-by-pixel sequences are $T = 784$ elements long. An illustration of this task is shown in figure 5.3. We use 5000 of the 60000 training examples as a validation set to perform early stopping with a patience of 5. The loss function is cross-entropy. Weights with the best validation loss are used to process the evaluation set. The full-capacity uRNN uses RMSprop-style gradient normalization.

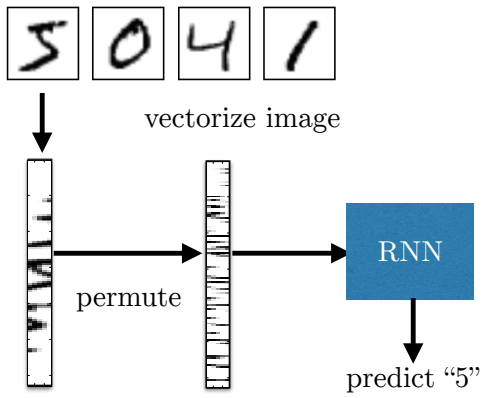

Figure 5.3: Illustration of pixel-by-pixel and permuted pixel-by-pixed MNIST classificaiton task.

Learning curves are shown in Figure 5.4, and a summary of classification accuracies is

Table 5.3: Results for unpermuted and permuted pixel-by-pixel MNIST. Classification accuracies are reported for trained model weights that achieve the best validation loss.

	Model	N	# parameters	Validation accurary	Evaluation accuracy
Unpermuted	LSTM	128	$\approx$ 68k	98.1	97.8
	LSTM	256	$\approx$270k	**98.5**	**98.2**
	Restricted-capacity uRNN	512	$\approx$ 16k	97.9	97.5
	Full-capacity uRNN	116	$\approx$ 16k	92.7	92.8
	Full-capacity uRNN	512	$\approx$270k	97.5	96.9
Permuted	LSTM	128	$\approx$ 68k	91.7	91.3
	LSTM	256	$\approx$270k	92.1	91.7
	Restricted-capacity uRNN	512	$\approx$ 16k	94.2	93.3
	Full-capacity uRNN	116	$\approx$ 16k	92.2	92.1
	Full-capacity uRNN	512	$\approx$270k	**94.7**	**94.1**

shown in Table 5.3. For the unpermuted task, the LSTM with $N = 256$ achieves the best evaluation accuracy of 98.2%. For the permuted task, the full-capacity uRNN with $N = 512$ achieves the best evaluation accuracy of 94.1%, which is state-of-the-art on this task. Both uRNNs outperform LSTMs on the permuted case, achieving their best performance after fewer traing epochs and using an equal or lesser number of trainable parameters. This performance difference suggests that LSTMs are only able to model local dependencies, while uRNNs have superior long-term memory capabilities. Despite not representing all unitary matrices, the restricted-capacity uRNN with $N = 512$ still achieves impressive test accuracy of 93.3% with only 1/16 of the trainable parameters, outperforming the full-capacity uRNN with $N = 116$ that matches number of parameters. This result suggests that further exploration into the potential trade-off between hidden state dimension N and capacity of unitary parameterizations is necessary.

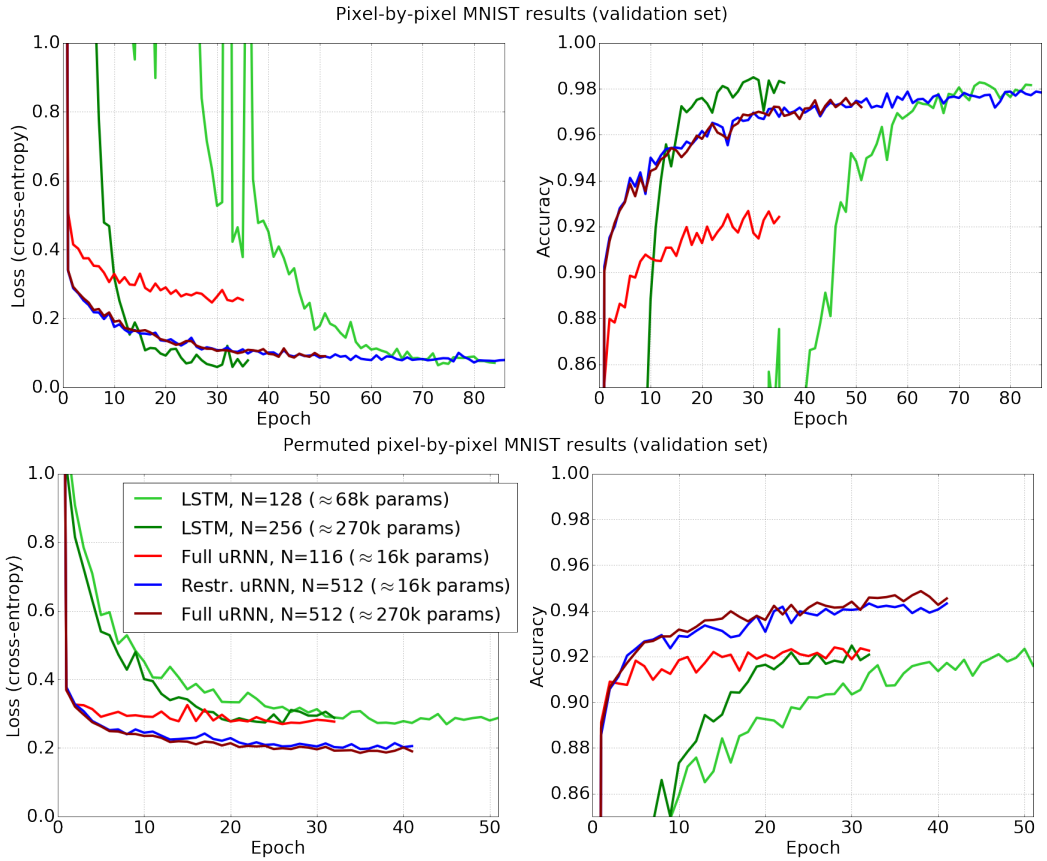

Figure 5.4: Learning curves for unpermuted pixel-by-pixel MNIST (top panel) and permuted pixel-by-pixel MNIST (bottom panel).

5.4 Conclusion

Unitary recurrent matrices prove to be an effective means of addressing the vanishing and exploding gradient problems. Section 5.1 provided a theoretical argument to quantify the capacity of constrained unitary matrices. Then, section 5.2 described a method for directly optimizing a full-capacity unitary matrix by constraining the gradient to lie in the differentiable manifold of unitary matrices. The effect of restricting the capacity of the unitary weight matrix was tested on system identification and memory tasks, in which full-capacity unitary recurrent neural networks (uRNNs) outperformed restricted-capacity uRNNs from

[3] as well as LSTMs. Full-capacity uRNNs also outperformed restricted-capacity uRNNs on log-magnitude STFT prediction of natural speech signals and classification of permuted pixel-by-pixel images of handwritten digits, and both types of uRNN significantly outperformed LSTMs.

Thus, this chapter has improved a state-of-the-art RNN designed to process nonstationary signals by reconsidering conventional model assumptions. In constrast to chapters 3 and 4, this assumption is not of a statistical nature at first glance. However, we will see in the following chapters that the uRNN does have a statistical model-based explanation as deep-unfolded sequential sparse coding. In this explanation, the unitary recurrence matrix constraint manifests as a particular constraint on the possible sparse coding dictionaries that are used in the model. Thus, by using full-capacity optimization, we are effectively expanding the set of possible sparse coding dictionaries that can be used such that vanishing/exploding gradient difficulties are prevented during training of the RNN.

Chapter 6

DEEP UNFOLDING OF STATISTICAL MODELS

This chapter describes deep unfolding in detail. First, some background and related work is discussed. Then the formulation of deep unfolding is provided, as well as a summary of the deep unfolding conversions that will be given as examples. The remainder of the chapter describes specific examples of deep unfolding. Several of these unfolding conversions are novel contributions and specifically designed to process nonstationary signals.

6.1 Background

"Deep unfolding" was originally coined by Hershey et al. [65], and refers to implementing the iterations of statistical model inference algorithms as differentiable computational graphs, or unfolded networks. These unfolded networks can be discriminatively trained on a supervised dataset using backpropagation, just as deep neural networks are trained (this procedure is described in section 2.5.1). But in contrast to deep neural networks, which use randomly-initialized layers whose combinations are refined through heuristic trial-and-error, the architecture of unfolded networks is inspired by model-based reasoning. This statistical model foundation provides principled initializations for the layers of an unfolded network, which can speed up training and lead to better performance of the network compared to using random weight initialization. Once the weights of an unfolded network are initialized using the statistical model, these weights can be untied, or uncoupled across layers during discriminative training. Untying the weights of unfolded networks is often advantageous, increasing the flexibility and capacity of unfolded networks to leverage information contained in the supervised training data.

Other attempts have been made to combine deep networks with generative models. For

example, Varani et al. [148] proposed a DNN where the last layer is a GMM. The GMM parameters are discriminatively trained jointly with the DNN parameters for ASR. Hoshen et al.'s approach [68] attempts to mimic the usual feature extraction pipeline in ASR. Kingma and Welling [81] proposed the variational autoencoder (VAE). A VAE includes stochastic nodes that sample from simple distributions, such as a spherical Gaussian, then runs these sampled values through a deep neural network in an attempt to match the distribution of the input data. The network is trained using a variational lower bound on the likelihood of the data. However, all these methods suffer the same drawback: the optimal network architectures can only be discovered by heuristic experimentation, and it is difficult to directly incorporate insight from domain knowledge.

Several iterative algorithms for probabilistic model inference have been unfolded, including nonnegative matrix factorization for audio source separation [86], and supervised topic modeling [18, 88]. Sparse recovery algorithms have also been unfolded, but only for the nonsequential case. In fact, one of the earliest examples of unfolding is by Gregor and LeCun [60], who proposed learned ISTA (LISTA). LISTA uses learned encoders and decoders to increase the speed and performance of the original ISTA algorithm. Rolfe and LeCun [126] unfolded ISTA under a nonnegativity constraint on the sparse coefficients. In this case, the nonlinear units of the unfolded network are ReLUs [107]. After adding a classification penalty term to the training cost function, Rolfe and LeCun dubbed the resulting network a discriminative recurrent[1] sparse autoencoder. These networks are similar to the deep sparse rectifier networks of Glorot et al. [57], except that the coding dictionaries are tied between layers. Sprechmann et al. [137] learned better optimization algorithms for parsimonious regularizations. Borgerding et al. [14] unfolded the approximate message passing (AMP) algorithm for sparse coding and observed improved performance compared to learned ISTA and FISTA. Kamilov and Mansour [76, 78] learned optimal nonlinear thresholding functions for ISTA and FISTA. Kamilov et al. [77] learned improved proximal filters for unfolded

[1]Note that the the recurrence described by Rolfe and LeCun is across *iterations* instead of across time steps. We use "iterations" to refer to the vertical stacking dimension in our model.

FISTA. We go beyond these works by considering sequential sparse coding and exploiting time-recurrent connections between sequential optimization iterations. Palangi et al. [118] proposed convolutional deep stacking networks to improve sparse recovery from multiple measurement vectors. This dissertation goes beyond these works by considering unfolding the extension of sparse coding to sequential data.

6.2 Formulation of Deep Unfolding

Given a statistical model with parameters θ and hidden variables ϕ, deep unfolding replaces the alternating minimization problems in (2.10) and (2.11) with one optimization problem, where the inference steps h_θ, which are designed replace the update (2.10) with inference of the hidden variables, become part of the training problem (2.22):

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_i f\left(x_i, y_i, \theta, \hat{\phi}_i\right) \\ \text{subject to} \quad & \hat{\phi}_i = h_\theta(x_i),\ i = 1, \ldots, N. \end{aligned} \tag{6.1}$$

Note that the loss in (6.1) is minimized and denoted as f instead of being maximized and equal to the log-likelihood as in (2.9). This is done to emphasize that the loss f used to train an unfolded network does not necessarily need to be equal to the log-likelihood, as in (2.9).

Creating networks from statistical model inference is much more straightforward than going from a network to a model. Going the opposite direction–starting from a network architecture and finding its underlying model–must be done by comparing the network's computational structures to strucures that appear in certain model's inference algorithms. In this chapter, we will see instances of both model-to-network and network-to-model conversions. An exhaustive list of these conversions is given in table 6.1.

6.3 Examples

In this section, specific examples of deep unfolding are described. We begin with feedforward examples, which are used as a foundation for recurrent examples.

Table 6.1: Comprehensive list of deep-unfolding conversions described in this dissertation. Conversions without references are novel contributions of this dissertation.

Model	**Algorithm**	**Direction**	**Network**	**Section**	**Reference**
Binary MRF $\{0, 1\}$	Mean-field	$\rightarrow$	FFNN using sigm (2.27) with input connections	6.3.1	[65]
Binary MRF $\{-1, 1\}$	Mean-field	$\rightarrow$	FFNN using tanh (2.28) with input connections	6.3.1	[65]
Nonnegative sparse coding	ISTA	$\rightarrow$	FFNN using ReLU with input connections	6.3.2	[60]
Sparse coding	ISTA	$\rightarrow$	FFNN using soft (2.19) with input connections	6.3.2	[60]
Sequential sparse coding	SISTA	$\rightarrow$	SISTA-RNN	6.3.3	
Nonnegative sequential sparse coding	SISTA	$\rightarrow$	DR-NMF	7.2.3	
Multichannel GMM	Variational EM	$\rightarrow$	Deep multichannel GMM	7.3.2	
Multiple MRFs	Mean-field	$\leftarrow$	GRU RNN using tanh	6.3.4	
Multiple MRFs and (nonnegative) sparse coding	Mean-field and SISTA	$\leftarrow$	GRU RNN using ReLU	6.3.4	
Sequential sparse coding with complex coefficients	SISTA	$\leftarrow$	Unitary RNN	6.3.5	

6.3.1 Mean Field Inference in Markov Random Fields to Feedforward Sigmoid Networks

This section provides an example of a model-to-network unfolding, where an inference algorithm for Markov random fields becomes a feedforward sigmoid or hyperbolic tangent network. This correspondence was originally described by Hershey et al. [65], and this section is a review of their work.

First, this section describes mean field inference in Markov random fields (MRFs). Then, we show how this inference algorithm unfolds to a deep feedforward sigmoid network. We will restrict our attention to pairwise MRFs, which have N_u hidden binary variables u_i, $i = 1..N_u$ and N_v observed continuous variables v_ℓ, $\ell = 1..N_v$. The binary variables can take on values in $\{0, 1\}$ or $\{-1, 1\}$. The posterior probability of the hidden variables is given by

$$p(\mathbf{u}|\mathbf{v}) \propto \exp\Big(\sum_{(i,j)\in\mathcal{E}_{uu}} \mathbf{\Psi}(u_i, u_j) + \sum_{(i,\ell)\in\mathcal{E}_{uv}} \mathbf{\Psi}(u_i, v_\ell)\Big), \tag{6.2}$$

where $\mathcal{E}_{uu}$ are the edges between the hidden nodes, $\mathcal{E}_{uv}$ are the edges between hidden and visible nodes, and $\mathbf{\Psi}$ are pairwise log-potential functions. The posterior (6.2) can also be written [65] as

$$p(\mathbf{u}|\mathbf{v}) \propto \exp\Big(\frac{1}{2}\mathbf{u}^\mathsf{T}\mathbf{A}\mathbf{u} + \mathbf{u}^\mathsf{T}\mathbf{c} + \mathbf{u}^\mathsf{T}\mathbf{B}\mathbf{v}\Big), \tag{6.3}$$

where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{c}$ are derived from the log-potentials $\mathbf{\Psi}$. See [65, Appendix A] for an example of this derivation when the hidden and visible variables are binary. Extension of this derivation to continuous visible variables, as we use here, is straightforward.

Exact inference in this model is intractable. But approximate inference is possible using the variational mean field (MF) approximation. MF infers the posterior probabilities of the hidden variables u_i by minimizing the Kullback-Leibler (KL) divergence between an approximate posterior $q(\mathbf{u})$ and the true posterior $p(\mathbf{u}|\mathbf{v})$, which is an upper bound on the negative log-likelihood of the observed data $\mathbf{v}$:

$$\hat{\mathbf{q}}_\mathbf{u} = \underset{\mathbf{q}_\mathbf{u}}{\operatorname{argmin}}\, D_{KL}\big(q(\mathbf{u}) \,||\, p(\mathbf{u}|\mathbf{v})\big) = -\sum_{\mathbf{u}} q(\mathbf{u}) \log \frac{p(\mathbf{u}, \mathbf{v})}{q(\mathbf{u})} \geq -\log\left(\sum_{\mathbf{u}} p(\mathbf{u}, \mathbf{v})\right) = -\log p(\mathbf{v}). \tag{6.4}$$

This procedure assumes that the approximate posterior $q(\mathbf{u})$ factors across the hidden nodes: $q(\mathbf{u}) = \prod_{i=1}^{N_u} q_{u_i}$. Note that we use the notation $q(\mathbf{u}) \in [0, 1]$ to denote the scalar approximate posterior probability and $\mathbf{q_u} \in [0, 1]^{N_u}$ for the vector consisting of the individual approximate probabilities q_{u_i}, $i = 1..N_u$.

Solving (6.4) yields the following iterative update for the expected values of the hidden variables, where k indexes iterations:

$$\hat{\mathbf{q}}_{\mathbf{u}}^{(k)} = \sigma\Big(\mathbf{A}\hat{\mathbf{q}}_{\mathbf{u}}^{(k-1)} + \mathbf{B}\mathbf{v} + \mathbf{c}\Big), \tag{6.5}$$

where σ is a sigmoid function. Since the elements of $\mathbf{u}$ are Bernoulli random variables that can only take on the values 0 or 1, $\mathbf{u} \in \{0, 1\}^{N_u}$, then after K iterations the expected value of $\mathbf{u}$ can be computed as $E_{\hat{q}}\{u_i\} = \hat{q}_{u_i}^{(K)}$. If the elements of $\mathbf{u}$ can only take on values -1 or 1, $\mathbf{u} \in \{-1, 1\}^{N_u}$, then $E_{\hat{q}}\{u_i\} = 2\hat{q}_{u_i}^{(K)} - 1$, which is equal to (6.5) where the sigmoid function σ (2.27) is replaced with a hyperbolic tangent function (2.28).

Notice the similarity of (6.5) to the forward pass of a feedforward neural network (2.26), with one significant difference: only the first layer of the feedforward neural network is connected to the input. In contrast, the unfolded MF sigmoid or hyperbolic tangent networks (6.5) are connected to the input. Note that if we instead assume that each layer has its own MRF and that the layer is performing a single iteration of mean field inference, the unfolded MF network is equivalent to a feedforward neural network with a sigmoid or hyperbolic tangent activation.

6.3.2 Sparse Coding to Feedforward Soft-Thresholding Networks

For the nonsequential sparse coding problem (2.16), we described ISTA in section 2.4. Gregor and LeCun [60] were the first to describe the correspondence between ISTA and a feedforward ReLU network.

Notice the similarity of algorithm 1 in section 2.4 to the forward pass of a feedforward neural network (2.26) with a ReLU activation (2.29), again with one significant difference: only the first layer of the feedforward neural network is connected to the input. In contrast,

the forward pass of the unfolded ISTA networks given by algorithm 1 are connected to the input.

Algorithm 2 Sequential iterative soft-thresholding algorithm (SISTA)

Input: observation sequence $\mathbf{x}_{1:T}$, measurement matrix $\mathbf{A}$ dictionary $\mathbf{D}$, predictor $\mathbf{F}$, initial coefficients $\hat{\mathbf{h}}_0$

for $t = 1$ to T **do**
  $\mathbf{h}_t^{(0)} \leftarrow \mathbf{D}^{-1}\mathbf{F}\mathbf{D}\hat{\mathbf{h}}_{t-1}$ *# Initial estimate for* $\mathbf{h}_t$
  for $k = 1$ to K **do**
    $\mathbf{z} \leftarrow \big[\mathbf{I} - \frac{1}{\alpha}\mathbf{D}^\top(\mathbf{A}^\top\mathbf{A} + \lambda_2\mathbf{I})\mathbf{D}\big]\mathbf{h}_t^{(k-1)} + \frac{1}{\alpha}\mathbf{D}^\top\mathbf{A}^\top\mathbf{x}_t$
    $\mathbf{h}_t^{(k)} \leftarrow \mathrm{soft}_{\lambda_1/\alpha}\big(\mathbf{z} + \frac{\lambda_2}{\alpha}\mathbf{D}^{-1}\mathbf{F}\mathbf{D}\hat{\mathbf{h}}_{t-1}\big)$
  $\hat{\mathbf{h}}_t \leftarrow \mathbf{h}_t^{(K)}$ *# Assign estimate for* $\mathbf{h}_t$
return $\hat{\mathbf{h}}_{1:T}$

6.3.3 Sequential Sparse Coding to Recurrent Soft-Thresholding Networks

Now assume we want to model a sequence of observations $\mathbf{x}_t$, $t = 1..T$, where these sequential observations are not necessarily independent. When we are processing nonstationary signals, this case is of particular interest. To model this dependence over time, we will assume that the sparse coefficient vector $\mathbf{h}_t$ is correlated with the previous coefficient vector $\mathbf{h}_{t-1}$ such that the signal $\mathbf{s}_t = \mathbf{D}\mathbf{h}_t$ is linearly predictable from $\mathbf{s}_{t-1}$: $\mathbf{s}_t = \mathbf{F}\mathbf{s}_{t-1} + \mathbf{v}_t$, where $\mathbf{v}_t$ is zero-mean Gaussian noise representing the prediction error. The probabilistic model for this formulation uses a different prior on $\mathbf{h}_t$ that is conditioned on $\mathbf{h}_{t-1}$:

$$\begin{aligned} \mathbf{x}_t &\sim \mathcal{N}(\mathbf{A}\mathbf{D}\mathbf{h}_t, \sigma^2\mathbf{I}), \\ p\big(\mathbf{h}_t \,|\, \mathbf{h}_{t-1}\big) &\propto \exp\Big\{-\nu_1\|\mathbf{h}_t\|_1 - \frac{\nu_2}{2}\|\mathbf{D}\mathbf{h}_t - \mathbf{F}\mathbf{D}\mathbf{h}_{t-1}\|_2^2\Big\}. \end{aligned} \tag{6.6}$$

This prior encourages $\mathbf{h}_t$ to be sparse while enforcing correlation between $\mathbf{s}_t = \mathbf{D}\mathbf{h}_t$ and $\mathbf{s}_{t-1} = \mathbf{D}\mathbf{h}_{t-1}$ through the matrix $\mathbf{F}$. The prior on $\mathbf{h}_t$ in (6.6) is similar to the prior for

elastic net regularization in the nonsequential case, where $\mathbf{h}$ is penalized by both ℓ_1 and ℓ_2 norms [168]. For nonsequential elastic net, the prior can be shown to be a Gaussian scale mixture [92].

Minimizing the joint negative log-likelihood of $\mathbf{x}_{1:T}$ and $\mathbf{h}_{1:T}$ under the generative model (6.6) is equivalent to solving the optimization problem

$$\min_{\mathbf{h}_{1:T}} \quad \sum_{t=1}^{\mathsf{T}} \Big(\frac{1}{2} \big\| \mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t \big\|_2^2 + \lambda_1 \big\| \mathbf{h}_t \big\|_1 + \frac{\lambda_2}{2} \| \mathbf{D}\mathbf{h}_t - \mathbf{F}\mathbf{D}\mathbf{h}_{t-1} \|_2^2 \Big), \tag{6.7}$$

with $\lambda_1 = 2\sigma^2\nu_1$ and $\lambda_2 = 2\sigma^2\nu_2$.

We dub the iterative algorithm for solving the optimization problem (6.7) sequential ISTA (SISTA), which is described in algorithm 2 and derived in appendix A. Note that algorithm 2 is a straightforward modification of algorithm 1, where the modifications are an outer loop over time step t, the transform $\mathbf{I} - \frac{1}{\alpha}\mathbf{D}^\mathsf{T}(\mathbf{A}^\mathsf{T}\mathbf{A} + \lambda_2\mathbf{I})\mathbf{D}$ instead of $\mathbf{I} - \frac{1}{\alpha}\mathbf{D}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{D}$ in line 4, and the the extra additive term $\frac{\lambda_2}{\alpha}\mathbf{D}^{-1}\mathbf{F}\mathbf{D}\hat{\mathbf{h}}_{t-1}$ in line 5 before the application of soft-thresholding.

The estimate of the previous state $\hat{\mathbf{h}}_{t-1}$ in lines 5 and 6 of algorithm 2 is determined by the order of the iterative updates. We desire a low latency, which means a signal estimate $\hat{\mathbf{s}}_t$ is computed as soon as the data $\mathbf{x}_t$ is provided. Given only data for $t = 1$, the known part of the objective in (6.7) is a convex function of $\mathbf{h}_1$. Thus, the output $\mathbf{h}_1^{(K)}$ after K iterations is the best estimate of the global optimum given data up to $t = 1$. The optimal estimate of the next hidden state $\mathbf{h}_2$ should then use $\hat{\mathbf{h}}_1 = \mathbf{h}_1^{(K)}$ as part of the estimation of the next time step $t = 2$. Applying this logic across time, the optimal choice for minimum latency is $\hat{\mathbf{h}}_{t-1} = \mathbf{h}_{t-1}^{(K)}$. Also, the best initialization $\mathbf{h}_t^{(0)}$ for $\mathbf{h}_t$ is the linear prediction $\mathbf{D}^{-1}\mathbf{F}\mathbf{D}\hat{\mathbf{h}}_{t-1}$.

Notice that if the RNN nonlinearity σ in (2.32) is set to the soft-thresholding operation (2.19) with bias $b_n = (\lambda_1/\alpha)$ for $n = 1..N$, the forward RNN computation in (2.32) and (2.33) corresponds to the SISTA algorithm described in algorithm 2 under the following conditions, which are also illustrated in the right panel of figure 6.1:

1. The input nodes $\mathbf{x}_{1:T}$ are connected to every hidden node $\mathbf{h}_{1:T}^{(1:K)}$ using the matrices $\mathbf{V}^{(1:K)}$.

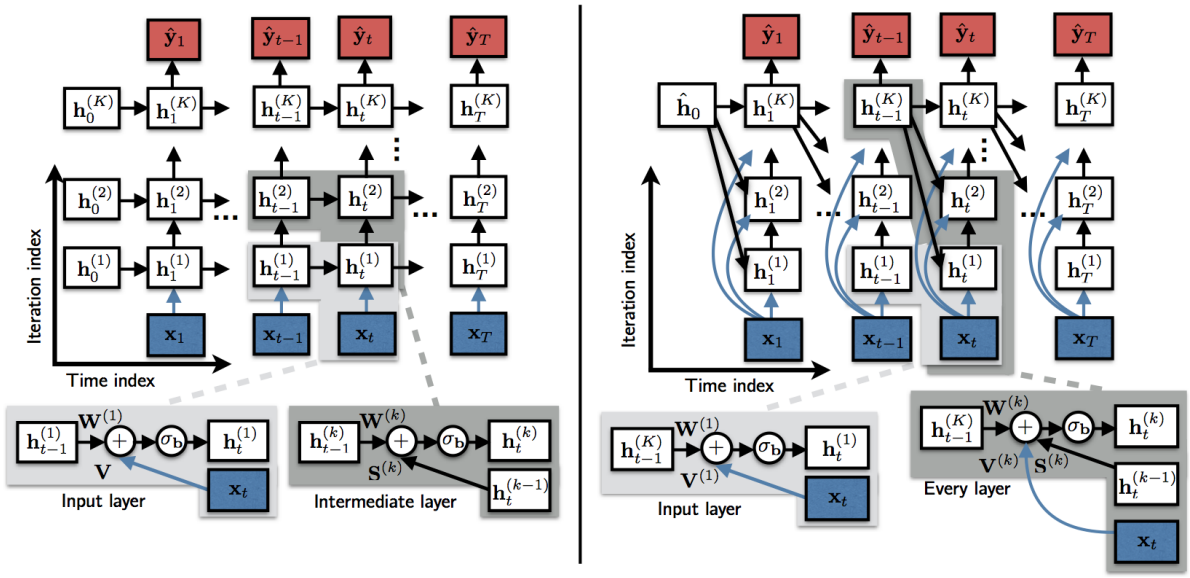

Figure 6.1: Comparison between standard stacked RNN architecture (left) as described by equations (2.32) and (2.33) and unfolded SISTA RNN (right) as implemented in algorithm 2 under the conditions described in section 6.3.3. Colored nodes are inputs and outputs and white nodes are hidden states (i.e. estimates of sparse recovery coefficients). Reprinted from [158], ©2017 IEEE.

2. The previous state estimate $\hat{\mathbf{h}}_{t-1} = \mathbf{h}_{t-1}^{(K)}$ is used instead of $\hat{\mathbf{h}}_{t-1} = \mathbf{h}_{t-1}^{(k)}$ in the standard RNN.

3. The RNN parameters are constrained as follows, with $\mathbf{P} = \mathbf{D}^{-1}\mathbf{F}\mathbf{D}$:

$$\mathbf{V}^{(k)} = \frac{1}{\alpha}\mathbf{D}^\mathsf{T}\mathbf{A}^\mathsf{T}, \quad \forall k, \tag{6.8}$$

$$\mathbf{S}^{(k)} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^\mathsf{T}(\mathbf{A}^\mathsf{T}\mathbf{A} + \lambda_2\mathbf{I})\mathbf{D}, \quad k > 1, \tag{6.9}$$

$$\mathbf{U}^{(1)} = \frac{\alpha + \lambda_2}{\alpha}\mathbf{P} - \frac{1}{\alpha}\mathbf{D}^\mathsf{T}(\mathbf{A}^\mathsf{T}\mathbf{A} + \lambda_2\mathbf{I})\mathbf{D}\mathbf{P}, \tag{6.10}$$

$$\mathbf{U}^{(k)} = \frac{\lambda_2}{\alpha}\mathbf{P}, \quad k > 1, \tag{6.11}$$

$$\mathbf{Z} = \mathbf{D}, \quad \mathbf{c} = \mathbf{0}. \tag{6.12}$$

Under these conditions, the SISTA RNN output $\hat{\mathbf{y}}_{1:T}$ is equivalent to the reconstructed signal

$\hat{\mathbf{s}}_{1:T} = \mathbf{D}\hat{\mathbf{h}}_{1:T}$ from the original SISTA. Training the layer-wise SISTA RNN parameters

$$\boldsymbol{\theta}_{\text{SISTA}} = \{\hat{\mathbf{h}}_0, \mathbf{A}^{(1:K)}, \mathbf{D}^{(1:K)}, \mathbf{F}^{(1:K)}, \alpha^{(1:K)}, \lambda_1^{(1:K)}, \lambda_2^{(1:K)}\}, \tag{6.13}$$

corresponds to optimizing decoupled functions of the original SISTA parameters $\{\mathbf{A}, \mathbf{D}, \mathbf{F}, \alpha, \lambda_1, \lambda_2\}$. Thus, training learns generalized settings of the SISTA parameters that improve performance of deterministic SISTA with respect to a cost function on training data.

6.3.4 Model-Based Explanation of Gated Recurrent Unit (GRU) Recurrent Neural Networks

Now we turn to explaining existing neural network architectures using model-based thinking, which is a case of "reverse unfolding." The first example that we will examine is the GRU RNN as described in section 2.5.5, which is a popular RNN architecture.

Notice that the form of the equation for the reset vector (2.41) and the update vector (2.42) are identical to a single iteration of the mean field update (6.5) for binary random variables $\boldsymbol{\rho}_t \in \{0,1\}^N$ and $\boldsymbol{\zeta}_t \in \{0,1\}^N$ under the following conditions: the initial distributions are $\hat{\mathbf{q}}_{\boldsymbol{\rho}_t}^{(0)} = \mathbf{0}$ and $\hat{\mathbf{q}}_{\boldsymbol{\zeta}_t}^{(0)} = \mathbf{0}$, the visible variables are $\mathbf{v} = [\mathbf{x}_t; \mathbf{h}_{t-1}]$, the matrices $\mathbf{B}_r$ and $\mathbf{B}_z$ have block forms of

$$\mathbf{B}_r = \begin{bmatrix} \mathbf{U}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_r \end{bmatrix} \qquad \mathbf{B}_z = \begin{bmatrix} \mathbf{U}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_z \end{bmatrix}, \tag{6.14}$$

and $\mathbf{c}_r = \mathbf{b}_r$ and $\mathbf{c}_z = \mathbf{b}_z$. Thus, we can view $\mathbf{r}_t$ and $\mathbf{z}_t$ as the posterior probabilities for binary hidden random variables $\boldsymbol{\rho}_t$ and $\boldsymbol{\zeta}_t$ in a fully connected MRF where the observed data are $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$, and the GRU is performing a single iteration of MF inference within a time step to compute $\mathbf{r}_t = \hat{\mathbf{q}}_{\boldsymbol{\rho}_t}^{(1)}$ and $\mathbf{z}_t = \hat{\mathbf{q}}_{\boldsymbol{\zeta}_t}^{(1)}$. The hidden state $\mathbf{h}_t$ is the expected value of a hidden random vector $\boldsymbol{\eta}_t$.

This correspondence leads to a statistical model interpretation for the reset and update vectors. The ith element of the reset vector $\boldsymbol{\rho}_t$ models the change in support of the ith element of the hidden vector $\boldsymbol{\eta}_t$ from time step $t-1$ to time step t:

$$\begin{aligned} \rho_{t,i} = 0 &: \eta_{t,i}^{(0)} = \eta_{t-1,i}, \\ \rho_{t,i} = 1 &: \eta_{t,i}^{(0)} = 0. \end{aligned} \tag{6.15}$$

The ith element of the update variable $\boldsymbol{\zeta}_t$ determines the distribution of the ith element of $\boldsymbol{\eta}_t$:

$$\begin{aligned} \zeta_{t,i} = 0: &\quad \eta_{t,i} \sim \mathcal{M}_{\boldsymbol{\theta}}(\mathbf{x}_t), \\ \zeta_{t,i} = 1: &\quad \eta_{t,i} = \eta_{t-1,i}, \end{aligned} \tag{6.16}$$

where $\mathcal{M}_{\boldsymbol{\theta}}(\mathbf{x}_t)$ is a probabilistic model with parameters $\boldsymbol{\theta}$ and observed variables $\mathbf{x}_t$. Thus, (2.40) is marginalizing over $\boldsymbol{\zeta}_t$ and taking the expected value of $\boldsymbol{\eta}_t$:

$$h_{t,i} = E_{q_{\zeta_{t,i}}}\{E_{\mathcal{M}}\{\eta_{t,i}|\zeta_{t,i}\}\} = q_{\zeta_{t,i}} \cdot E_{\mathcal{M}}\{h_{t,i}|\zeta_{t,i}=1\} + (1 - q_{\zeta_{t,i}}) \cdot E_{\mathcal{M}}\{h_{t,i}|\zeta_{t,i}=0\}. \tag{6.17}$$

The conditional likelihood of the hidden variables for this model is

$$p(\boldsymbol{\eta}_{1:T}, \boldsymbol{\rho}_{1:T}, \boldsymbol{\zeta}_{1:T}|\mathbf{x}_{1:T}) \propto \underbrace{p(\boldsymbol{\eta}_0)}_{\text{Prior on } \mathbf{h}_0} \prod_{t=1}^{\mathsf{T}} \underbrace{p(\boldsymbol{\eta}_t|\boldsymbol{\eta}_{t-1}, \mathbf{x}_t, \boldsymbol{\rho}_t, \boldsymbol{\zeta}_t)}_{\text{Transition for } \boldsymbol{\eta}_t \text{ from } t-1} \underbrace{p(\boldsymbol{\rho}_t|\boldsymbol{\eta}_{t-1}, \mathbf{x}_t)}_{\text{MRF for reset}} \underbrace{p(\boldsymbol{\zeta}_t|\boldsymbol{\eta}_{t-1}, \mathbf{x}_t)}_{\text{MRF for update}}. \tag{6.18}$$

The model $\mathcal{M}$ determines the activation ϕ and parameters $\boldsymbol{\theta}$. We now describe two notable examples.

Markov random field

If $\mathcal{M}$ is a pairwise MRF where $\boldsymbol{\eta}_t \in \{-1, 1\}^N$, then ϕ is a hyperbolic tangent (as in the original GRU paper [21]). In this case, the second term in (2.40) corresponds to a single MF inference update (6.5) for $\boldsymbol{\eta}_t$. In this case, the binary variables $\boldsymbol{\rho}$ model the change in support from time step $t-1$ to time step t, which is governed by a MRF with parameters $\{\mathbf{A}_r, \mathbf{B}_r, \mathbf{c}_r\}$. The parameters of the model $\mathcal{M}$ in this case are $\boldsymbol{\theta} = \{\mathbf{A}, \mathbf{B}, \mathbf{c}\}$, which are functions of the log-potentials of a pairwise MRF between binary hidden nodes $\mathbf{h}_t$ and continuous observed data $\mathbf{v}_t = [\mathbf{x}; \mathbf{h}_{t-1}]$ for each time step.

Sparse coding

As described in section 6.3.2, if ϕ is a ReLU, then the second term in (2.40) can be seen as one iteration of ISTA for nonnegative sparse coding. In this case, the binary variables $\boldsymbol{\rho}_t$ model the change in support from time step $t-1$ to time step t, which is governed by a

MRF with parameters $\{\mathbf{A}_r, \mathbf{B}_r, \mathbf{c}_r\}$. The parameters of the model are $\boldsymbol{\theta} = \{\mathbf{D}, \alpha, \lambda_1\}$, and the network parameters $\{\mathbf{V}, \mathbf{U}, \mathbf{b}\}$ can be written as a function of the ISTA parameters $\boldsymbol{\theta}$ as follows:

$$\mathbf{V} = \frac{1}{\alpha}\mathbf{D}^{\mathsf{T}}, \qquad \mathbf{U} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathsf{T}}\mathbf{D}, \qquad \mathbf{b} = -\frac{\lambda_1}{\alpha}\mathbf{1}. \tag{6.19}$$

If the sparse hidden states $\boldsymbol{\eta}_t$ do not have a nonnegativity constraint, then ϕ is a soft-thresholding function (2.19), and the second term in (2.40) can also be seen as one iteration of ISTA with the same parameters as the nonnegative case.

6.3.5 Model-Based Explanation of Unitary Recurrent Neural Networks

Since unitary RNNs use a soft-thresholding nonlinearity, they can be thought of as equivalent to a SISTA-RNN with a single iteration, $K = 1$, at each time step and no Gaussian prediction model between adjacent time steps. Furthermore, the unitary constraint on the recurrence matrix places a constraint on the dictionary $\mathbf{D}$ that we derive in this section.

For unitary RNNs, using (6.10) with $\mathbf{P} = \mathbf{I}$ (i.e., warm-start of $\mathbf{h}_t$ estimation with $\hat{\mathbf{h}}_{t-1}$), $\lambda_2 = 0$, $\mathbf{A} = \mathbf{I}$, and replacing transposes with Hermitian transposes, we can write the complex-valued recurrence matrix $\mathbf{U}$ as

$$\mathbf{U} = \mathbf{I} - \frac{1}{\alpha}\mathbf{D}^{\mathsf{H}}\mathbf{D} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^{\mathsf{H}} \tag{6.20}$$

with $\boldsymbol{\Sigma} = \mathbf{I} - \frac{1}{\alpha}\text{diag}\boldsymbol{\lambda}$, where $\boldsymbol{\lambda}$ are the eigenvalues and $\mathbf{V}$ the eigenvectors of $\mathbf{D}^H\mathbf{D}$. If the recurrent matrix $\mathbf{U}$ is unitary, using the definition of unitary matrices then the eigenvalues λ_i must satisfy

$$\left|1 - \frac{\lambda_i}{\alpha}\right| = 1 \Rightarrow \lambda_i = 0 \text{ or } 2\alpha. \tag{6.21}$$

Thus, $\mathbf{D} \in \mathbb{C}^{F \times N}$, with no restrictions except that the nonzero singular values of $\mathbf{D}$ are equal to either 0 or $\sqrt{2\alpha}$. Thus, a uRNN is equivalent to a SISTA-RNN with one iteration per time step using a constrained complex-valued dictionary $\mathbf{D}$.

6.4 Conclusion

In this chapter, the formulation of the deep unfolding framework was provided and relevant literature was summarized. Then, we explored multiple examples of deep unfolding. These examples included instances of model-to-network unfolding as well as instances of network-to-model unfolding. Model-to-network unfolding creates new types of architectures inspired by statistical model inference algorithms. In particular, we reviewed how mean field inference in a Markov random field unfolds to a feedforward sigmoid or hyperbolic tangent network with input connections and that ISTA for sparse coding unfolds to a feedforward ReLU or soft-thresholding network with input connections. Then, we generalized these existing results to sequential data, and we showed how a sequential version of ISTA unfolds to a new type of recurrent neural network, the SISTA-RNN. In the next chapter, we will modify the SISTA-RNN with nonnegativity constraints and different recurrence to unfold ISTA for sparse NMF.

We also saw how network-to-model unfolding can provide principled model-based explanations for existing successful DNN architectures. In particular, we saw how GRU RNNs can be viewed as performing single iterations of mean field inference for multiple MRFs, optionally combined with a sparse coding model. We also saw that unitary RNNs can be seen as performing a single iteration of ISTA at each time step in a sparse coding model with complex-valued coefficients, where the sparse coding dictionary has constraints on its singular values.

In the next chapter, these architectures will be used and modified for processing of realistic nonstationary signals.

Chapter 7

APPLICATIONS OF DEEP UNFOLDING TO NONSTATIONARY SIGNALS

This chapter describes several applications of deep unfolding to statistical signal processing of nonstationary signals. First, a synthetic task of sequential compressed sensing of images from columnwise observations will be presented. Despite being a synthetic example, this case is a good analogue to processing sequential frames of a spectrogram. Furthermore, the statistical model is simple and well-understood, which means that insight can be easily gained by inspecting what the unfolded network learns.

Next, applications to audio are described. These include separation of speech from both single and multichannel noisy and reverberant recordings, as well as classification of audio. The single-channel speech separation network is based on the widely-used sparse nonnegative matrix factorization (SNMF) method. We will see that unfolding the SISTA algorithm to solve the SNMF problem results in a nonnegative recurrent network, referred to as deep recurrent NMF (DR-NMF).

Multichannel speech separation unfolds a variational inference algorithm for a model where speech spectra are modeled using Gaussian mixture models and the complex-valued frequency-domain channel matrices are explicitly modeled. The resulting network directly processes complex-valued frequency domain multichannel observations, and produces estimates of the sources' complex-valued STFTs. This simple unfolded network lends itself to many extensions that we propose for future work.

For audio classification, the DR-NMF network is also used. Here, we can make use of one of the advantage of deep unfolding: the parameters of the model (i.e., the weights of the network) can be tuned to any particular task. In this case, by using a training loss function

that minimizes classification error, the DR-NMF network can be easily leveraged to perform classification instead of enhancement.

7.1 Sequential Compressed Sensing of Images

To demonstrate the advantage of training a SISTA-RNN, we use a similar experimental setup as Asif and Romberg [5, §V.B]. In this setup, the time-varying signals signal vectors $\mathbf{s}_t$ of dimension $N = 128$ are the columns of $N \times N$ grayscale images. Thus, the 'time' dimension is actually column index, and all sequences are length $T = 128$. The images are taken from the Caltech-256 dataset [61], which consists of 30607 color images of varying sizes. We convert the images to grayscale, clip out centered square regions, and resize to 128×128 using bicubic interpolation. We randomly designate 80% of the data, or 24485 images, for training. The remaining 20% is split into validation and test sets of 3061 images each.

The columns of each image are observed through a $M \times N$ measurement matrix $\mathbf{A}$ with $M = 32$ for a compression factor of 4, with values chosen randomly with equal probability from[1] $\pm 1/(3\sqrt{M})$. The observations are these compressed measurements: $\mathbf{x}_t = \mathbf{A}\mathbf{s}_t$. Since we do not expect the columns of natural images to change very much from column to column, the prediction matrix $\mathbf{F}$ is set to an identity matrix. The dictionary $\mathbf{D}$ consists of Daubechies-8 orthogonal wavelets with four levels of decomposition.

We use unsupervised sparse algorithms as baselines that test relevant assumptions and compare to prior work. These baselines are implemented in Matlab. First, we use SISTA as described in algorithm 2 with fixed step size $\alpha = 1$, $\lambda_1 = 0.02$, and $\lambda_2 = 0.002$. The regularization parameters are chosen using a random hyperparameter search on training data, which sampled $\ell_1, \ell_2 \sim \mathcal{U}(-3, 1)$ and set $\lambda_1 = 10^{\ell_1^*}$ and $\lambda_2 = 10^{\ell_2^*}$ for the best parameters ℓ_1^* and ℓ_2^* in terms of MSE. As another baseline we use SpaRSA[2] [167] for each time step, denoted as sequential SpaRSA (SSpaRSA). SpaRSA is equivalent to ISTA with an adaptive step size

[1]These values are slightly different from Asif and Romberg [5], who use $\pm 1/\sqrt{M}$. We use smaller values of $\pm 1/(3\sqrt{M})$ here so that the norm of the total measurement matrix $\mathbf{AD}$ is less than 1, which means that SISTA will converge with a fixed step size of $\alpha = 1$ [28].

[2]Available from `https://www.lx.it.pt/~mtf/SpaRSA/`.

adjustment and a gradual decrease in λ_1, which allows convergence in fewer iterations.

Since the SISTA-RNN has a fixed number of layers K (i.e., iterations), it is important to test the performance for a fixed K versus a variable K. As such, for both SISTA and SSpaRSA, we either use a fixed number of iterations $K = 3$ or run the algorithms to convergence, where convergence is defined as the relative objective function improvement being less than 10^{-4}. Initialization is also important. We test oracle and non-oracle versions of the baselines, where the oracle version sets $\hat{\mathbf{h}}_0$ to the ground-truth coefficients from the first column of the original image given by $\mathbf{D}^\mathsf{T}\mathbf{s}_0$. Non-oracle versions use $\hat{\mathbf{h}}_0 = \mathbf{0}$. As a state-of-the-art baseline, we also use the ℓ_1-homotopy algorithm run to convergence as described and implemented[3] by Asif and Romberg [5] with oracle initial coefficients and joint optimization of 3 time steps at once.

As a supervised baseline, we train a generic stacked RNN with $K = 3$ as described by (2.32) and (2.33) with a soft-thresholding nonlinearity (2.19). Parameters of this generic RNN are initialized randomly using the suggestion of [56]. For our proposed method, we train a $K = 3$ layer unfolded SISTA-RNN. The parameters of the SISTA-RNN are initialized either randomly [56] or using baseline non-oracle SISTA with fixed $K = 3$ (first row of table 7.1) using the relationships (6.8)-(6.12). Training of all supervised models are implemented in Python using Theano [144]. The training cost function f is MSE between the outputs $\hat{\mathbf{y}}_{1:T}$ and the training references $\mathbf{s}_{1:T}$, which is optimized using backpropagation and stochastic gradient descent with a mini-batch size of 50, an initial learning rate of 10^{-4}, and RMSProp [147] with momentum 0.9 and averaging parameter 0.1 to adapt the learning rate. MSE on the validation set, which is plotted in figure 7.1, is used to determine training convergence. Code to replicate these results is available online[4].

Results are shown in table 7.1 in terms of use of oracle initialization, number of iterations K, number of training examples I, MSE and peak signal-to-noise ratio (PSNR) of the reconstructed signals across the test set. Notice that compared to both the best-performing

[3]Available from `https://github.com/sasif/L1-homotopy`.

[4]`stwisdom.github.io/sista-rnn`

Table 7.1: Results for sequential sparse recovery in terms of oracle initialization, number of iterations K, number of training examples I, mean-squared error (MSE), and peak signal-to-noise ratio (PSNR) on the test set. Reprinted from [158], ©2017 IEEE.

	Algorithm	Oracle?	# iter. K	# tr. I	MSE	PSNR (dB)
Baselines	SISTA	No	3	None	4740	12.1
	SISTA to convergence	No	≤ 1825	None	3530	13.4
	SSpaRSA to convergence	No	≤ 420	None	3520	13.4
	SISTA	Yes	3	None	4160	13.3
	SISTA to convergence	Yes	≤ 694	None	2400	15.0
	SSpaRSA to convergence	Yes	≤ 225	None	2440	15.0
	ℓ_1-homotopy [5]	Yes	≤ 314	None	1490	17.1
	Generic RNN, rand. init.	No	3	24885	720	20.7
Proposed	Trained SISTA-RNN, rand. init.	No	3	24485	637	21.2
	Trained SISTA-RNN, SISTA init.	No	3	24485	**541**	**22.2**

unsupervised baseline, oracle ℓ_1-homotopy [5] and the generic supervised RNN baseline, our proposed trained SISTA-RNN achieves the best objective performance. The trained SISTA-RNN achieves these results without oracle information and reduced computation using a smaller number of fixed iterations $K = 3$. Also, note from figure 7.1 that the SISTA-RNN trains substantially faster than a generic RNN. The SISTA-RNN architecture also performs well even when initialized randomly, instead of with equivalent SISTA parameters. In summary, by combining supervised training with network architecture and parameter initializations provided by SISTA, our proposed trained SISTA-RNN outperforms all baselines and exhibits distinct advantages. Two examples of reconstructed images are shown in figure 7.2.

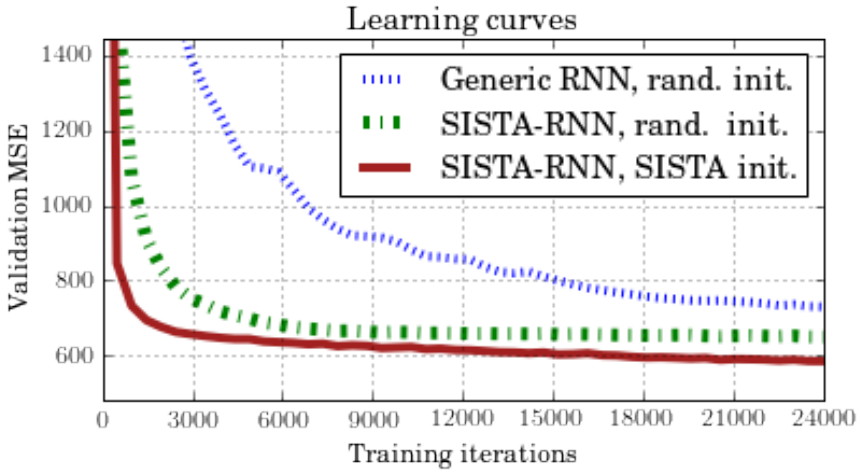

Figure 7.1: Learning curves for supervised methods, showing that the SISTA-RNN trains faster than the generic RNN. Reprinted from [158], ©2017 IEEE.

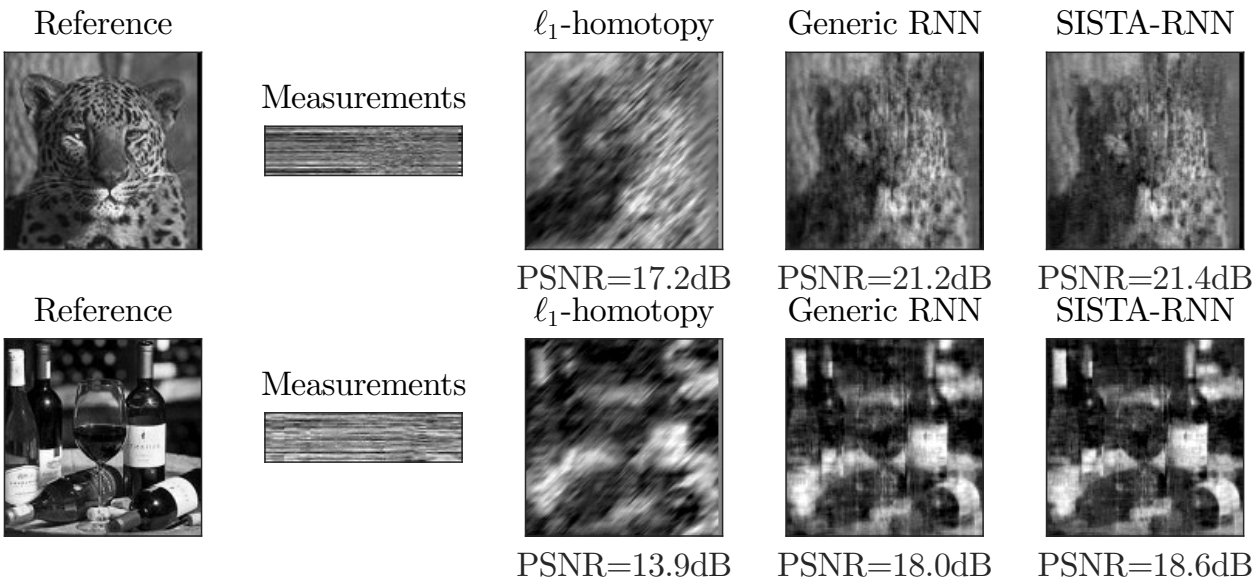

Figure 7.2: Reconstructed images from the test set. Reprinted from [158], ©2017 IEEE.

7.1.1 Interpretability of the SISTA-RNN

The SISTA-RNN is interpretable. For example, it learns new settings of SISTA parameters as $\lambda_1 = 3.07$, $\lambda_2 = -0.04$ and $\alpha = 2.02$. The greater value of λ_1 suggests increased importance of the sparsity penalty, and greater α suggests a smaller iteration step size. Notice that the learned λ_2, which is the regularization weight on the ℓ_2-norm linear prediction error in (6.7), is negative, and thus loses its interpretability as the product of variance σ^2 and precision ν_2. This fact suggests a modification to the SISTA-RNN architecture: add a nonnegativity constraint on λ_2. Doing so is an example of feeding back information we learn from the trained model to improve its design. Figure 7.2 shows example outputs of the different systems and figure 7.3 shows visualizations of the initial and learned SISTA parameters. Notice that the dictionary $\mathbf{D}$ and prediction matrix $\mathbf{F}$ remain the same, which suggests they are good matches to the data.

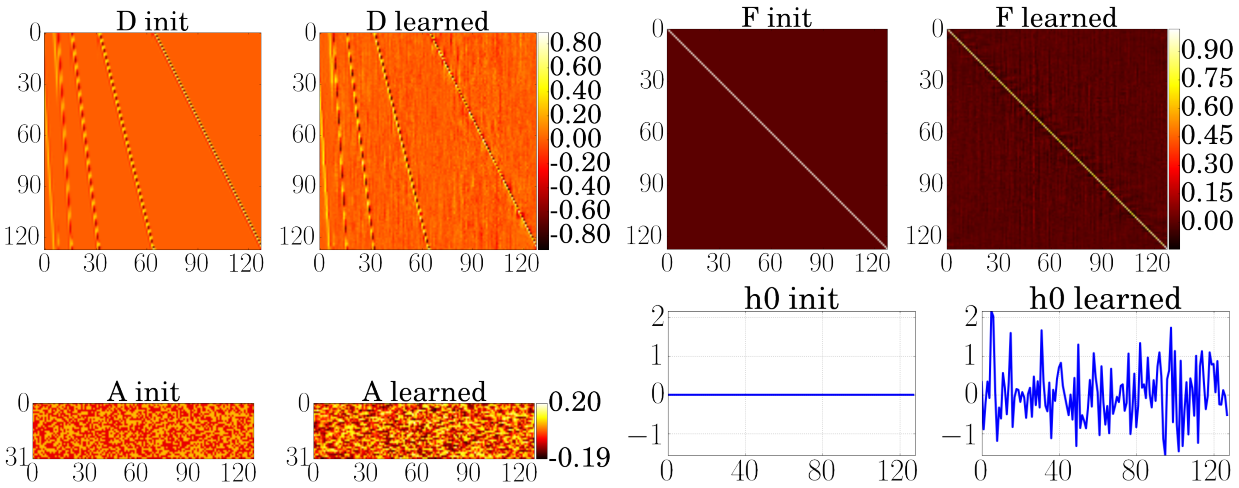

Figure 7.3: Visualizations of some initialized and learned SISTA parameters (6.13) for the SISTA-RNN. See text for settings of λ_1, λ_2, and α.

7.2 Single-Channel Audio Source Separation

Assume that we observe D samples $x_d = y_d + v_d$, $d = 1..D$, of a noisy audio signal, where $y_{1:D}$ is a desired clean speech signal and $v_{1:D}$ is additive noise that we wish to remove. From these noisy samples, the $F \times T$ nonnegative magnitude or power spectrogram matrix $\mathbf{X}$ is computed from the complex-valued short-time Fourier transform (STFT) matrix $\mathbf{X}_{\mathbb{C}}$ of x: $\mathbf{X}_{\mathbb{C}} = \text{STFT}\{x_{1:D}\}$ and $\mathbf{X} = |\mathbf{X}_{\mathbb{C}}|$.

7.2.1 NMF for Speech Separation

As described in section 2.3.3, NMF assumes that $\mathbf{X}$ can be decomposed into the product of a $F \times N$ elementwise nonnegative dictionary $\mathbf{W}$ and a $N \times T$ elementwise nonnegative activation matrix $\mathbf{H}$: $\mathbf{X} \approx \hat{\mathbf{X}} = \mathbf{WH}$, where F is the number of frequency bins, N is the number of NMF basis vectors, and T is the number of spectrogram frames. Usually, the activation matrix $\mathbf{H}$ is constrained to be sparse, which promotes parsimonious representations and avoids trivial solutions [34, 87]. In this section, we will focus on the $\beta = 2$ case, but the method can be used with other values of β.

To separate speech, a dictionary $\mathbf{W}^{(y)}$ is first trained on clean speech. Then a noise dictionary $\mathbf{W}^{(v)}$ is trained by using the concatenated overall dictionary $\mathbf{W} = [\mathbf{W}^{(y)}, \mathbf{W}^{(v)}]$, where the clean speech dictionary $\mathbf{W}^{(y)}$ remains fixed and only the noise dictionary $\mathbf{W}^{(v)}$ and the overall activation matrix $\mathbf{H} = [\mathbf{H}^{(y)}; \mathbf{H}^{(v)}]$ is updated (recall that the notation ";" indicates row concatenation). Figure 7.4 illustrates the dictionary $\mathbf{W}$ learned on the CHiME2 dataset along with the activations $\mathbf{H}$ estimated for a particular noisy spectrogram. To infer the separated speech at test time, the problem (2.15) is solved with just $\mathbf{H}$ as a variable, keeping the overall dictionary $\mathbf{W}$ fixed.

To reconstruct the separated speech signal, a time-frequency filter, or mask, is computed with elements between 0 and 1:

$$\hat{\mathbf{M}} = \frac{\hat{\mathbf{Y}}}{\hat{\mathbf{Y}} + \hat{\mathbf{V}}}, \tag{7.1}$$

where division is elementwise, $\hat{\mathbf{Y}} = \mathbf{W}^{(y)}\mathbf{H}^{(y)}$ is the estimated spectrogram of clean speech,

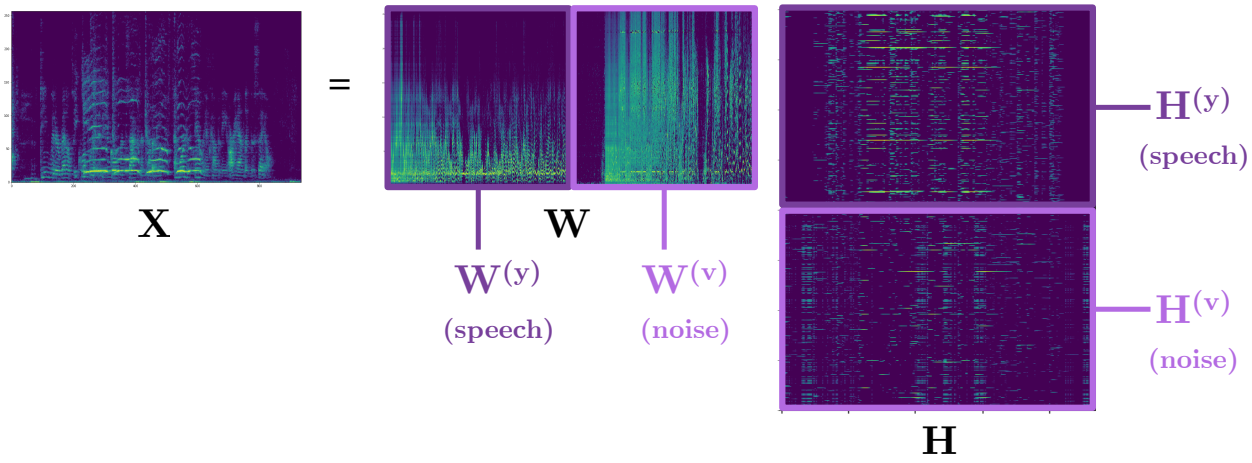

Figure 7.4: Illustration of NMF dictionary $\mathbf{W}$ trained on CHiME2 data and sparse activations $\mathbf{H}$ computed for a single spectrogram $\mathbf{X}$. The speech and noise subsets of the dictionary and activations are indicated, and the dictionary elements are sorted greedily by similarity. Matrices in the figure are not to scale.

and $\hat{\mathbf{V}} = \mathbf{W}^{(v)}\mathbf{H}^{(v)}$ is the estimated spectrogram of noise. This mask is applied to the complex STFT matrix $\mathbf{X}_{\mathbb{C}}$ and the estimated speech signal is the inverse STFT: $\hat{y} = \text{STFT}^{-1}\{\hat{\mathbf{M}} \odot \mathbf{X}_{\mathbb{C}}\}$.

7.2.2 Iterative Soft-Thresholding Algorithm (ISTA) for NMF

The conventional optimization algorithm used to solve problem (2.15) is alternating multiplicative updates [89]. However, the convergence of multiplicative updates can be slow and backpropagating through these updates is challenging [86], so we consider another optimization algorithm that is not often applied to NMF: ISTA.

ISTA can be used at test time to solve the NMF problem for $\mathbf{H}$ by running the algorithm independently for each time frame. That is, in algorithm 1, $\mathbf{x}$ will be the tth column of $\mathbf{X}$, $\mathbf{D}$ will be the trained dictionary $\mathbf{W}$, and $\mathbf{h}$ will be the tth column of the solution $\mathbf{H}$. We

will elect to use a fixed number of ISTA iterations, which we will refer to as K.

However, running ISTA independently on each time frame neglects potential correlation between adjacent frames. Thus, we make ISTA sequential, or recurrent in time, by allowing the ISTA iterations for frame t to use the output of the previous frame $t-1$ as initialization, providing a "warm start". That is, we will set $\mathbf{h}_t^{(0)}$ equal to $\mathbf{h}_{t-1}^{(K)}$, which is the output of K ISTA iterations from frame $t-1$. This procedure is described in algorithm 3. More sophisticated types of recurrence could also be incorporated, such as modeling the sequence of sparse activations $\mathbf{h}_{1:T}$ with a dynamical system [43], which is a promising avenue for future work.

This algorithm is similar to SISTA given in algorithm 2, except that a simpler observation model is used here where the observation matrix $\mathbf{A} = I$, the prediction matrix $\mathbf{D}^{-1}\mathbf{F}\mathbf{D} = \mathbf{P} = \mathbf{I}$, and no linear prediction model is used, so $\lambda_2 = 0$. Also, we replace $\mathbf{D}$ with $\mathbf{W}$ to emphasize that the dictionary matrix is now elementise nonnegative. The modified nonnegative ISTA algorithm with warm-starts is given in algorithm 3.

Algorithm 3 Warm start nonnegative ISTA

Input: observations $\mathbf{x}_{1:T}$, dictionary $\mathbf{W}$, initial coefficients $\mathbf{h}_0^{(K)}$

for $t = 1$ to T **do**
  $\mathbf{h}_t^{(0)} \leftarrow \mathbf{h}_{t-1}^{(K)}$ # *"warm start" from* $t-1$
  for $k = 1$ to K **do**
    $\mathbf{z} \leftarrow (\mathbf{I} - \frac{1}{\alpha}\mathbf{W}^\mathsf{T}\mathbf{W})\mathbf{h}_t^{(k-1)} + \frac{1}{\alpha}\mathbf{W}^\mathsf{T}\mathbf{x}_t$
    $\mathbf{h}_t^{(k)} \leftarrow \mathrm{ReLU}\,(\mathbf{z} - \lambda/\alpha)$
return $\mathbf{h}^{(K)}$

7.2.3 Unfolding ISTA to Deep Recurrent NMF

In this section, we unfold the nonnegative sequential ISTA algorithm 3 for sparse NMF that we described in the last section, which results in a DR-NMF network. The right panel of figure 7.5 illustrates the architecture of the DR-NMF network, which is the unfolded sequential ISTA algorithm. Note that the "warm starts," which are initialization of frame t with the solution from frame $t-1$, manifest as recurrent connections across time.

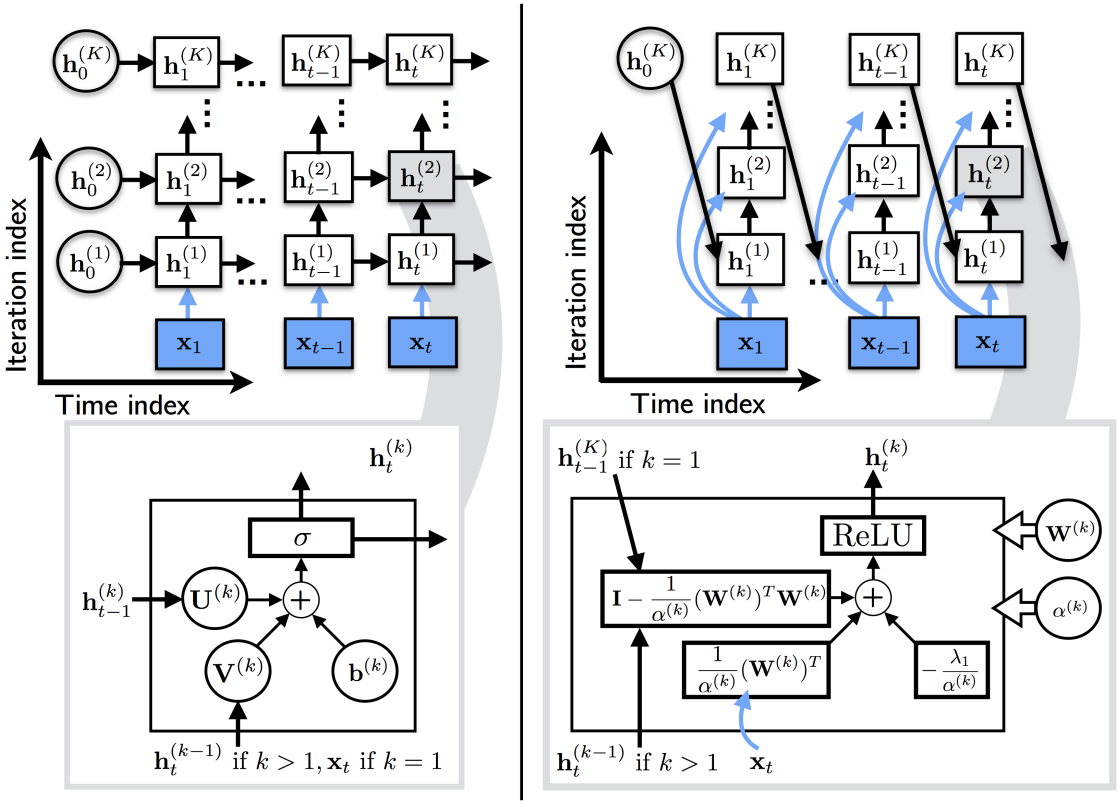

Figure 7.5: Left panel: architecture of conventional stack of RNNs, corresponding to equation (2.32). Right panel: architecture of DR-NMF network, which is algorithm 3 unfolded into a computational graph. Circles indicate trainable weights, and k indexes layers/iterations. The value of the sparse regularization weight λ_1 is a fixed constant. Reprinted from [166], ©2017 IEEE.

Since the nonlinear activation function for nonnegative ISTA is a ReLU, we can see that the unfolded sequential ISTA algorithm is essentially a conventional stack of RNNs that use a ReLU activation function, with two significant differences. First, for each time step t, the input is connected to each node in the deep stack. Second, the only recurrent connection between deep stacks at adjacent time frames is from the top node $\mathbf{h}_{t-1}^{(K)}$ at $t-1$ to the bottom node $\mathbf{h}_t^{(1)}$ at t. For comparison, the left panel of figure 7.5 shows the architecture of a conventional stacked RNN, where the output of the kth RNN layer is described by (2.32), where σ is an activation function with bias $\mathbf{b}$, e.g. the LSTM activation function [67], and for the first layer ($k=1$), $\mathbf{h}_t^{(0)} = \mathbf{x}_t$.

To train a stacked RNN or DR-NMF network with a supervised dataset $\{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1:I}$ with I examples, we solve the problem

$$\underset{\theta}{\text{minimize}} \quad \frac{1}{I}\sum_{i=1}^{I} \ell\left(\mathbf{Y}_i, q_\theta(\mathbf{X}_i)\right), \tag{7.2}$$

where ℓ is a training loss function, q_θ is the neural network output, and θ are the weights of the network. We use stochastic gradient descent, where backpropagation through q_θ is used to compute the gradients with respect to the trainable parameters θ.

For the speech separation problem, an input example $\mathbf{X}$ is the $F \times T$ magnitude spectrogram of a noisy audio file, while the output $\mathbf{Y}$ is the $F \times T$ magnitude spectrogram of the corresponding clean speech signal that we are trying to predict with the neural network. For our experiments, we use the signal approximation cost function [156]. This cost function assumes that the network estimates a $F \times T$ masking matrix $\hat{\mathbf{M}}$ using (7.1) and multiplies this mask elementwise with the noisy input, $q_\theta(\mathbf{X}_i) = \hat{\mathbf{M}} \odot \mathbf{X}$. Then the signal approximation loss is the mean squared error between the true clean spectrogram $\mathbf{Y}$ and the estimated clean spectrogram $\hat{\mathbf{M}} \odot \mathbf{X}$:

$$\ell(\mathbf{Y}, q_\theta(\mathbf{X})) = \sum_{f,t} (Y_{f,t} - \hat{M}_{f,t} X_{f,t})^2, \tag{7.3}$$

which corresponds to maximizing the signal-to-noise ratio (SNR) of magnitude spectra in the time-frequency domain.

Since DR-NMF corresponds to an optimization algorithm that solves the NMF optimization problem (2.15), we can initialize DR-NMF using sparse NMF. The overall training procedure is described by the following steps:

1. Train clean speech dictionary $\mathbf{W}^{(y)}$ on clean speech audio using "well-done" sparse NMF multiplicative updates as described by Le Roux et al. [87].

2. Train noise dictionary $\mathbf{W}^{(v)}$ on noisy speech audio by building the overall dictionary $\mathbf{W} = [\mathbf{W}^{(y)}, \mathbf{W}^{(v)}]$ and only updating $\mathbf{W}^{(v)}$ using "well-done" sparse NMF multiplicative updates [87].

3. Initialize DR-NMF network with the learned dictionary $\mathbf{W}$ and ISTA optimization parameters α and $\mathbf{h}_0$.

4. Train the DR-NMF parameters $\theta = \{\mathbf{W}^{(1:K)}, \alpha^{(1:K)}, \mathbf{h}_0\}$, where the dictionary $\mathbf{W}$ and inverse step size α are untied across layers, by solving the problem (7.2) using stochastic gradient descent with the signal approximation loss (7.3).

When initializing the DR-NMF network, the ISTA inverse step size α must be chosen appropriately to ensure that a fixed number K of iterations achieves sufficient decrease in the sparse NMF objective function (2.15). Since one iteration of ISTA performs a gradient descent step before thresholding, α corresponds to the Lipschitz smoothness of f, the smooth part of the cost function in (2.20). Thus, for squared error ($\beta = 2$), we have the following lemma for bounding α:

Lemma 7.2.1 *If $\mathbf{A} = \mathbf{I}$, the nonnegative dictionary $\mathbf{W} \in \mathbb{R}_+^{F \times N}$ is constrained to have unit-norm columns, the sparse coefficients $\mathbf{h}$ are nonnegative, and if we assume that the maximum inner product between two different columns is upper-bounded by δ, then the Lipschitz smoothness L of the smooth part of the cost function in (2.17) is bounded as*

$$L \geq 2(1 + \delta(N - 1))^2. \tag{7.4}$$

Proof: The Lipschitz smoothness L of the function $f(\mathbf{h}) = \|\mathbf{x} - \mathbf{W}\mathbf{h}\|_2^2$ is given by twice the largest eigenvalue of $\mathbf{W}^\mathsf{T}\mathbf{W}$ [9, Example 2.2]:

$$L = 2\lambda_{max}\{\mathbf{W}^\mathsf{T}\mathbf{W}\}. \tag{7.5}$$

Recall that for a matrix $\mathbf{Z}$, the 2-norm is equal to the square root of the largest eigenvalue of $\mathbf{Z}$ [58, page 57]:

$$\|\mathbf{Z}\|_2 = \max_{\|\mathbf{x}\|_2=1} \|\mathbf{Z}\mathbf{x}\|_2 = \sqrt{\lambda_{max}\{\mathbf{Z}\}}. \tag{7.6}$$

A useful bound on the 2-norm is the following [58, corrollary 2.3.2]:

$$\|\mathbf{Z}\|_2 \leq \sqrt{\|\mathbf{Z}\|_1 \|\mathbf{Z}\|_\infty}, \tag{7.7}$$

where the 1-norm and ∞-norm are, respectively,

$$\|\mathbf{Z}\|_1 = \max_{j=1..N} \sum_{i=1}^{N} |Z_{i,j}|, \tag{7.8}$$

$$\|\mathbf{Z}\|_\infty = \max_{i=1..N} \sum_{j=1}^{N} |Z_{i,j}|. \tag{7.9}$$

Since $\mathbf{W}$ is elementwise nonnegative and has unit-norm columns, $\mathbf{W}^\mathsf{T}\mathbf{W}$ has diagonal elements all equal to 1 and off-diagonal elements that are bounded between 0 and δ, where δ is the maximum coherence between any two distinct columns of $\mathbf{W}$. Thus, the sum of any row or column is bounded by $1 + \delta(N-1)$, which means that

$$(1 + \delta(N-1))^2 \geq \|\mathbf{W}^\mathsf{T}\mathbf{W}\|_1 \|\mathbf{W}^\mathsf{T}\mathbf{W}\|_\infty. \tag{7.10}$$

Using (7.6), (7.7), and (7.10) we can say that

$$(1 + \delta(N-1))^2 \geq \lambda_{max}\{\mathbf{W}^\mathsf{T}\mathbf{W}\} \tag{7.11}$$

Thus, using (7.5), we have the following bound on the Lipschitz smoothness L:

$$L \geq 2\big(1 + \delta(N-1)\big)^2. \tag{7.12}$$

□

To ensure nonnegativity of the DR-NMF weights, the logs of the weights are optimized through an elementwise exp function. For example, for the nonnegative weight α, we optimize $\widetilde{\alpha}$, which is initialized with $\log(\epsilon + \alpha)$, and use $\exp(\widetilde{\alpha})$ for the model weight. To maintain nonnegativity and unit-norm columns of $\mathbf{W}$, we optimize $\widetilde{\mathbf{W}}$, which is initialized with $\log(\epsilon + \mathbf{W})$, and use $\exp(\widetilde{\mathbf{W}})\text{diag}^{-1}\left(\sqrt{\sum_f \exp(\widetilde{W}_{f,:})^2}\right)$ as the model weights.

7.2.4 Experiments and Discussion

Table 7.2: Results in terms of validation loss (dev. loss) and signal-to-disortion ratio (SDR) in dB on the CHiME2 development and test sets using 100% (center section) or 10% (right section) of the training data. K is the number of layers or iterations, N is the LSTM hidden state dimension or the number of NMF basis vectors, and P is the total number of trainable parameters. Reprinted from [166], ©2017 IEEE.

	Architecture				**100% of training set**			**10% of training set**		
	Model	K	N	P	Dev. loss	Dev. SDR	Eval. SDR	Dev. loss	Dev. SDR	Test SDR
	SNMF, MU	200	200	50k	0.0987	7.14	8.18	0.1319	6.51	7.47
	SNMF, MU	200	2000	500k	0.0846	7.71	8.61	0.0890	7.43	8.37
Baselines	LSTM	2	54	100k	0.0408	11.51	12.53	0.0512	10.34	11.35
Baselines	LSTM	2	244	1M	0.0339	11.95	12.90	0.0481	10.59	11.57
Baselines	LSTM	5	70	250k	0.0426	10.90	11.94	0.0542	10.22	11.26
Baselines	LSTM	5	250	2.5M	0.0344	**12.35**	**13.30**	0.0566	10.25	11.32
Proposed	DR-NMF	2	200	100k	0.0320	10.94	11.86	0.0362	10.39	11.33
Proposed	DR-NMF	2	2000	1M	0.0295	11.21	12.11	0.0354	10.54	11.43
Proposed	DR-NMF	5	200	250k	0.0286	11.14	12.04	0.0332	10.88	11.80
Proposed	DR-NMF	5	2000	2.5M	**0.0266**	11.31	12.19	**0.0316**	**11.12**	**11.99**

For these experiments, the CHiME2 corpus is used, [153] which consists of utterances from the Wall Street Journal (WSJ-0) dataset that are convolved with binaural room impulse responses (RIRs) and mixed with real-world nonstationary noise at six different SNRs from

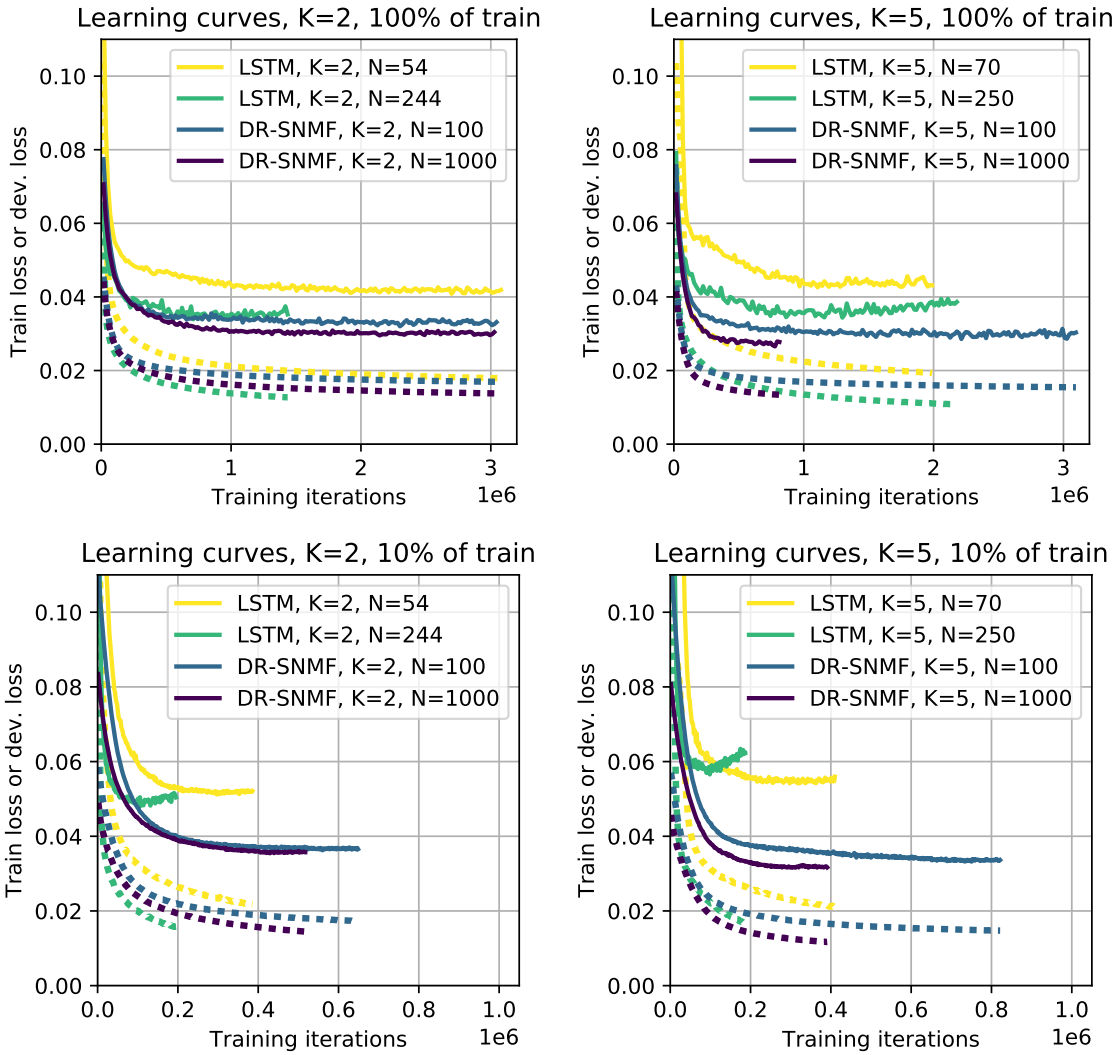

Figure 7.6: Learning curves for deep models. Dotted lines are training loss and solid lines are validation loss, where validation loss is computed on the CHiME2 development set. Notice that large LSTM networks generalize well when 100% of the training data is used (top two panels), but they tend to quickly overfit when only 10% of the training data is used (bottom two panels). In contrast, DR-NMF networks achieve good generalization performance and the lowest validation loss using either 100% or 10% of the training data (all panels). Reprinted from [166], ©2017 IEEE.

−6dB to 9dB, spaced by 3dB. The nonstationary noise was recorded in a home environment, and contains a variety of challenging noise types, including music, radio, television, children, and appliances. The RIRs were recorded in the same environment. The training set consists of 7138 utterances, the development set consists of 2460 utterances, and the test set consists of 1980 utterances, which are equally distributed across the six SNRs. All audio is sampled at 16kHz and only the left channel is used. Separation performance is measured using signal-to-distortion ratio (SDR) from the BSS Eval Matlab toolbox [150, 151].

Two baseline methods are compared to DR-NMF: sparse NMF (SNMF) using well-done multiplicative updates (MU), which we use to initialize the DR-NMF networks, and stacks of LSTM networks. SNMF is run for 200 MU iterations at test time. The depth K of the LSTM stacks are chosen to match the depth of the DR-NMF networks, and the hidden node count N of the LSTMS are chosen to match the counts P of trainable parameters in the DR-NMF networks.

The STFT uses 512-sample square-root Hann windows for analysis and synthesis with a hop of 128 samples. Thus, the feature dimension of input spectrogram frames is $F = 257$. For training deep models, the input spectrograms are split into sequences no longer than 500 frames. Deep network training uses the Adam optimizer [80] with a batch size of 32 and default parameters, except for the learning rate. LSTM network training uses gradient clipping to 1 and a learning rate of 10^{-4}, and DR-NMF network training uses no gradient clipping and a learning rate 10^{-3}. During training, model weights with the lowest validation loss are saved. Early stopping on the validation loss with a patience of 50 epochs determines training convergence. All deep networks are implemented in Keras [22] using Theano [144] as a backend. For sparse NMF, we use the Matlab implementation of well-done multiplicative updates [87]. Code to replicate these results is available online[5].

The results are shown in table 7.2. Notice that when deep models are trained with 100% of the training set (center section of table 7.2), the largest LSTM model achieves the best

[5] `https://github.com/stwisdom/dr-nmf`

mean SDR of 12.35 dB on the CHiME2 development set. However, despite not achieving the best SDR, the largest DR-NMF model achieves the best validation loss of 0.0266. This indicates a discrepancy between SDR, which is computed in the time domain, and the signal approximation loss (7.3) computed in the magnitude spectrogram domain.

When only 10% of the training set is used, DR-NMF networks achieve superior performance in terms of SDR and validation loss (right section of table 7.2), achieving 11.12 dB SDR on the development set, compared to the best LSTM score of 10.59 dB. DR-NMF also outperforms Le Roux et al.'s deep NMF, which achieves 10.20 SDR on the development set using $P = 1.2$M trainable parameters, $N = 2000$ basis vectors, and $K = 25$ total MU iterations, the last 3 of which are untied and trained [86, table 2].

Figure 7.6 shows the learning curves for training the deep networks, which provide insight into the generalization ability of the various models. Notice that large LSTM networks quickly overfit (i.e., the validation loss starts to increase while the training loss continues to decrease) when only provided with 10% of the training set (right panels of figure 7.6), while DR-NMF networks are consistently robust to overfitting. This suggests that DR-NMF networks exhibit stronger generalization performance compared to LSTMs and thus perform better when provided with less training data. Also, DR-NMF networks achieve the lowest validation loss in all cases.

7.3 Multi-Channel Audio Source Separation

We assume that J acoustic sources x^j are recorded by I microphones. Let $\mathbf{Y}_{f,t} \in \mathbb{C}^I$ be the complex-valued STFT coefficients of the I microphones at frame $t \in \{1..T\}$ and frequency $f \in \{1..F\}$. The STFT window and FFT lengths are both taken to be $2(F-1)$. The ith microphone signal is given by

$$Y^i_{f,t} = \sum_j B^{i,j}_f X^j_{f,t} + V^i_{f,t}, \tag{7.13}$$

where $X^j_{f,t}$ is the STFT coefficient of the jth source, $V^i_{f,t}$ is additive, zero-mean, circular, complex-valued Gaussian noise, and $B^{i,j}_f$ is the value at frequency f of the FFT of the channel

$\mathbf{b}^{i,j}_{1:N_c}$ from source j to microphone i, where we assume a narrowband channel model: that is, the channel impulse response $\mathbf{b}^{i,j}_{1:N_c}$ is shorter than the analysis window length: $N_c \leq N_w$. By using a narrowband assumption, the effect of the channel is a complex-valued gain $B^{i,j}_f$ in each frequency bin f for each microphone-source pair (i, j).

7.3.1 Source Separation Using the Multichannel GMM

For this problem, the conventional model will be used, where each time-frequency bin of each source is modeled with a zero-mean, circular, complex-valued Gaussian, the variance of which is dependent on the source state. Attias [6] originally proposed such a model, using a multinomial distribution for the source states, leading to a GMM for each source. The sources are mapped onto an array of microphones via linear time-invariant channel models B_f. The model is formulated as follows: for each time t, a source's state is given by $z^j_t \in \{1..Z\}$, which controls a pattern of variances across frequency, given by $1/\gamma^{j,z}_f$, where $\gamma^{j,z}_f$ are state-dependent precisions. That is,

$$\left(X^j_{f,t} | z^j_t = z\right) \sim \mathcal{N}_{\mathbb{C}}(0, 1/\gamma^{j,z}_f). \tag{7.14}$$

Each channel is assumed to have a small amount of additive, independent, zero-mean, circular, complex-valued Gaussian noise. The observations are thus distributed as

$$\left(Y^i_{f,t} | X^{1:J}_{f,t}\right) \sim \mathcal{N}_{\mathbb{C}}\Big(\sum_j B^{i,j}_f X^j_{f,t}, 1/\psi^i_f\Big), \tag{7.15}$$

where ψ^i_f is a precision for the additive sensor noise $V^i_{f,t}$. The states z^j for source j have priors $\pi^{j,z} := p(z^j = z)$, where z is a value in $\{1..Z\}$. The channel model $\mathbf{B}_f$ is here considered a parameter. A graphical model is shown in Fig. 7.7.

Exact inference in this model is intractable because the E-step requires summing over an exponential number of terms ($\mathcal{O}\left(Z^J\right)$) in the marginalization over states. However, an approximate variational algorithm [73] can be derived, which was done by Attias [6]. The approximate inference algorithm uses the variational approximation

$$q(X^{1:J}_{f,t}, z^{1:J}_t) = \Big[\prod_f \prod_j q(X^j_{t,f} | z^j_t)\Big]\Big[\prod_j q(z^j_t)\Big], \tag{7.16}$$

where $q(X_{f,t}^j|z_{f,t}^j = z) = \mathcal{N}_{\mathbb{C}}\big(X_{f,t}^j; \bar{\mu}_{f,t}^{j,z}, 1/\bar{\gamma}_f^{j,z}\big)$, and $q(z_{f,t}^j = z) = \bar{\pi}_t^{j,z}$. In this variational approximation, $\bar{\mu}_{f,t}^{j,z}$ is the state-dependent variational posterior mean and $\bar{\gamma}_f^{j,z}$ is the state-dependent variational posterior precision of source j at time-frequency (t, f). The variational updates are given in Attias [6, eq. (10)-(15)].

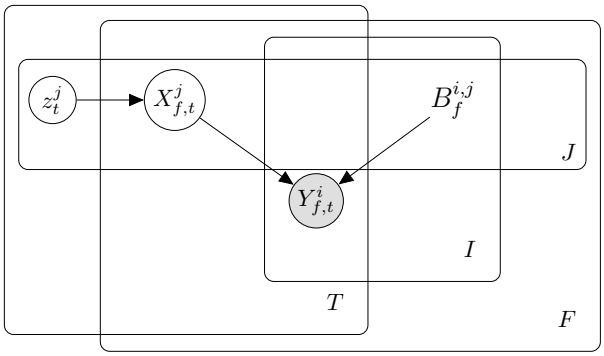

Figure 7.7: Graphical model of the multichannel GMM.

7.3.2 Unfolding the Multichannel GMM

Here we apply the deep unfolding framework in the context of the MCGMM. A key difference with the application considered in previous deep unfolding work [65, 86] is that several updates in the complex-valued unfolded MCGMM involve non-holomorphic functions of complex-valued variables. Because of these non-holomorphic functions, the usual complex gradient is not sufficient to perform gradient descent. Also, gradients in a real-imaginary representation can be algebraically cumbersome. Fortunately, we can overcome these issues by using a generalization of the complex gradient defined using Wirtinger calculus [84].

In this section, we formulate a complex-valued computational network inspired by the unfolded MCGMM variational algorithm. Algorithm 4 describes the sequence of updates performed in each layer of the network. A computational graph of the last two layers of the network is shown in figure 7.8.

We make two simplifications to the variational updates of the MCGMM in order to make them easier to take gradients through. We desire to avoid matrix inversions and to

make the updates synchronous across all sources in each layer. First, instead of solving a $J \times J$ linear system of equations to estimate the variational source estimates, as in Attias [6, eq. (12)], we elect to perform "synchronous" updates of the state-dependent source means given by (4), which are an alternative variational update. If these updates are performed on one source at a time, the variational bound is maintained. But performing these updates synchronously breaks the variational lower bound on the log-likelihood. In practice, we have not observed degradation of the separation performance, as long as the synchronous updates are preceded by at least a few iterations of the original variational updates. Using the synchronous updates, the output estimate of source j in layer k is the variational posterior mean $\hat{X}_{f,t}^{j,(k)}$, given by (9).

Our second simplification concerns the cross-source covariance matrix $\hat{\mathbf{\Sigma}}_{f,t}^{\hat{X}\hat{X}}$. Attias [6] assumed a full covariance matrix, which necessitates a matrix inversion in the M-step for the update of the channel B given by line 17 of algorithm 4. To avoid this matrix inversion, we make the reasonable assumption that sources are uncorrelated, and constrain the cross-source covariance matrix $\hat{\mathbf{\Sigma}}_{f,t}^{\hat{X}\hat{X}}$ in line 15 of algorithm 4 to be diagonal.

In preliminary experiments, discriminatively training the state priors $\boldsymbol{\pi}^{(k)}$ and GMM variances $\boldsymbol{\gamma}^{(k)}$ in each layer through these updates using the cost function in (7.23) did not yield substantial improvements. As such, in the next section we consider extending the generative model such that it has greater representational power. In particular, we will focus on improving estimation of source states, since these are essential for effective source separation.

7.3.3 MRF Extension of the MCGMM

We would like to improve the unfolded MCGMM network's ability to estimate the correct state for each source at the output. One way to accomplish this is to add feedback to the network such that the estimated posterior log-likelihoods $L_t^{j,z,(k)}$ of the states in layer k (5) use information about the estimated posterior state likelihoods $\pi_t^{j,z,(k-1)}$ (7) in the previous layer, $k-1$.

Such a mechanism exists in a deep unfolded pairwise binary MRF. As described in section 6.3.1, unfolding mean-field inference in a binary MRF leads to a deep feed-forward sigmoid network [65]. Given a MRF with M hidden binary random variables s_m, log potentials Ψ_{ss}, and the log-likelihood of the observed data L_{obs}, the posterior distribution can be written as

$$p(\mathbf{s}|\mathbf{v}) \propto \exp\left(\frac{1}{2}\mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{s} + \mathbf{s}^{\mathsf{T}}\mathbf{c} + \mathbf{s}^{\mathsf{T}}\mathbf{L}_{\text{obs}}\right) \tag{7.17}$$

where $\mathbf{s} := s_{1:M}$, $\mathbf{A} \in \mathbb{R}^{M\times M}$, $A_{m,m} = 0$ for all m, $A_{m_1,m_2} = A_{m_2,m_1}$ for $m_1 \neq m_2$, $\mathbf{c} \in \mathbb{R}^M$ are derived from the log potentials Ψ_{ss}, and $\mathbf{L}_{\text{obs}} \in \mathbb{R}^M$ [65, Appendix A].

The variational posterior probability $\bar{\boldsymbol{\pi}}^{(k)} := \{q^{(k)}(s_m)\}_{m=1:M}$ in iteration k of the mean-field inference algorithm is then

$$\bar{\boldsymbol{\pi}}^{(k)} = \sigma\left(\mathbf{A}\bar{\boldsymbol{\pi}}^{(k-1)} + \mathbf{c} + \mathbf{L}_{\text{obs}}\right), \tag{7.18}$$

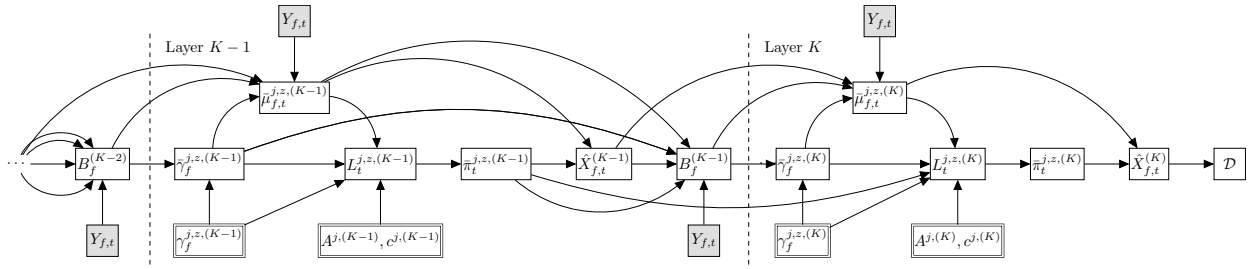

Figure 7.8: Last two layers of the unfolded deep MCGMM. Boxes with double lines are the discriminatively-trained source parameters, and shaded boxes represent the observed data. Reprinted from [161], ©2016 IEEE.

where σ is the element-wise sigmoid function. Notice that $\mathbf{A}$ and $\mathbf{c} + \mathbf{L}_{\text{obs}}$ define an affine transformation, and if these parameters are untied across layers, $\mathbf{A}^{(k)}$ and $\mathbf{c}^{(k)}$, then equation (7.18) is equivalent to one layer of a deep feed-forward sigmoid network. Discriminatively training the $\mathbf{A}^{(k)}$ and $\mathbf{c}^{(k)}$ in each layer is equivalent to finding a different set of log potential

Algorithm 4 Simplified variational EM algorithm for the MCGMM, where $\langle (\cdot)_t \rangle_t := \frac{1}{T}\sum_{t=1}^{T}(\cdot)_t$.

Input: multichannel mixture STFT $\mathbf{Y}_{1:F,1:T}$, sensor precision $\boldsymbol{\psi}_f$, source parameters $\boldsymbol{\gamma}_{1:F}^{1:J,1:Z}$, $\boldsymbol{\pi}^{1:J,1:Z}$, initial channel estimate $\mathbf{B}_{1:F}^{(0)}$

for $k = 1$ to K **do**

Run E-step:

$\bar{\gamma}_f^{j,z,(k)} = \big[\mathbf{B}_f^{(k-1)}\big]_{:,j}^{\mathsf{H}} \psi_f \big[\mathbf{B}_f^{(k-1)}\big]_{:,j} + \gamma_f^{j,z,(k)}$

$\bar{\mu}_{f,t}^{j,z,(k)} = \frac{\big[\mathbf{B}_f^{(k-1)}\big]_{:,j}^{\mathsf{H}} \boldsymbol{\psi}_f}{\bar{\gamma}_f^{j,z,(k)}} \Big(\mathbf{Y}_{f,t} - \big[\mathbf{B}_f^{(k-1)}\big]_{:,\setminus j} \hat{X}_{f,t}^{\setminus j,(k-1)} \Big)$

$L_t^{j,z,(k)} = \log \pi^{j,z} + \sum_f \log \frac{\gamma_f^{j,z,(k)}}{\bar{\gamma}_f^{j,z,(k)}} + \sum_f \bar{\gamma}_f^{j,z,(k)} \left| \bar{\mu}_f^{j,z,(k)} \right|^2$

$\bar{\pi}_t^{j,z,(k)} = \Big[\mathrm{softmax}\Big(\mathbf{L}_t^{j,1:Z,(k)} \Big) \Big]_z$

$\hat{X}_{f,t}^{j,(k)} = \sum_z \bar{\pi}_t^{j,z,(k)} \bar{\mu}_{f,t}^{j,z,(k)}$

Run M-step:

$\hat{\boldsymbol{\Sigma}}_f^{YX} = \Big\langle \mathbf{Y}_{f,t} \big(\hat{\mathbf{X}}_{f,t}^{(k)} \big)^{\mathsf{H}} \Big\rangle_t$

$\big[\hat{\boldsymbol{\Sigma}}_f^{\hat{X}\hat{X}} \big]_{j,j} = \Big\langle \sum_z \bar{\pi}_t^{j,z,(k)} \Big(\frac{1}{\bar{\gamma}_f^{j,z,(k)}} + \left| \bar{\mu}_{f,t}^{j,z,(k)} \right|^2 \Big) \Big\rangle_t$

$\mathbf{B}_f^{(k)} = \hat{\boldsymbol{\Sigma}}_f^{Y\hat{X}} \Big(\hat{\boldsymbol{\Sigma}}_f^{\hat{X}\hat{X}} \Big)^{-1}$

return estimated source STFTs $\hat{\mathbf{X}}_{1:F,1:T}^{1:J,(K)}$ and layer-wise intermediate variables

functions for the MRF for each iteration, such that the result of K iterations of inference minimizes the discriminative cost function. The expression $\mathbf{A}^{(k)}\bar{\boldsymbol{\pi}}^{(k-1)} + \mathbf{c}^{(k)}$ is essentially a prior on the state log-likelihoods that varies from iteration to iteration, with feedback from the previously estimated state likelihoods $\bar{\boldsymbol{\pi}}^{(k-1)}$.

To apply this in our model we can replace the multinomial state $z_t^j \in \{1..Z\}$ of a source with a MRF as in the above. To do this, let each multinomial state z_t^j be mapped to Z binary random variables $s_t^{j,z}$ in a fully-connected MRF, where $\mathbf{s}_t^{j,1:Z}$ is constrained to be one-hot. We use the variational approximation $q(\mathbf{s}_t^{j,1:Z}) = \prod_z \bar{\pi}_t^{j,z}$ for the binary random variables $s_t^{j,z}$, with variational probabilities $\bar{\pi}_t^{j,z} := q(s_t^{j,z} = 1, s_t^{j,z'} = 0,\ \forall z' \neq z)$. Rather than performing unconstrained mean-field updates, here for continuity with our GMM model, we constrain the variational posterior to behave like multinomial mixture states. As such $\bar{\pi}_f^{j,z}$ is the variational probability that the zth element of $\mathbf{s}_t^{j,1:Z}$ is set to 1, and the other elements are set to 0. Then, if we unfold mean field inference for the hidden binary states $s_t^{j,z}$, we replace the multinomial prior $\log \pi^{j,z}$ in the update (5) with

$$L_{\text{prior},t}^{j,z,(k)} = \left[\mathbf{A}^{(k)}\right]_{z,:} \bar{\boldsymbol{\pi}}_t^{j,1:Z,(k-1)} + \left[c^{(k)}\right]_z, \tag{7.19}$$

where the parameters $\mathbf{A}^{(k)} \in \mathbb{R}^{Z\times Z}$ and $\mathbf{c}^{(k)} \in \mathbb{R}^Z$ can be layer-dependent. When $\mathbf{A}^{(k)} = \mathbf{0}$ and $c^{(k)} = \log \pi^{j,z}$ for all k, the new update (7.20) simplifies to the original variational update (5). Although the synchronous mean-field updates break the variational bound, we expect discriminative training to compensate such approximations.

The new update for $L_t^{j,z,(k)}$ that replaces (5) is thus

$$L_t^{j,z,(k)} = L_{\text{prior},t}^{j,z,(k)} + \alpha L_{\text{acoustic},t}^{j,z,(k)}, \tag{7.20}$$

with

$$L_{\text{acoustic},t}^{j,z,(k)} = \sum_f \log \frac{\gamma_f^{j,z,(k)}}{\bar{\gamma}_f^{j,z,(k)}} + \sum_f \bar{\gamma}_f^{j,z,(k)} \left|\bar{\mu}_f^{j,z,(k)}\right|^2. \tag{7.21}$$

Equation (7.21) is the part of the log-likelihood corresponding to acoustic information and α is an "acoustic weight" that expresses the importance of the acoustic evidence over the prior. We refer to the resulting network as a deep MCGMM (DMCGMM).

7.3.4 Experiments and Discussion

We use a modified version[6] of the SimData and multicondition training (mcTrain) data components of the REVERB challenge dataset [82]. Each file consists of a single-channel speech utterance from the WSJCAM0 dataset [125] reverberated using measured 8-channel reverberation impulse responses (RIRs) in different rooms. SimData uses RIRs from three different rooms, and mcTrain uses RIRs from 6 different rooms. Stationary noise that was recorded in each particular room is added at 20 dB SNR. To create a dataset of overlapping speech, a second speech signal is reverberated using a RIR from a different position in the same room and added to the original file. No normalization of the power of the reverberated speech sources is performed, in order to simulate realistic conditions. The power ratio between the spatial images of speaker 1 and speaker 2 ranges from about -15 dB to $+15$ dB. All mixes are between 6 and 10 seconds long. The training set contains 15763 mixes, the development set contains 965 mixes, and the evaluation set contains 1435 mixes.

The initial source precisions $\gamma_f^{j,z,(0)}$ were trained on a gender-specific split of the WSJ-CAM0 training set. That is, two separate 256-component GMMs were trained for male and female speakers. Each GMM was first trained on the log-magnitude STFTs. Then, using the frame labels ℓ from the result, the GMM precisions γ_f^z were set to be $1/\sum_{t:\ell(t)=z}|X_{f,t}|^2$. Then these gender-specific GMMs were concatenated into a 512-component GMM. The MRF parameters are initialized as $A^{(0)} = 0$ and $c^{(0)} = \log \pi^z$. Both sources use the same source model.

Since our main interest here is to observe the performance improvement of the DMCGMM over the conventional MCGMM, we used an oracle least-squares initialization for the channel model for each file:

$$\mathbf{B}_f^{(0)} = \hat{\mathbf{\Sigma}}_f^{YX} \left(\hat{\mathbf{\Sigma}}_f^{XX}\right)^{-1}, \tag{7.22}$$

where $\hat{\mathbf{\Sigma}}_f^{YX}$ is the frequency-domain cross-covariance between the microphone observations $\mathbf{Y}_{f,t}$ and reference sources $\mathbf{X}_{f,t}$, and $\hat{\mathbf{\Sigma}}_f^{XX}$ is the covariance between the reference sources

[6]Thanks to Michael Mandel for building this dataset during JSALT 2015.

$\mathbf{X}_{f,t}$. The sensor precision $\boldsymbol{\psi}_f$ is set to 5 times the sample precision of the data $\mathbf{Y}_{f,1:T}$.

For each file, 10 iterations of variational EM updates, as described in Section 7.3.1, are run. The output of these iterations is fed to a network of K DMCGMM layers, as described in Section 7.3.2. The parameters $\Theta^{(k)} = \{\mathbf{A}^{(k)}, \mathbf{c}^{(k)}, \boldsymbol{\gamma}_{1:F}^{1:J,1:Z,(k)}\}$ are untied between layers and discriminatively trained. We use an "error-to-source" (ESR) cost function given by

$$\mathcal{D}_{ESR}(\hat{\mathbf{X}}_{f,t}^{(K)}, \mathbf{X}_{f,t}) = \sum_j \frac{\sum_{f,t} \left| \hat{X}_{f,t}^{j,(K)} - X_{f,t}^j \right|^2}{\sum_{f,t} \left| X_{f,t}^j \right|^2}, \tag{7.23}$$

where $\hat{X}_{f,t}^{(K)}$ are the estimated source STFT coefficients from the last (Kth) layer and $X_{f,t}$ are the clean single-channel references. By minimizing (7.23), the signal-to-noise ratio of both sources is maximized.

Performance is measured using signal-to-distortion ratios of the first channel of the source spatial image estimates (SDRim) and the source estimates (SDR). SDRim is computed using `bss_eval_images` from the BSS Eval toolbox [152], and SDR is computed as the signal-to-noise ratio when the reference signal is allowed an arbitrary gain estimated using least-squares. The SDRs for "no processing" in table 7.3 are very low because the noisy mixtures contain a large amount of reverberation[7]. We incrementally train DMCGMMs layerwise, using the parameters with the best validation mean SDRim for each successive layer. To ensure the GMM source precisions $\gamma_f^{j,z,(k)}$ remain nonnegative, we optimize $\lambda_f^{j,z,(k)} := \log \gamma_f^{j,z,(k)}$, and replace all instances of $\gamma_f^{j,z,(k)}$ in the updates with $\exp \lambda_f^{j,z,(k)}$. The $\mathbf{A}^{(k)}$ variables are unconstrained. Stochastic gradient descent is used for backpropagation with a batch size of 2 files (about 500 STFT frames on average). An initial learning rate of 20 gave the best results. Momentum of 0.9 is used. A validation set, with 65 randomly selected files from the development set, is scored every 300 gradient steps and used for early stopping. Stochastic gradient descent is performed for 30 epochs, with files randomly shuffled in each epoch.

Matlab was used to implement the MCGMM variational inference algorithm, the forward

[7]Note that only the gain is adapted because a more flexible adaptation between the reference and signal would constitute an oracle microphone array method, and our aim here is to show the SNR without processing.

pass of the DMCGMM, and gradient computations for discriminative training. All computations are performed on a NVIDIA GPU using the Matlab Parallel Processing Toolbox. For a 10 second mixture, this implementation takes about 2 seconds to perform the MCGMM variational algorithm and about 500 milliseconds to perform a DMCGMM forward pass and gradient computation for backpropagation.

Table 7.3 shows the results on the evaluation set, which are given in SDRim and SDR. Scores are averaged over in categories based on input SDR. Notice that both SDRim and SDR generally increase as the number of discriminatively-trained DMCGMM layers increases.

7.4 Audio Classification

As mentioned before, a very useful feature of unfolded networks is that they can be trained with any differentiable loss function. The previous sections have described applications to audio source separation, which are inherently regression problems. In this section, we consider training unfolded DR-NMF networks using a training loss function for classification.

This is not the first time that NMF has been used for classification. By adding a discriminative classification term to the NMF training cost in (2.15), the trained NMF dictionary $\mathbf{W}$ and the activations $\mathbf{H}$ are encouraged not only represent the input data well, but also to classify the data according to the training labels. These approaches are generally referred to as "task-driven" [99]. In task-driven dictionary learning for classification, additional classifier parameters are introduced. These parameters are used by a classifier to process the dictionary coefficients and make a decision. The most simple classifier that can be used is logistic regression. When there are C possible classification labels, logic regresison transforms the input vector of sparse coefficients $\mathbf{h}$ of dimension N by an affine transform with output dimension C and applies a softmax function:

$$\hat{\mathbf{y}} = \text{softmax}\big(\mathbf{Z}\mathbf{h} + \mathbf{c}\big), \tag{7.24}$$

where $\mathbf{Z} \in \mathbb{R}^{C \times N}$ and $\mathbf{c} \in \mathbb{R}^{C}$. The resulting vector $\hat{\mathbf{y}}$ corresponds to a probability mass function over the labels: $\hat{y}_c = p(y = c)$, where y is a categorical random variable with C

possible outcomes. Logistic regression training on I labeled examples $\{\mathbf{h}_i, \mathbf{y}_i\}_{i=0:I-1}$ where $\mathbf{y}_i$ are one-hot indicator vectors of the true labels is

$$\underset{\mathbf{Z},\mathbf{c}}{\text{minimize}} \quad \sum_i f_{CE}(\text{softmax}\big(\mathbf{Z}\mathbf{h}_i + \mathbf{c}\big), \mathbf{y}_i) + \lambda_2 \|\mathbf{Z}\|_2, \tag{7.25}$$

which minimizes the cross-entropy (2.25) between the predicted posterior distribution $\hat{\mathbf{y}}$ given by (7.24) and the one-hot indicator vectors $\mathbf{y}$, which are the "true" distribution. Notice the ℓ_2 regularization with weight λ_2, which prevents overfitting and learning large values in the matrix $\mathbf{Z}$. This optimization problem does not have a closed form solution, so gradient descent algorithms need to be used to solve it. Notice that the problem is convex, which means the solution of the problem will be the global minimum.

For task-driven dictionary learning, both the dictionary $\mathbf{W}$ and the classifier parameters $\{\mathbf{Z}, \mathbf{c}\}$ need to be trained. The simplest approach is to train the dictionary $\mathbf{W}$ first, then use the sparse coefficients $\mathbf{H}$ as input features and train the classifier to map $\mathbf{H}$ to $\mathbf{y}$. Mairal et al. [100] observed better results by jointly training dictionary $\mathbf{W}$ and the classifier parameters $\{\mathbf{Z}, \mathbf{c}\}$, which modifies the optimization problem (2.15) to the following problem:

$$\begin{aligned} \underset{\mathbf{W}\in\mathcal{W},\mathbf{Z},\mathbf{c}}{\text{minimize}} \quad & D_\beta(\mathbf{X}||\mathbf{W}\mathbf{H}^\star) + \lambda_1\|\mathbf{H}^\star\|_1 + \nu \sum_t \ell(\hat{\mathbf{y}}_t, \mathbf{y}_t) \\ \text{subject to} \quad & \hat{\mathbf{y}}_t = \text{softmax}\big(\mathbf{Z}\mathbf{H}^\star_{:,t} + \mathbf{c}\big) \text{ for } t = 0..T-1, \\ & \mathbf{H}^\star = \underset{\mathbf{H}\geq 0}{\text{argmin}}\ D_\beta(\mathbf{X}||\mathbf{W}\mathbf{H}) + \lambda_1\|\mathbf{H}\|_1. \end{aligned} \tag{7.26}$$

This new problem is difficult to solve directly, since $\mathbf{H}^\star$ is the optimum solution of an optimization problem, and is thus not differentiable. To get around this, Mairal et al. [99] proposed an algorithm that alternates between computing $\mathbf{H}^\star$ given the current dictionary and classifier parameters and performing a single stochastic gradient descent step to update the dictionary $\mathbf{W}$ and classifer parameters $\{\mathbf{Z}, \mathbf{c}\}$.

In this section, we will use the acoustic scene classification task of the DCASE 2016 challenge [154]. The data consists of 30-second binaural audio clips recorded in 15 different real-world environments:

1. Bus - traveling by bus in the city (vehicle)
2. Cafe / Restaurant - small cafe/restaurant (indoor)
3. Car - driving or traveling as a passenger, in the city (vehicle)
4. City center (outdoor)
5. Forest path (outdoor)
6. Grocery store - medium size grocery store (indoor)
7. Home (indoor)
8. Lakeside beach (outdoor)
9. Library (indoor)
10. Metro station (indoor)
11. Office - multiple persons, typical work day (indoor)
12. Residential area (outdoor)
13. Train (traveling, vehicle)
14. Tram (traveling, vehicle)
15. Urban park (outdoor).

The development set contains 78 30-second clips (39 minutes) for each scene, and the evaluation set contains 26 30-second clips (13 minutes). The split between development and evaluation set is based on location, which ensures that locations in the development set are

different than locations in the evaluation set. Four cross-validation fold splits of the development set are provided by the challenge. The challenge organizers provided a baseline system that uses MFCCs and GMMs. The MFCCs use 40ms windows with hop of 20ms, yielding 20 MFCC static coefficients including the 0th. Delta and delta-delta features are concatenated, for a total feature dimension of 60. The GMM uses 16 components per class. For a particular input example, the MFCC features are computed and are scored using the GMM for each scene. The assigned label corresponds to the label of the GMM that yields maximum log-likelihood.

The best-performing system in the challenge in terms of classification accuracy on the evaluation set was proposed by Eghbal-Zadeh [35]. This system is actually an ensemble of two systems. The first system uses i-vectors [30] extracted from the MFCCs of four audio channels built from the left and right channels of the audio: left, right, left-plus-right, and left-minus-right. The i-vector scores are calibrated [16] and combined by taking their mean. The second system is a convolutional neural network (CNN). These two systems are fused together by calibrating and combining their scores using a logistic regression model, as described by [16].

Bisot et al. [13] proposed task-driven sparse NMF for the acoustic scene classification problem. Their input features consists of a constant-Q magnitude spectrogram, where this spectrogram is pooled using averaging into one-second long, nonoverlapping segments. For an input audio file, the audio samples are normalized to have a maximum absolute value of 1 and the constant-Q spectrogram is normalized per frequency bin using the mean and standard deviation computed on the training set before pooling. Thus, each frequency bin is normalized to be zero-mean and unit-variance according to the training set statistics. The sparse activations $\mathbf{H}$ are inferred, and then averaged across frames. This averaged vector is fed to a logistic regression classifier, which outputs the posterior class probabilities. The class decision is made by selecting the label with the maximum posterior probability. Similar to Mairal et al.'s approach, they proposed an alternating algorithm for jointly training the dictionary $\mathbf{W}$ and classifier parameters $\{\mathbf{Z}, \mathbf{c}\}$. To further improve the performance of the

system, four instances of the system are combined using late fusion, where the four instances are combinations of two different dictionary sizes $N = 256$ and $N = 512$ and two different initializations for NMF.

Scores for these baseline approaches are summarized in table 7.4. Notice that Bisot et al.'s method only achieves the second best development and evaluation accuracy. However, the first best system by Eghbal-Zadeh et al. is an ensemble between two different systems. Thus, Bisot et al.'s task-drive sparse NMF system is the best single-model system. Because of this, the sparse coefficients from NMF seem to be very good features for a logistic classifier.

Since DR-NMF from section 7.2.3 solves the sparse NMF problem, it seems a very good fit for audio classification. We will modify DR-NMF to perform classification by taking inspiration from Bisot et al.'s method. We will adopt the same feature extraction as Bisot et al., which is constant-Q spectrogram frames pooled over one-second durations with a hop of one-second. These frames are normalized to be zero-mean and unit-variance per frequency bin using the mean and standard deviation computed from the training data. Since the hidden states of the DR-NMF network $\mathbf{h}_t$ are equivalent to the columns of the sparse NMF activation matrix $\mathbf{H}$, a pooling across all time steps is added to the output of a K-layer DR-NMF, where the number of time steps T is equal to the number of frames in the feature sequence, which we we define as

$$\bar{\mathbf{h}} = \frac{1}{T} \sum_t \mathbf{h}_t. \tag{7.27}$$

Next, an affine transform with parameters $\mathbf{Z} \in \mathbb{R}^{C \times N}$ and $\mathbf{c} \in \mathbb{R}^C$ is added to the pooled activations, where $C = 15$ is the number of classes. Finally, a softmax activation (2.24) is applied to the output of this affine layer. Thus, the output of the DR-NMF classification network for a single audio sequence is

$$\hat{\mathbf{y}} = \operatorname{softmax}\big(\mathbf{Z}\bar{\mathbf{h}} + \mathbf{c}\big), \tag{7.28}$$

where element i of $\hat{\mathbf{y}}$ is the posterior probability that the input audio sequence is of class i.

The nonnegative dictionary $\mathbf{W}$ for the DR-NMF network is initialized using by first training a sparse NMF model using well-done multiplicative updates [87]. To compare directly

to Bisot et al.'s approach, we choose a dictionary size of $N = 256$. The inverse step size α is chosen as $N/4$, which satisfies the bound (7.4) and the initial hidden state $\mathbf{h}_0$ is set to $\mathbf{0}$. Using this initial model, the hidden states are predicted by the RNN on the training set and then averaged across time to produce a single vector $\bar{\mathbf{h}}_i$ for each training example $i = 0..I-1$. Then the parameters $\mathbf{Z}$ and $\mathbf{c}$ of a logistic regression model are trained on the feature vectors $\bar{\mathbf{h}}$, where the optimization problem (7.25) is solved using the limited-memory BFGS algorithm [111, 94], a quasi-Newton method that is the default for the `sklearn` library's logistic regression implementation [120].

After initialization using sparse NMF and logistic regression, the parameters $\{\mathbf{W}, \alpha, \mathbf{h}_0\}$ are untied across layers: $\{\mathbf{W}^{(1:K)}, \alpha^{(1:K)}, \mathbf{h}_0^{(1:K)}\}$. The logistic regression parameters $\{\mathbf{Z}, \mathbf{c}\}$, which are only present at the last layer, are also included, making the trainable weights $\{\mathbf{W}^{(1:K)}, \alpha^{(1:K)}, \mathbf{h}_0^{(1:K)}, \mathbf{Z}, \mathbf{c}\}$. Using a CE loss function, the classification DR-NMF network is trained on a supervised dataset. The Adam algorithm [80] is used with a batch size of 10 and a learning rate of 10^{-4}. Through hyperparameter sweeps, a sparsity weight of $\lambda_1 = 0.1$ and a ℓ_2 regularization of $\lambda_2 = 10$ were found to yield the best cross-fold classification accuracy on the development set.

The results are shown in table 7.4. Notice that the ensemble system submitted by Eghbal-Zadeh et al. performs best both in terms of cross-fold accuracy on the development set (89.9) and classification accuracy on the evaluation set (89.7). However, the system by Bisot et al. uses only a single model, and to build an ensemble only combines instances of this model with different initializations and parameters. Note that a single instance of the proposed DR-NMF model is competitive with a single instance of Bisot et al.'s model on the development set (84.6 versus 85.0). Ensembling of the DR-NMF model would likely result in further improved scores.

Table 7.3: Source separation results on the evaluation set for the MCGMM and the deep MCGMM (DMCGMM). Units are in dB, given as SDRs of the source image (SDRim) and of the source (SDR). Results are given for various desired input SDRim. Reprinted from [161], ©2016 IEEE.

MCGMM var.	DMCGMM	Desired input SDRim in dB							
EM layers	layers	−9	−6	−3	0	3	6	9	All
		Mean output SDRim in dB							
No proc.	—	-9.61	-5.87	-3.00	-0.01	2.97	5.84	**9.51**	-0.06
10	—	-0.55	3.17	5.42	6.75	7.51	7.68	7.12	5.88
15	—	-0.60	3.18	5.48	6.83	7.60	7.75	7.14	5.94
10	1	-0.70	3.16	5.59	6.97	7.70	7.86	7.21	6.02
10	2	-0.20	3.54	5.86	7.14	7.81	7.94	7.17	6.23
10	3	**0.78**	**4.28**	**6.17**	7.18	7.61	7.53	6.57	6.32
10	4	-0.17	3.73	5.96	**7.29**	**7.96**	**8.16**	7.40	**6.37**
		Mean output SDR in dB							
No proc.[7]	—	-27.32	-23.07	-22.03	-19.74	-18.43	-17.80	-18.39	-20.50
10	—	-0.43	1.77	3.52	4.48	5.44	6.17	6.86	4.19
15	—	-0.40	1.80	3.48	4.36	5.25	5.92	6.51	4.07
10	1	0.34	2.46	4.10	4.93	5.76	6.34	6.88	4.63
10	2	0.82	2.89	**4.49**	**5.27**	**6.07**	**6.59**	**7.07**	**4.97**
10	3	**1.11**	**3.01**	4.43	5.08	5.74	6.18	6.47	4.80
10	4	0.88	2.88	4.36	5.13	5.89	6.45	6.96	4.85

Table 7.4: Classification accuracies of various systems on the DCASE0216 scene classification dataset. Dev. accuracy indicates mean classification accuracy across the four folds of the development set, and eval. accuracy indicates classification accuracy on the evaluation set. Ensemble methods are indicated by "ens." The evaluation scores for the proposed methods are computed by summing the log-likelihoods of the four systems trained on the four cross-validation development set folds.

	Method	**Dev. accuracy**	**Eval. accuracy**
Baselines	Challenge baseline: MFCCs+GMM	72.5	77.2
	Eghbal-Zadeh et al. [35]: CNN	79.5	83.3
	Eghbal-Zadeh et al. [35]: i-vectors (ens.)	83.9	88.7
	Eghbal-Zadeh et al. [35]: i-vectors+CNN (ens.)	89.9	89.7
	Bisot et al. [13]: task-driven SNMF	85.0	–
	Bisot et al. [13]: task-driven SNMF (ens.)	86.2	87.7
Proposed	DR-NMF ($K = 10$, CE loss)	83.8	85.9
	DR-NMF ($K = 20$, CE loss)	84.5	86.2
	DR-NMF ($K = 30$, CE loss)	84.6	85.9

Chapter 8

CONCLUSION

In this dissertation, we have improved modeling and processing of nonstationary signals. Two main approaches were employed: reexamining conventional assumptions of nonstationary signals, and deep unfolding of statistical models of nonstationary signals into deep networks. This chapter summarizes the main contributions of the dissertation and briefly discusses future work.

8.1 Main Contributions

8.1.1 Contribution 1: Improving Statistical Models of Nonstationary Signals Using Noncircularity

In chapter 3, we described the properties of noncircular complex-valued Gaussian distribution models of the STFT. The theoretical noncircularity of the STFT was studied for several classes of nonstationary signals. Then, practical estimators were proposed for the statistical parameters of these distributions directly from the STFT of a nonstationary signal. Two estimators were proposed: averaging over time within subbands and averaging over multitapers. The properties of these estimators were empirically studied using tones and transients, which are fundamental components of nonstationary signals.

The estimators and insight from chapter 3 were then applied to realistic detection tasks of nonstationary signals in chapter 4. These realistic tasks included voice activity detection and detection of acoustic transients. In both cases, noncircularity-based detectors improved detection performance compared to conventional detectors that assume complex-valued STFT coefficients of nonstationary signals are circular.

8.1.2 Contribution 2: Improving Unitary Recurrent Neural Networks for Nonstationary Signals

Chapter 5 examined a recently-proposed neural network model for processing nonstationary signals: the unitary recurrent neural network. We showed that the original proposed parameterization of the unitary matrix in this model was deficient, in the sense that it cannot provably represent all possible unitary matrices. To remedy this problem, we proposed a method of optimizing the unitary recurrence matrix over the entire set of unitary matrices, which we refer to as full-capacity optimization. Full-capacity uRNNs were compared to the conventional restricted-capacity uRNNs and state-of-the-art LSTMs on several benchmark tasks that test the ability of RNNs to process nonstationary signals, including long-term memory, spectrogram prediction, and classification of pixel-by-pixel handwritten MNIST digits. The full-capacity uRNNs achieved competitive or superior performance on all these tasks, which indicates that the full-capacity uRNN has improved capability to model and process nonstationary signals.

8.1.3 Contribution 3: Unfolding Statistical Models of Nonstationary Signals to Deep Networks

The third contribution of this dissertation is the extension of the deep unfolding framework originally proposed by Hershey et al. [65]. In particular, in chapter 6, after reviewing existing instances of unfolding statistical model inference to deep networks, we showed how a sequential version of the iterative soft-thresholding algorithm (ISTA) for sparse coding of a sequence of input vectors unfolds to a novel RNN, the SISTA-RNN. If a nonnegativity constraint is applied to the dictionary and sparse coefficients in the sparse coding model, then the SISTA-RNN is equivalent to a deep recurrent NMF (DR-NMF) network, which produces better solutions to the sparse NMF optimization problem (2.15) by exploiting "warm-start" initialization connections between adjacent time steps. We also described reverse unfolding of a network to a statistical model. Using the knowledge of MRF and sparse coding unfolding,

we were able to provide a model-based explantion for gated recurrent unit (GRU) RNNs, which are a state-of-the-art neural network architecture. We also gave a reverse unfolding of the unitary RNN, which is equivalent to a complex-valued SISTA-RNN with one iteration, $K = 1$, at each time step and with an additional constraint on the singular values of the complex-valued sparse coding dictionary.

In chapter 7, we applied deep unfolding to specific nonstationary signal processing tasks which illustrate the benefits of deep unfolded networks. First, we used a column-wise image compressed sensing task to compare a SISTA-RNN with both conventional statistical model inference using sequential ISTA as well as state-of-the-art neural networks including RNNs and LSTMs. The SISTA-RNN can be initialized using a sparse coding model and was able to achieve the best recovery performance overall, while also maintaining the interpretability of its weights during training.

Next, DR-NMF networks were tested on single-channel audio source separation. On this task, DR-NMF networks outperformed the conventional statistical model of NMF, but achieved slightly lower performance compared to LSTM networks. However, DR-NMF networks did achieve the best performance overall when only 10% of the training data is available, which indicates they may be more effective when trained with less data. We hypothesize that the model-based initialization helps the network's weights start training from an advantageous setting. DR-NMF networks also maintain their interpretability after training, in that the weights of the network correspond directly to model and algorithm parameters, including iteration-dependent inverse step sizes $\alpha^{(k)}$ for ISTA and NMF dictionaries $\mathbf{W}^{(k)}$.

We also applied deep unfolding to multichannel (i.e., multimicrophone) and multispeaker audio source separation. In this case, we unfolded a factorial GMM model, where spectrograms of audio sources were modeled by a GMM, observed through a narrowband frequency-domain channel matrix, and the state responsibilities of each GMM were modeled using a Markov random field. By unfolding the variational EM algorithm for this statistical model, a novel deep network was created that can directly process complex-valued multichannel STFTs. We found that the deep unfolded model could be trained to achieve better separa-

tion performance versus the baseline statistical model inference algorithm.

DR-NMF networks are also tested on the task of audio scene classification. Since the second-best submitted system in the DCASE 2016 acoustic scene classification challenge was a task-driven sparse NMF model, we anticipated that DR-NMF networks would be effective, since they solve the NMF problem and can be trained to classify using a cross-entropy training loss. Indeed, DR-NMF networks were nearly able to achieve the performance of the task-driven NMF system, and we are optimistic that further hyperparameter tuning and ensembling will continue to improve performance.

8.2 Future Work

In this section, future work is organized by two topics: improving statistical models and unfolding statistical models.

8.3 Improving Statistical Models

For the noncircular extensions of the conventional model of the STFT, only noncircular complex-valued Gaussians were considered. But other types of noncircular distributions could also be considered, and some recent work has shown that heavy-tailed distribution models of the STFT can improve processing of nonstationary speech signals. For example, Jukić and Doclo [74] derived a WPE algorithm under a Laplacian residual distribution, and observed improved performance. Liutkus *et al.* [95] derived nonnegative matrix factorization under a Cauchy model, and Şimşekli *et al.* [169] with alpha-stable distributions; both observed improved speech separation performance using these models. The class of noncircular elliptical distributions include these distributions as special cases [130], and can further exploit noncircularity.

Multichannel extensions of noncircular models may be very useful because of the blind source separation applications provided by noncircularity [130]. Okopal et al. [114, 113] demonstrated initial success recovering speech information in the presense of very loud noise under a synthetic channel configuration. The extent to which this principle can work with

realistic frequency-domain channel matrices is an open question.

8.4 Unfolding Statistical Models

Many interesting extensions of the deep unfolded networks we described in this dissertation are possible. Because they are based on statistical models, any modification to the model manifests as a modification to the network architecture. Using FISTA or AMP instead of ISTA in the SISTA-RNN and DR-NMF networks could potentially improve the performance of the networks, since these algorithms improve the convergence of ISTA. It would also be interesting to add better models of dynamics, especially to the DR-NMF model, using, e.g., the nonnegative dynamical systems of Fevotte et al. [43]. Another potentially effective dynamical model would replace the warm-starts in the DR-NMF network with GRU-style gates, which we now understand to be iterations of mean field inference in pairwise Markov random fields. Using the correspondence of GRUs with mean field inference, additional iterations can be added to the mean field iterations, which would embed deep networks within the recurrent neural networks. This would be a model-based explanation for the various network-in-network architectures that have recently been proposed [93].

There are many other statistical models and related algorithms that are ripe to be unfolded. As the number of deep unfolding conversions grows, we anticipate that the community's understanding of the unreasonable effectiveness of existing deep network architectures will also increase. Also, as more statistical models are unfolded, novel and effective deep network architectures will be discovered through principled means, potentially speeding up the already rapid progress of artifical intelligence. We hope that the reader is inspired to try unfolding statistical models used in their own particular area and reap the benefits of combining domain knowledge and model-based thinking with the data-driven adaptation of backpropagation and deep learning.

BIBLIOGRAPHY

[1] C. Ahuja, K. Nathwani, and R. M. Hegde. A Complex Matrix Factorization approach to Joint Modeling of Magnitude and Phase for Source Separation. *arXiv:1411.6741 [cs]*, November 2014. arXiv: 1411.6741.

[2] M. Arjovsky, A. Shah, and Y. Bengio. Unitary Evolution Recurrent Neural Networks. *arXiv:1511.06464 [cs]*, November 2015. arXiv: 1511.06464.

[3] M. Arjovsky, A. Shah, and Y. Bengio. Unitary Evolution Recurrent Neural Networks. In *International Conference on Machine Learning (ICML)*, June 2016.

[4] M. S. Asif, A. Charles, J. Romberg, and C. Rozell. Estimation and dynamic updating of time-varying signals with sparse variations. In *Proc. ICASSP*, pages 3908–3911, Prague, Czech Republic, May 2011.

[5] M. S. Asif and J. Romberg. Sparse Recovery of Streaming Signals Using ℓ_1-Homotopy. *IEEE Transactions on Signal Processing*, 62(16):4209–4223, August 2014.

[6] H. Attias. New EM algorithms for source separation and deconvolution with a microphone array. In *Proc. IEEE ICASSP*, volume 5, pages V–297–300 vol.5, April 2003.

[7] R. Badeau. Gaussian modeling of mixtures of non-stationary signals in the Time-Frequency domain (HR-NMF). In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 253–256, October 2011.

[8] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[9] A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, January 2009.

[10] J. Benesty, J. Chen, and Y. Huang. A Widely Linear Distortionless Filter for Single-Channel Noise Reduction. *IEEE Signal Processing Letters*, 17(5):469 –472, May 2010.

[11] J. Benesty, J. Chen, and Y. (Arden) Huang. On widely linear Wiener and tradeoff filters for noise reduction. *Speech Communication*, 52(5):427–439, May 2010.

[12] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[13] V. Bisot, R. Serizel, S. Essid, and G. Richard. Supervised Nonnegative Matrix Factorization for Acoustic Scene Classification. 2016.

[14] M. Borgerding, P. Schniter, and S. Rangan. AMP-Inspired Deep Networks for Sparse Linear Inverse Problems. *IEEE Transactions on Signal Processing*, 65(16):4293–4308, August 2017.

[15] M. Brookes. VOICEBOX: Speech Processing Toolbox for MATLAB. [Online]. Available: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html, 2002.

[16] N. Brummer. *Measuring, refining and calibrating speaker and language information extracted from speech*. PhD thesis, Stellenbosch: University of Stellenbosch, 2010.

[17] A. Chambolle, R. A. D. Vore, N.-Y. Lee, and B. J. Lucier. Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage. *IEEE Transactions on Image Processing*, 7(3):319–335, March 1998.

[18] J. Chen, J. He, Y. Shen, L. Xiao, X. He, J. Gao, X. Song, and L. Deng. End-to-end Learning of LDA by Mirror-Descent Back Propagation over a Deep Architecture. *arXiv:1508.03398 [cs]*, August 2015. arXiv: 1508.03398.

[19] S. Chen, D. Donoho, and M. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Review*, 43(1):129–159, January 2001.

[20] Y. Chen, Y. Gu, and A. O. Hero. Sparse LMS for system identification. In *Proc. ICASSP*, pages 3125–3128, April 2009.

[21] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[22] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[23] P. Clark. *Coherent Demodulation of Nonstationary Random Processes*. PhD thesis, University of Washington, 2012.

[24] P. Clark, I. Kirsteins, and L. Atlas. Complementary envelope estimation for frequency-modulated random signals. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 5363–5367. IEEE, 2013.

[25] I. Cohen. Optimal speech enhancement under signal presence uncertainty using log-spectral amplitude estimator. *IEEE Signal Processing Letters*, 9(4):113–116, 2002.

[26] T. Cooijmans, N. Ballas, C. Laurent, and A. Courville. Recurrent Batch Normalization. *arXiv:1603.09025*, 2016.

[27] C. V. Cotton and D. P. Ellis. Spectral vs. spectro-temporal features for acoustic event detection. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on*, pages 69–72. IEEE, 2011.

[28] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[29] D. B. Dean, S. Sridharan, R. J. Vogt, and M. W. Mason. The QUT-NOISE-TIMIT corpus for the evaluation of voice activity detection algorithms. In *Proc. Interspeech*, Makuhari, Japan, September 2010.

[30] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

[31] J.-P. Delmas, A. Oukaci, and P. Chevalier. On the asymptotic distribution of GLR for impropriety of complex signals. *Signal Processing*, 91(10):2259–2267, October 2011.

[32] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[33] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[34] J. Eggert and E. Korner. Sparse coding and NMF. In *2004 IEEE International Joint Conference on Neural Networks, 2004. Proceedings*, volume 4, pages 2529–2533 vol.4, July 2004.

[35] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer. Cp-jku submissions for dcase-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks. 2016.

[36] S. El Hihi and Y. Bengio. Hierarchical Recurrent Neural Networks for Long-Term Dependencies. In *NIPS*, volume 400, page 409. Citeseer, 1995.

[37] Y. Ephraim and D. Malah. Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 32(6):1109–1121, December 1984.

[38] Y. Ephraim and D. Malah. Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(2):443–445, 1985.

[39] J. Eriksson and V. Koivunen. Complex random vectors and ICA models: identifiability, uniqueness, and separability. *IEEE Transactions on Information Theory*, 52(3):1017–1029, March 2006.

[40] S. Ewert and M. Muller. Using score-informed constraints for NMF-based source separation. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 129–132, March 2012.

[41] A. Farina. Simultaneous measurement of impulse response and distortion with a swept-sine technique. In *Audio Engineering Society Convention 108*. Audio Engineering Society, 2000.

[42] C. Févotte, N. Bertin, and J.-L. Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830, 2009.

[43] C. Févotte, J. Le Roux, and J. R. Hershey. Non-negative dynamical system with application to speech and audio. In *Proc. ICASSP*, pages 3158–3162, Vancouver, Canada, 2013.

[44] M. A. T. Figueiredo and R. D. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, August 2003.

[45] W. M. Fisher, G. R. Doddington, and K. M. Goudie-Marshall. The DARPA speech recognition research database: specifications and status. In *Proc. DARPA Workshop on Speech Recognition*, pages 93–99, 1986.

[46] J. Fritsch and M. Plumbley. Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 888–891, May 2013.

[47] J. Ganseman, P. Scheunders, and S. Dixon. Improving PLCA-based score-informed source separation with invertible Constant-Q Transforms. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 2634–2638, 2012.

[48] W. A. Gardner. *Statistical spectral analysis: a nonprobabilistic theory*. Prentice Hall, Englewood Cliffs, N.J., 1988.

[49] W. A. Gardner, A. Napolitano, and L. Paura. Cyclostationarity: Half a century of research. *Signal Processing*, 86(4):639–697, April 2006.

[50] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett. DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon Technical Report N*, 93, 1993.

[51] P. Garrigues and L. E. Ghaoui. An Homotopy Algorithm for the Lasso with Online Observations. In *Advances in Neural Information Processing Systems*, pages 489–496, 2008.

[52] F. Germain, D. L. Sun, and G. J. Mysore. Speaker and noise independent voice activity detection. In *Proc. Interspeech*, pages 732–736, 2013.

[53] H. Ghaemmaghami, B. J. Baker, R. J. Vogt, and S. Sridharan. Noise robust voice activity detection using features extracted from the time-domain autocorrelation function. In *Proc. Interspeech*, Makuhari, Japan, 2010.

[54] R. Gilmore. *Lie groups, physics, and geometry: an introduction for physicists, engineers and chemists*. Cambridge University Press, 2008.

[55] E. G. Gladyshev. Periodically and Almost-Periodically Correlated Random Processes with a Continuous Time Parameter. *Theory of Probability & Its Applications*, 8(2):173–177, January 1963.

[56] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, volume 9, pages 249–256, 2010.

[57] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[58] G. H. Golub and C. F. van Loan. Matrix computations. *Matrix computations/Gene H. Golub, Charles F. Van Loan. Baltimore: Johns Hopkins University Press, 1996.(Johns Hopkins studies in the mathematical sciences)*, 1996.

[59] E. Grais and H. Erdogan. Initialization of nonnegative matrix factorization dictionaries for single channel source separation. In *Signal Processing and Communications Applications Conference (SIU), 2013 21st*, pages 1–4, April 2013.

[60] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 399–406, 2010.

[61] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. techreport CNS-TR-2007-001, California Institute of Technology, 2007.

[62] E. Habets, J. Benesty, I. Cohen, S. Gannot, and J. Dmochowski. New Insights Into the MVDR Beamformer in Room Acoustics. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):158–170, 2010.

[63] A. K. Halberstadt. *Heterogeneous acoustic measurements and multiple classifiers for speech recognition*. PhD thesis, Massachusetts Institute of Technology, 1998.

[64] T. Heittola, A. Mesaros, T. Virtanen, and A. Eronen. Sound event detection in multisource environments using source separation. *Proc CHiME*, pages 36–40, 2011.

[65] J. R. Hershey, J. L. Roux, and F. Weninger. Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures. *arXiv:1409.2574 [cs, stat]*, September 2014. arXiv: 1409.2574.

[66] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[67] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[68] Y. Hoshen, R. J. Weiss, and K. W. Wilson. Speech Acoustic Modeling from Raw Multichannel Waveforms. In *Proc. IEEE ICASSP*, Brisbane, Australia, April 2015.

[69] A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM*, 5(4):339–342, 1958.

[70] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, February 2015.

[71] ITU-T P.862. Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs, 2000.

[72] R. Jaiswal, D. Fitzgerald, E. Coyle, and S. Rickard. Shifted nmf using an efficient constant-q transform for monaural sound source separation. *ISSC, Dublin*, 2011.

[73] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, November 1999.

[74] A. Jukić and S. Doclo. Speech dereverberation using weighted prediction error with Laplacian model of the desired signal. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5172–5176, May 2014.

[75] H. Kameoka, N. Ono, K. Kashino, and S. Sagayama. Complex NMF: A new sparse representation for acoustic signals. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3437–3440. IEEE, 2009.

[76] U. S. Kamilov and H. Mansour. Learning Optimal Nonlinearities for Iterative Thresholding Algorithms. *IEEE Signal Processing Letters*, 23(5):747–751, May 2016.

[77] U. S. Kamilov, H. Mansour, and D. Liu. Learning Convolutional Proximal Filters. In *Proc. SPARS*, June 2017.

[78] U. S. Kamilov and H. Mansour. Learning MMSE Optimal Thresholds for FISTA. In *Proc. iTWIST*, August 2016.

[79] B. J. King and L. Atlas. Single-Channel Source Separation Using Complex Matrix Factorization. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2591–2597, November 2011.

[80] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, December 2014.

[81] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[82] K. Kinoshita, M. Delcroix, T. Yoshioka, T. Nakatani, E. Habets, R. Haeb-Umbach, V. Leutnant, A. Sehr, W. Kellermann, and R. Maas. The REVERB challenge: A common evaluation framework for dereverberation and recognition of reverberant speech. In *Proc. IEEE WASPAA*, New Paltz, NY, 2013.

[83] D. Kitamura, H. Saruwatari, S. Nakamura, Y. Takahashi, K. Kondo, and H. Kameoka. Divergence optimization in nonnegative matrix factorization with spectrogram restoration for multichannel signal separation. In *2014 4th Joint Workshop on Hands-free Speech Communication and Microphone Arrays (HSCMA)*, pages 92–96, May 2014.

[84] K. Kreutz-Delgado. The Complex Gradient Operator and the CR-Calculus. *arXiv:0906.4835 [math]*, June 2009. arXiv: 0906.4835.

[85] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv:1504.00941*, 2015.

[86] J. Le Roux, J. R. Hershey, and F. J. Weninger. Deep NMF for Speech Enhancement. In *Proc. IEEE ICASSP*, Brisbane, Australia, 2015.

[87] J. Le Roux, F. J. Weninger, and J. R. Hershey. Sparse NMF–half-baked or well done? *MERL Technical Report*, 2015.

[88] C. H. Lee and J. T. Chien. Deep unfolding inference for supervised topic model. In *Proc. ICASSP*, pages 2279–2283, Shanghai, China, March 2016.

[89] D. D. Lee and H. S. Seung. Algorithms for Non-negative Matrix Factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001.

[90] S. Lee, S. H. Park, and K.-M. Sung. Beamspace-Domain Multichannel Nonnegative Matrix Factorization for Audio Source Separation. *IEEE Signal Processing Letters*, 19(1):43–46, January 2012.

[91] A. Lefevre, F. Bach, and C. Fevotte. Itakura-Saito nonnegative matrix factorization with group sparsity. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 21–24, May 2011.

[92] Q. Li and N. Lin. The Bayesian elastic net. *Bayesian Analysis*, 5(1):151–170, March 2010.

[93] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[94] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[95] A. Liutkus, D. Fitzgerald, and R. Badeau. Cauchy nonnegative matrix factorization. In *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5, October 2015.

[96] M. Loève. *Probability Theory*. Springer-Verlag, New York, 1977.

[97] P. C. Loizou. *Speech Enhancement: Theory and Practice*. CRC Press, Boca Raton, FL, June 2007.

[98] P. Magron, R. Badeau, and B. David. Complex NMF under phase constraints based on signal modeling: application to audio source separation. In *Proc. ICASSP*, Shanghai, China, 2016.

[99] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):791–804, 2012.

[100] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach. Supervised dictionary learning. In *Advances in neural information processing systems*, pages 1033–1040, 2009.

[101] D. M. Malioutov, S. R. Sanghavi, and A. S. Willsky. Sequential Compressed Sensing. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):435–444, April 2010.

[102] R. Martin. Noise power spectral density estimation based on optimal smoothing and minimum statistics. *IEEE Transactions on Speech and Audio Processing*, 9(5):504–512, July 2001.

[103] A. Mesaros, T. Heittola, O. Dikmen, and T. Virtanen. Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 606–618. IEEE Computer Society, 2015.

[104] F. Millioz and N. Martin. Circularity of the STFT and Spectral Kurtosis for Time-Frequency Segmentation in Gaussian Environment. *IEEE Transactions on Signal Processing*, 59(2):515–524, 2011.

[105] Y. Murakami, M. Yamagishi, M. Yukawa, and I. Yamada. A sparse adaptive filtering using time-varying soft-thresholding techniques. In *Proc. ICASSP*, pages 3734–3737, Dallas, TX, March 2010.

[106] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, MA, August 2012.

[107] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML*, pages 807–814, 2010.

[108] A. Napolitano. *Generalizations of Cyclostationary Signal Processing: Spectral Analysis and Applications*. John Wiley & Sons, August 2012.

[109] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[110] N. Nichols. *Marine mammal species detection and classification*. PhD, University of Washington, 2015.

[111] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[112] P. D. O'Grady and B. A. Pearlmutter. Convolutive non-negative matrix factorisation with a sparseness constraint. In *Proc. MLSP*, pages 427–432, Maynooth, Ireland, 2006.

[113] G. Okopal, S. Wisdom, and L. Atlas. Speech Analysis With the Strong Uncorrelating Transform. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(11):1858–1868, November 2015.

[114] G. Okopal, S. Wisdom, and L. Atlas. Estimating the Noncircularity of Latent Components within Complex-Valued Subband Mixtures with Applications to Speech Processing. In *Proc. Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, November 2014.

[115] A. Ozerov and C. Fevotte. Multichannel nonnegative matrix factorization in convolutive mixtures. With application to blind audio source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009*, pages 3137–3140, April 2009.

[116] A. Ozerov and C. Fe?votte. Multichannel Nonnegative Matrix Factorization in Convolutive Mixtures for Audio Source Separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):550–563, March 2010.

[117] A. Ozerov, E. Vincent, and F. Bimbot. A General Flexible Framework for the Handling of Prior Information in Audio Source Separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(4):1118–1133, May 2012.

[118] H. Palangi, R. Ward, and L. Deng. Exploiting correlations among channels in distributed compressive sensing with convolutional deep stacking networks. In *Proc. ICASSP*, pages 2692–2696, Shanghai, China, March 2016.

[119] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to Construct Deep Recurrent Neural Networks. *arXiv:1312.6026 [cs, stat]*, December 2013.

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

[121] D. B. Percival and A. T. Walden. *Spectral Analysis for Physical Applications*. Cambridge University Press, June 1993.

[122] J. Ramírez, J. C. Segura, C. Benítez, Á. de la Torre, and A. Rubio. Efficient voice activity detection algorithms using long-term speech information. *Speech Communication*, 42:271–287, 2004.

[123] B. Rivet, L. Girin, and C. Jutten. Log-Rayleigh Distribution: A Simple and Efficient Statistical Representation of Log-Spectral Coefficients. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):796–802, March 2007.

[124] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *Proc. ICASSP*, volume 2, pages 749–752, 2001.

[125] T. Robinson, J. Fransen, D. Pye, J. Foote, and S. Renals. Wsjcam0: A British English Speech Corpus For Large Vocabulary Continuous Speech Recognition. In *Proc. IEEE ICASSP*, pages 81–84, Detroit, MI, 1995.

[126] J. T. Rolfe and Y. LeCun. Discriminative recurrent sparse auto-encoders. *arXiv:1301.3775*, 2013.

[127] A. Sard. The measure of the critical values of differentiable maps. *Bulletin of the American Mathematical Society*, 48(12):883–890, 1942.

[128] H. Sawada, H. Kameoka, S. Araki, and N. Ueda. Multichannel Extensions of Non-Negative Matrix Factorization With Complex-Valued Data. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):971–982, May 2013.

[129] J. Schmidhuber. Learning Complex, Extended Sequences Using the Principle of History Compression. *Neural Computation*, 4(2):234–242, March 1992.

[130] P. J. Schreier and L. L. Scharf. *Statistical Signal Processing of Complex-Valued Data: The Theory of Improper and Noncircular Signals*. Cambridge University Press, February 2010.

[131] P. Schreier and L. Scharf. Stochastic time-frequency analysis using the analytic signal: why the complementary distribution matters. *IEEE Transactions on Signal Processing*, 51(12):3071–3079, 2003.

[132] M. L. Seltzer, D. Yu, and Y. Wang. An investigation of deep neural networks for noise robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7398–7402. IEEE, 2013.

[133] U. Simsekli, J. Le Roux, and J. Hershey. Non-negative source-filter dynamical system for speech enhancement. In *Proc. IEEE ICASSP*, pages 6206–6210, Florence, Italy, May 2014.

[134] P. Smaragdis, C. Fevotte, G. Mysore, N. Mohammadiha, and M. Hoffman. Static and Dynamic Source Separation Using Nonnegative Factorizations: A unified view. *IEEE Signal Processing Magazine*, 31(3):66–75, May 2014.

[135] P. Smaragdis. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In *Independent Component Analysis and Blind Signal Separation*, pages 494–499. Springer, 2004.

[136] J. Sohn, N. S. Kim, and W. Sung. A statistical model-based voice activity detection. *IEEE Signal Processing Letters*, 6(1):1–3, January 1999.

[137] P. Sprechmann, A. M. Bronstein, and G. Sapiro. Learning Efficient Sparse and Low Rank Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1821–1833, September 2015.

[138] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[139] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. Plumbley. Detection and Classification of Acoustic Scenes and Events. *IEEE Transactions on Multimedia*, 17(10):1733–1746, October 2015.

[140] D. L. Sun and G. J. Mysore. Universal speech models for speaker independent single channel source separation. In *Proc. ICASSP*, pages 141–145, Vancouver, Canada, 2013.

[141] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[142] C. Taal, R. Hendriks, R. Heusdens, and J. Jensen. An algorithm for intelligibility prediction of time-frequency weighted noisy speech. *IEEE Trans. on Audio, Speech, and Language Processing*, 19(7):2125–2136, September 2011.

[143] H. D. Tagare. Notes on optimization on Stiefel manifolds. Technical report, Yale University, 2011.

[144] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv: 1605.02688*, May 2016.

[145] D. Thomson. Spectrum estimation and harmonic analysis. *Proceedings of the IEEE*, 70(9):1055–1096, September 1982.

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

[147] T. Tieleman and G. Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude, 2012. Published: Coursera: Neural Networks for Machine Learning.

[148] E. Variani, E. McDermott, and G. Heigold. A Gaussian Mixture Model Layer Jointly Optimized with Discriminative Features within a Deep Neural Network Archtecture. In *Proc. IEEE ICASSP*, Brisbane, Australia, 2015.

[149] N. Vaswani. Kalman filtered Compressed Sensing. In *Proc. (ICIP)*, pages 893–896, October 2008.

[150] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462 –1469, July 2006.

[151] E. Vincent. BSS Eval toolbox version 3.0. `http://bass-db.gforge.inria.fr/bss_eval`.

[152] E. Vincent, S. Araki, F. Theis, G. Nolte, P. Bofill, H. Sawada, A. Ozerov, V. Gowreesunker, D. Lutter, and N. Q. Duong. The signal separation evaluation campaign (2007–2010): Achievements and remaining challenges. *Signal Processing*, 92(8):1928–1936, 2012.

[153] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni. The second 'CHiME' speech separation and recognition challenge: An overview of challenge systems and outcomes. In *Proc. ASRU*, pages 162–167, Olomouc, Czech Republic, 2013.

[154] T. Virtanen, A. Mesaros, T. Heittola, M. Plumbley, P. Foster, E. Benetos, and M. Lagrange. *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*. Tampere University of Technology. Department of Signal Processing, 2016.

[155] P. D. Welch. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on audio and electroacoustics*, 15(2):70–73, 1967.

[156] F. Weninger, H. Erdogan, S. Watanabe, E. Vincent, J. Le Roux, J. R. Hershey, and B. Schuller. Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR. In *Latent Variable Analysis and Signal Separation*, pages 91–99. Springer, 2015.

[157] S. Wisdom, L. Atlas, J. Pitton, and G. Okopal. Benefits of noncircular statistics for nonstationary signals. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 554–558, November 2016.

[158] S. Wisdom, T. Powers, J. Pitton, and L. Atlas. Building recurrent networks by unfolding iterative thresholding for sequential sparse recovery. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4346–4350, March 2017.

[159] S. Wisdom, L. Atlas, and J. Pitton. On Spectral Noncircularity of Natural Signals. In *submitted to Sensor and Multichannel (SAM) 2016.*, 2016.

[160] S. Wisdom, L. Atlas, and J. Pitton. On spectral noncircularity of natural signals. In *Sensor Array and Multichannel Signal Processing Workshop (SAM), 2016 IEEE*, pages 1–5. IEEE, 2016.

[161] S. Wisdom, J. Hershey, J. Le Roux, and S. Watanabe. Deep Unfolding for Multichannel Source Separation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[162] S. Wisdom, G. Okopal, L. Atlas, and J. Pitton. Voice Activity Detection Using Subband Noncircularity. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brisbane, Austrailia, April 2015.

[163] S. Wisdom, T. Powers, L. Atlas, and J. Pitton. Enhancement of Reverberant and Noisy Speech by Extending Its Coherence. In *Proceedings of the REVERB Challenge Workshop*, Florence, Italy, May 2014.

[164] S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas. Full-Capacity Unitary Recurrent Neural Networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4880–4888. Curran Associates, Inc., 2016.

[165] S. Wisdom, T. Powers, J. Pitton, and L. Atlas. Interpretable Recurrent Neural Networks Using Sequential Sparse Recovery. In *NIPS Workshop on Interpretable Machine Learning for Complex Systems, arXiv:1611.07252*, December 2016.

[166] S. Wisdom, T. Powers, J. Pitton, and L. Atlas. Deep Recurrent NMF for Speech Separation by Unfolding Iterative Thresholding. In *Proc. IEEE WASPAA*, New Paltz, NY, October 2017.

[167] S. J. Wright, R. D. Nowak, and M. A. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

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

[169] U. Şimşekli, A. Liutkus, and A. T. Cemgil. Alpha-Stable Matrix Factorization. *IEEE Signal Processing Letters*, 22(12):2289–2293, December 2015.

Appendix A
DERIVATION OF ISTA

In this appendix we derive the ISTA algorithms for nonsequential and sequential sparse recovery.

A.1 ISTA for Nonsequential Sparse Recovery

Here we review the derivation of an iterative shrinkage and thresholding algorithm (ISTA) for any optimization problem of the form

$$\min_{\mathbf{h}} \quad L(\mathbf{h}) + \lambda R(\mathbf{h}), \tag{A.1}$$

of which (2.17) is an example with $L(\mathbf{h}) = \frac{1}{2}\|\mathbf{x}-\mathbf{ADh}\|_2^2$ and $R(\mathbf{h}) = \|\mathbf{h}\|_1$. Our presentation follows Wright et al. [167].

Often problems of the form (A.1) can be difficult to solve, such as the ℓ_1-regularized ℓ_2 problems we consider in this dissertation. ISTA provides a solution to problem (A.1) using an iterative sequence of subproblems, each of which can be solved efficiently:

$$\mathbf{h}^{(k)} = \operatorname*{argmin}_{\mathbf{h}} \quad (\mathbf{h} - \mathbf{h}^{(k-1)})^{\mathsf{T}}\mathbf{G}(\mathbf{h}^{(k-1)}) + \frac{\alpha}{2}\|\mathbf{h} - \mathbf{h}^{(k-1)}\|_2^2 + \lambda R(\mathbf{h}), \tag{A.2}$$

where $\mathbf{G}(\mathbf{h}^{(k-1)}) = \nabla L(\mathbf{h}^{(k-1)})$ is the gradient of the loss function at the previous estimate $\mathbf{h}^{(k-1)}$. An equivalent form of (A.2) is

$$\mathbf{h}^{(k)} = \operatorname*{argmin}_{\mathbf{h}} \quad \frac{1}{2}\|\mathbf{h} - \mathbf{u}^{(k-1)}\|_2^2 + \frac{\lambda}{\alpha}R(\mathbf{h}), \tag{A.3}$$

where

$$\mathbf{u}^{(k-1)} = \mathbf{h}^{(k-1)} - \frac{1}{\alpha}\mathbf{G}(\mathbf{h}^{(k-1)}). \tag{A.4}$$

For the specific choice of $L(\mathbf{h}) = \frac{1}{2}\|\mathbf{x} - \mathbf{ADh}\|_2^2$, the gradient is

$$\mathbf{G}(\mathbf{h}) = -\mathbf{D}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}(\mathbf{x} - \mathbf{ADh}). \tag{A.5}$$

When $R(\mathbf{h}) = \|\mathbf{h}\|_1 = \sum_{n=1}^{N} |h_n|$, it is not differentiable at any $h_n = 0$, and thus has subdifferential

$$\frac{\delta}{\delta h_n} R(\mathbf{h}) = \begin{cases} -1, & h_n < 0 \\ [-1, 1], & h_n = 0 \\ 1, & h_n > 0, \end{cases} \tag{A.6}$$

where $[a, b]$ denotes a continuous range from a to b. Combining (A.3-A.6), taking the gradient with respect to $\mathbf{h}^{(k)}$, setting this gradient equal to 0, and solving for $\mathbf{h}^{(k)}$ yields

$$\mathbf{h}^{(k)} = \text{soft}_{\lambda/\alpha}\left((\mathbf{I} - \frac{1}{\alpha}\mathbf{D}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{D})\mathbf{h}^{(k-1)} + \frac{1}{\alpha}\mathbf{D}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{x} \right), \tag{A.7}$$

which is the ISTA update step in algorithm 1.

A.2 ISTA for Sequential Sparse Recovery

Here we derive ISTA to solve the optimization problem (6.7) for sequential sparse recovery. Using the variables

$$\bar{\mathbf{D}} = \begin{bmatrix} \mathbf{A}\mathbf{D} \\ -\sqrt{\lambda_2}\mathbf{D} \end{bmatrix}, \quad \bar{\mathbf{x}}_t = \begin{bmatrix} \mathbf{x}_t \\ -\sqrt{\lambda_2}\mathbf{F}\mathbf{D}\mathbf{h}_{t-1} \end{bmatrix}, \tag{A.8}$$

we can write the problem (6.7) in an equivalent form

$$\min_{\mathbf{h}_{1:T}} \quad \sum_{t=1}^{\mathsf{T}} \Big(\frac{1}{2}\big\|\bar{\mathbf{x}}_t - \bar{\mathbf{D}}\mathbf{h}_t\big\|_2^2 + \lambda_1 \big\|\mathbf{h}_t\big\|_1 \Big). \tag{A.9}$$

Problem (A.9) is of similar form to problem (2.17) for each time step. Thus, for each time step t, given that we have computed an estimate $\hat{\mathbf{h}}_{t-1}$ for the previous time step, we can plug in the quantities $\bar{\mathbf{D}}$ and $\bar{\mathbf{x}}_t$ from (A.8) for $\mathbf{A}\mathbf{D}$ and $\mathbf{x}$ within the ISTA update (A.7). If we use the estimate after the Kth iteration as the estimate of time step $t-1$, $\hat{\mathbf{h}}_{t-1} = \mathbf{h}_{t-1}^{(K)}$, and if we initialize the optimization for time step t using the prediction from the previous time step's estimate, $\mathbf{h}_t^{(0)} = \mathbf{D}^\mathsf{T}\mathbf{F}\mathbf{D}\hat{\mathbf{h}}_{t-1}$, then iterative optimization is equivalent to SISTA in algorithm 2.

VITA

Scott Wisdom grew up in Colorado Springs, Colorado. Driven by a passion for computers and literature, he earned a B.S. in Electrical and Computer Engineering and a B.A. in English Literature from the University of Colorado Boulder. After working as a software engineer at Fluke Networks in Everett, Washington for two years, he returned to graduate school and earned a M.S. in Electrical Engineering at the University of Washington in Seattle in 2014, with a thesis title of, "Improved Statistical Signal Processing of Nonstationary Random Processes Using Time-Warping." During the course of his PhD, he completed internships at Microsoft Research in Redmond, Washington and Mitsubishi Electric Research Labs in Cambridge, Massachusetts. During the summer of 2015, he was a graduate student member of the Far Field Speech team at the Jelinek Workshop on Speech and Language Technology held at the University of Washington in Seattle. After graduation, he will be joining Affectiva in Boston, Massachusetts as a senior research scientist.