Generative models for neural time series with structured domain priors

by

Andrew Hyungsuk Song

B.S., Massachusetts Institute of Technology (2015)
M.Eng., Massachusetts Institute of Technology (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author . . . .
Department of Electrical Engineering and Computer Science
December 22, 2021

Certified by . . . .
Emery N. Brown
Edward Hood Taplin Professor of Medical Engineering and
Computational Neuroscience
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Generative models for neural time series with structured domain priors

by

Andrew Hyungsuk Song

Submitted to the Department of Electrical Engineering and Computer Science
on December 22, 2021, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

When I initially set out to research in the intersection of statistical signal processing and neuroscience (neural signal processing), my research advisor, Professor Emery N. Brown, explained at length that the signals from seemingly complex neural/biological systems are not purely random, but rather those that have latent structures that can be recovered with *principled* approaches. This insight has stuck with me since that moment and my research throughout graduate school has been understanding and practicing what I thought was the appropriate neural signal processing framework. In this thesis, I define this framework from the Bayesian/optimization perspective and emphasize translating and integrating the clinical and scientific domain knowledge, obtained from constant interaction/collaboration with the experimental neuroscientists and clinicians. The thesis specifically focuses on uncovering latent structures in the neural time series data, by using domain priors/constraints, such as Gaussian process, shift-invariance, sparsity, and smoothness, among many others. It is demonstrated in the thesis that the Bayesian approach with careful integration of these constraints produces results/structures in the data that are not only intepretable but also better performing for the metrics of interest.

Thesis Supervisor: Emery N. Brown
Title: Edward Hood Taplin Professor of Medical Engineering and Computational Neuroscience

Acknowledgments

First of all, I want to express my deepest gratitude to my thesis supervisor professor Emery N. Brown. When I was in the very early stages of my research career, with a vague idea of wanting to research in computational neuroscience, he took a chance in me and has supported me ever since. Although it could have been a simple gesture on his side, it meant a lot to me and my career trajectory has been transformed since then. It has been a true honor and immense learning experience for me to be able to discuss not only neuroscience but also statistics, data science, and education in general. Without Emery's guidance, care, and support, I would not be the person I am now. Next, I want to thank my co-supervisor professor Demba Ba, who was like a big brother to me. When we met for the first time, we were both at the start of our respective careers as a professor and Ph.D. student - it was a memorable journey as we navigated and advanced our careers together. I cherish all the late-night discussions and whiteboard sessions we have had in the Maxwell Dworkin building. I also want to thank other members of my thesis committee, professor Polina Golland and professor Thomas Heldt, for agreeing to mentor/advise me on my research, although my research was not in their fields.

Outside of the committee, there are so many people I would like to thank, reaching back to 2009, when I first came here as an undergraduate student. I want to thank Professor George Verghese, who had been my undergraduate academic advisor and to whom I could just go and talk about my struggles and achievements all the time. I also want to thank Doctor Oluwaseun Akeju, who took me under his wings when I first sought out to do neuroscience research and has since been my great friend and advisor. The same gratitude also goes to Francisco Flores, Sourish Chakravarty, and Anne Smith, who at various times of my graduate studies have mentored me and helped me develop as an independent researcher.

Next, shout out to all the friends that I met at MIT, who truly made my life enjoyable. Without them, I honestly think I would not have made it through the finish line. I want to thank all the members of the Neuroscience Statistics Research Lab

(NSRL) at MIT and Computation, Representation, and Information in Sign Processing group (CRISP) at Harvard, for putting up with me, always being eager to discuss anything, and making fun memories. I also want to thank all of my friends in the Korean community for all the food/drinks/memories we shared over all these years.

Thank you Hanseul Kim, for always being my side for the last four years. You have been my lover, soulmate, comedian (!), and just everything I could have ever wished for. Without your unwavering support and understanding as a fellow Ph.D. student, I would not have made it this far. I am excited about the future that we are going to share together for many many more years.

Finally, I want to express my eternal gratitude to my family for their unconditional support throughout all these years. To my dad Jaebok Song: It was an honor for me to follow your footsteps into the research career - I hope I made you really proud! To my mom Soomi Yoo: Thank you always for believing in me and thinking of me every day - Knowing that you would always be looking out for me has been the biggest comfort. To my brother Brian Changsuk Song: Thank you for being my best friend to whom I could just talk about anything!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With recent technological advances, it has become possible to collect and analyze neural data from diverse modalities to understand the systemic properties of the brain and its dynamics [1, 2, 3]. These include electroencephalogram (EEG) [4], magnetoencephalogram (MEG) [5], functional magnetic resonance imaging (fMRI) [6], 2-photon calcium imaging [7], and neural spikes [8]. From the signal processing perspective, this trove of multimodal data presents an unprecedented opportunity for an engineer to make a contribution to scientific discovery in neuroscience and engineering applications, such as brain-computer interface (BCI) [9] and brain-machine interface (BMI) [10].

In this thesis, we are interested in a principled framework for neural signal processing [11], or computational neuroscience, from the generative (Bayesian) perspective. Specifically, we treat the brain or a sub-system of the brain as a complicated generative model, which generates the neural data observations collected by the recording devices. Given the data, the goal is to solve the *inverse* problem of estimating/inferring certain properties of the neural system,. In other words, we establish the process of 1) designing/formulating a generative model, 2) training/optimizing the objective function yielded by the generative model (or some augmented version of this), and 3) performing posthoc analysis using the quantities derived from the posterior distribution. This is indeed a standard statistical inference procedure across many disciplines.

What now becomes important is how we adapt this standard paradigm to neuroscience, by incorporating the domain knowledge and constraints specific to the neural system and the data modalities. In addition to constant experimentation with different models and parameter settings by the engineers, this effort crucially depends on constant collaboration with the experimental neuroscientists and clinicians, who have first-hand intuitions and domain knowledge. Numerous examples can be found where domain knowledge has shaped the neural signal processing approaches. In spike sorting [8], a template matching approach is popular [2, 12, 13], since we understand that 1) the action potential of each neuron is distinct, 2) the shape of an action potential is maintained throughout the recording session (shift-invariance), and 3) the neurons emit action potentials *sparsely* due to biophysical constraints. In fMRI, the hemodynamic response function for blood-oxygen-level-dependent (BOLD) signal is modeled as a difference of two Gamma distribution functions, based on our understanding of how the blood oxygenation level changes in response to a single neural activity [14].

The main goal of the thesis is to explore the interplay between neural signal processing and the generative paradigm, with a focus on how different neural data modalities and neural systems impose constraints/priors on the whole procedure. We first start with reviewing the Bayesian framework.

1.1 Central theme: The Bayesian perspective

The central theme of the thesis revolves around the Bayesian perspective [15], captured in the ever-present Bayes' rule

$$p(\mathbf{x} \mid \mathbf{y}; \theta) = \frac{p(\mathbf{y} \mid \mathbf{x}; \theta) p(\mathbf{x}; \theta)}{p(\mathbf{y}; \theta)} = \frac{p(\mathbf{y} \mid \mathbf{x}; \theta) p(\mathbf{x}; \theta)}{\int_{\mathbf{x}} p(\mathbf{y} \mid \mathbf{x}; \theta) p(\mathbf{x}; \theta) d\mathbf{x}} \propto p(\mathbf{y} \mid \mathbf{x}; \theta) p(\mathbf{x}; \theta), \quad (1.1)$$

where $\mathbf{y}$ is an observation, $\mathbf{x}$ is a latent random variable, and θ represents a set of parameters for the likelihood and the prior distribution. Although Eq. 1.1 could further incorporate hyperpriors on θ, we restrict ourselves to the simplest form. This simple equation establishes that the *posterior* distribution $p(\mathbf{x} \mid \mathbf{y}; \theta)$ is simply a multiplication of the *likelihood* $p(\mathbf{y} \mid \mathbf{x}; \theta)$ and the *prior* $p(\mathbf{x}; \theta)$, up to some normalization constant $p(\mathbf{y}; \theta)$. Within the Bayesian framework, one needs to address the following three questions, regardless of the application domain.

1. **What is the latent x?**
2. **What should be the likelihood and the prior?**
3. **How do you perform posterior inference?**

We now address each of these questions in connection with neural signal processing.

1.1.1 What is the latent x?

As one would expect, the identity of the latent $\mathbf{x}$ (and consequently its dimension) is highly application-dependent. In neural signal processing, there have been three different levels of abstraction for the latent $\mathbf{x}$, although we note there exist grey areas between these categories.

1. **Unobserved quantity** The latent $\mathbf{x}$ corresponds to an intuitive and physical, yet unobserved quantity that one wants to estimate. Therefore the forward (generative) model is relatively simple, usually a combination of simple linear & nonlinear mappings. For example, $\mathbf{x}$ could represent a firing rate of the population of neurons [16, 17] or oscillatory sub-process of the EEG data [18, 19, 20].

2. **latent state** The latent $\mathbf{x}$ corresponds to an abstract brain state or identity. For example, $\mathbf{x}$ could represent the cognition level of animals [21, 22], sleep stages [23, 24], or different levels of unconsciousness [25, 26, 27]. It could also correspond to the cluster identity for a neuron in a functional clustering of neuron population clustering framework [28]. One needs to carefully specify the forward model mapping the latent states to the observations, for $\mathbf{x}$ to faithfully represent the brain states that one had originally intended.

3. **low-dimensional latent embedding** In this furthest level of abstraction, $\mathbf{x}$ represents an abstract low-dimensional embedding of neural observations, for

which the interpretation could be less trivial. The forward model is highly nonlinear and usually involves neural networks [29], also rendering the inverse problem highly nonlinear - the inference procedure usually relies on variational approximation [30, 31]. This has been an active research area in computational neuroscience, specifically the formulation of appropriate/robust deep generative models for specific neural data modalities and the interpretation/identification of the latent $\mathbf{x}$ [32, 33, 34].

In this thesis, latent $\mathbf{x}$ is defined exclusively as an unbiased quantity, i.e., from the first category. This allows the practitioner to directly interpret the estimated $\widehat{\mathbf{x}}$ and establish direct connection to a scientific/physical quantity of the neural system. For instance, in Chapters 2 and 3, $\widehat{\mathbf{x}}$ corresponds to a vector of estimated Fourier coefficients. In Chapter 4, $\widehat{\mathbf{x}}$ corresponds to a frequency-modulated & bandlimited time series. In Chapters 5, 6 and 7, $\widehat{\mathbf{x}}$ corresponds to the sparse codes that indicate the locations/amplitudes of the dictionary templates.

1.1.2 The formulation of likelihood and the prior

Among the three questions, the formulation of the likelihood and the prior is perhaps the most influenced by the domain constraints. Below are several examples of this relationship in neural signal processing.

- **Data type (likelihood)** If the data is continuous and real-valued, such as electrophysiological recording, it makes sense to use the Gaussian distribution. If the data is count-valued, such as neural spikes or other Poisson-type observations, it makes sense to use binomial, Poisson, or other generalized count distributions [35]. If the data is category-valued, such as animal behaviors or cognitive test scores, it is reasonable to use the multinomial distribution. The likelihood will then be dictated by 1) the choice of data distribution and 2) the choice of the link function that maps $\mathbf{x}$ to $\mathbf{y}$. In this thesis, we use the generalized linear model (GLM) framework [36] to account for many different distributions of the exponential family.

- **Shift-invariance (likelihood)** If a certain set of patterns is repeated in the data, we can incorporate the shift-invariance constraint into the generative model setup, which is a common practice in neural signal processing. In spike sorting, the algorithms assume that the action potential from each neuron is distinct and shift-invariant throughout the recording regime. In calcium imaging [7, 37] and functional MRI [6], we observe calcium fluorescent signal or blood-oxygen-level-dependent (BOLD) signal, respectively, modeled as a convolution between neural activities and modality-dependent shift-invariant kernels.

- **Sparsity (prior)** The idea that a signal can be decomposed into a *sparse* number of elements in some basis set is a powerful idea, which has found its footing in nearly all applications of science and engineering [38]. In neural signal processing, sparsity is used across different spatio-temporal scales. For example, the problem of source-localization [39, 40] for EEG, where the goal is to identify a sparse number of hypothetical sources for signals recorded across the EEG channels, requires sparsity. For neural spiking data, sparsity is often used for spike-sorting and calcium deconvolution problems [37], due to the refractoriness period of the neurons (neuron has to recharge for a certain period before firing again [41]). The commonly used sparsity constraints are ℓ_0 psuedo-norm, which penalizes the number of nonzero elements, ℓ_1 norm, which penalizes the sum of the absolute values of the elements, and mixed-norm.

- **Smoothness (likelihood & prior)** Smoothness is a fundamental concept that has different characterizations in different fields. From the signal processing perspective, the signal is bandlimited and does not (or minimally) contain a high-frequency content. From the statistical perspective, the first/second-order derivatives of the signal are bounded [42]. Intuitively, these different characterizations commonly describe that if a signal is smooth, the adjacent data points are not too different from each other.

  In neural signal processing, we use smoothness in two different manners. First, we can reasonably assume that the finite-length activation patterns/kernels in response to certain stimulus/activities are smooth. For example, smoothness is typically assumed, either explicitly or implicity, for the analysis frameworks for the action potential from a neuron [43], BOLD signal in fMRI [14], calcium transient in 2-photon calcium imaging [37], and point spread functions in optical imaging [44]. These kernels are either assumed to be linear combinations of smooth basis functions (explicit) or learned with a smooth regularizer function (implicit).

We can also impose smoothness on the latent process. Without external intervention and stimulus, the neural dynamics of the idle/resting-state brain are usually assumed to be smooth. The state-space priors [16, 45] or Gaussian process priors [46] have been popular choices for this. Examples include neuron firing rate [16, 17, 47], and the cognitive learning process [21, 22].

1.1.3 Posterior inference

The term broadly refers to the two related tasks: 1) estimating the parameters $\widehat{\theta}$ of the generative model by maximizing some criteria, such as the likelihood, and 2) obtaining a quantity of interest derived from the posterior distribution $\mathbb{E}_{p(\mathbf{x}|\mathbf{y};\widehat{\theta})}[f(\mathbf{x})]$, where f is a pre-defined function. Since these tasks are tightly coupled, we collectively refer to them as the posterior inference in this thesis.

The main challenge of the posterior inference is evaluating $\int_{\mathbf{x}} p(\mathbf{y} \mid \mathbf{x};\theta)p(\mathbf{x};\theta)d\mathbf{x}$ in the denominator of Eq. 1.1. This is due to the fact that for a given generative model setup, the prior is usually not a conjugate prior of the likelihood, which makes the integral in Eq. 1.1 intractable. In neural signal processing, this has usually been dealt in three different approaches: 1) exact posterior inference through Monte-Carlo sampling (with possibly data augmentation) [28, 48, 49], 2) approximate variational posterior inference [32, 33, 50], and 3) maximum-a-posteriori point estimation [37, 45, 51].

We use the last option throughout the thesis, and maximize the log-posterior objective $\log p(\mathbf{x} \mid \mathbf{y}; \theta)$

$$\max_{\mathbf{x},\theta} \log p(\mathbf{x} \mid \mathbf{y}; \theta) = \max_{\mathbf{x},\theta} \log p(\mathbf{y} \mid \mathbf{x}; \theta) + \log p(\mathbf{x}; \theta) + C, \tag{1.2}$$

where C represents the terms that are constant with respect to $\mathbf{x}$ and θ. This approach essentially casts the problem of Bayesian inference as an optimization problem, for which numerous off-the-shelf convex/non-convex optimization algorithms are available. Although we do not have access to the full posterior distribution contrary to the first/second approaches (we only have access to $\widehat{\mathbf{x}}_{\text{MAP}} = \arg\max \log p(\mathbf{x} \mid \mathbf{y}; \widehat{\theta})$), we demonstrate throughout the thesis that for the purposes of interpretation and downstream tasks, this is often good enough.

For the optimization of θ, we follow either of the two approaches. First, we can perform alternating minimization (or block coordinate descent) on $\mathbf{x}$ and θ until convergence. Second, if a validation dataset is available and θ is low-dimensional (i.e. hyperparameters of dimension 1 or 2), we can perform cross-validation. We predominantly use the first approach throughout the thesis, although we use cross-validation in Chapter 4 and Chapter 6.

We can further improve the efficiency of the optimization algorithm, depending on the choice of the likelihood and the prior, making it more scalable for a big-data regime. For instance, the state-space prior used in Chapters 2, 3, and 4 allows us to solve the optimization problem with several iterations of Kalman filtering/smoothing, which are more efficient than its convex optimization algorithm alternatives [18]. In Chapter 6, leveraging the exponential family likelihood and the quadratic penalty function induced by the prior, we can use the computationally efficient iteratively reweighted least squares (IRLS) algorithm to solve the optimization problem. In Chapter 7, we use the constrained auto-encoder neural network architecture to solve the problem more efficiently on GPU, leveraging the iterative nature of the proximal algorithms used for solving optimization problems with ℓ_1 sparsity constraints [52, 53].

1.2 Neural signal processing for time series with structured prior

Now that we have developed the Bayesian framework for neural signal processing, we examine two different approaches for modeling latent $\mathbf{x}$ in the context of time series.

1. **$\mathbf{x}$ as the latent process**: We treat $\mathbf{x}$ as the dynamical latent process that evolves at some timescale of interest (e.g., certain intervals, sampling timestamps). In this context, $\mathbf{x} \in \mathbb{R}^{d\times T}$, where $\mathbf{x}_t \in \mathbb{R}^d$ represents the d-dimensional latent process at time t. It will generally take form of the state-space model

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, Q), \\ p(\mathbf{y}_t \mid \mathbf{x}_t) &= f(\mathbf{x}_t) \end{aligned} \tag{1.3}$$

   where the state $\mathbf{x}_t$, the transition $\mathbf{A}$, the state covariance Q, and the link function f all depend on the specifics of the application. This is equivalent to having the following Markov process prior

$$p(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}) \quad \text{or} \quad \log p(\mathbf{x}_{1:T}) = \sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}). \tag{1.4}$$

2. **$\mathbf{x}$ as the latent codes**: We treat $\mathbf{x}$ as sparse latent codes that indicate the locations and amplitudes of temporal patterns with finite supports. We denote $\{\mathbf{h}_c\}_{c=1}^{C}$, where $\mathbf{h}_c \in \mathbb{R}^L \ \forall c$ and $L \ll T$, as the set of these temporal patterns and $\mathbf{x}^c \in \mathbb{R}^{T-L+1}$, as the *code vector* which has non-zero elements only for the locations of $\mathbf{h}_c$. We can then formulate

$$\begin{aligned} \|\mathbf{x}^c\|_p &\leq \beta_c, \forall c \quad \text{for } 0 \leq p \leq 1 \\ \mathbf{r}_t &= \sum_{c=1}^{C} (\mathbf{h}_c * \mathbf{x}^c)_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, Q) \\ p(\mathbf{y}_t \mid \{\mathbf{x}_t^c\}_{c=1}^{C}) &= f(\mathbf{r}_t), \end{aligned} \tag{1.5}$$

   with $\|\mathbf{x}^c\|_p \leq \beta_c$ representing a sparsity constraint. This is equivalent to having the following sparsifying prior

$$p(\mathbf{x}_{1:T}) \propto \prod_{t=1}^{T} \exp(-\lambda |\mathbf{x}_t|^p) \quad \text{or} \quad \log p(\mathbf{x}_{1:T}) = -\lambda \sum_{t=1}^{T} |\mathbf{x}_t|^p + C. \tag{1.6}$$

For $p = 1$, we have the Laplace prior, which leads to the popular LASSO regularization [38]. We note that these cover only a small subset of interesting time series priors, which includes sparse prior [18, 54] and nonlinear state-space prior [55].

1.3 Thesis organization

The thesis covers several neural signal processing pipelines with a focus on the neural time series data. This thesis is comprised of three major parts.

- **Time-frequency analysis of neural time series (Chapter 2, 3, 4)** We study several generative models for continuous neural time series, specifically EEG and electrophysiological recordings, with the main focus on modeling the evolving spectral dynamics in the time-frequency domain. We use Gaussian distribution for the likelihood and focus on random-walk or GP prior within the state-space framework.

  These chapters are based on the following works

  [56] **Andrew H. Song***, Sourish Chakravarty*, and Emery N. Brown, *A Smoother State Space Multitaper Spectrogram*, IEEE EMBC, 2018

  [24] **Andrew H. Song** et al., *Multitaper Infinite Hidden Markov Model for EEG*, IEEE EMBC, 2019

  [20] **Andrew H. Song**, Demba Ba, and Emery N. Brown, *PLSO: A generative framework for decomposing nonstationary timeseries into piecewise stationary oscillatory components*, Uncertainty in Artificial Intelligence (UAI), 2021

  [.] **Andrew H. Song***, Seong-eun Kim*, and Emery N. Brown, *Adaptive State-Space Multitaper Spectral Estimation*, Submitted to IEEE Signal Processing Letters, 2021

- **Smooth convolutional dictionary learning (Chapter 5, 6)** Convolutional dictionary learning (CDL), the problem of estimating shift-invariant templates from data, gained popularity with the seminal work of [57] in computational neuroscience. In addition to the shift-invariance and sparsity constraints, we use a smoothness constraint to further constrain the generative model. Chapter 5 and 6 study different CDL generative models for the same goal of learning smooth and interpretable dictionaries. We apply these frameworks on diverse biological data, such as spike sorting and super-resolution microscopy application, to demonstrate how smoothness constraints contribute to improved performance.

  These chapters are based on the following works
  [13] **Andrew H. Song**, Francisco Flores, and Emery N. Brown, *Convolutional Dictionary Learning with Grid Refinement*, IEEE Transactions on Signal Processing, 2020
  [58] **Andrew H. Song**, Bahareh Tolooshams, and Demba Ba, *Gaussian Process Convolutional Dictionary Learning*, IEEE Signal Processing Letters, 2021

- **Constrained neural networks for efficient inference (Chapter 7)** Whereas the previous two parts focused on how to set up a generative model and incorporate relevant domain constraints, we focus specifically on improving the optimization/inference procedure for a given generative model. Specifically, we introduce a highly constrained neural network autoencoder as an efficient inference and learning framework for CDL generative model. We show that this architecture can solve the CDL optimization orders of magnitudes faster than off-the-shelf optimization algorithms, a crucial improvement in analyzing neural data with explosive growth in size [2].

  These chapters are based on the following works
  [59] Bahareh Tolooshams*, **Andrew H. Song***, Simona Temereanca, and Demba Ba, *Convolutional dictionary learning based auto-encoders for natural exponential-family distributions*, International Conference on Machine Learning (ICML), 2020

[60] Alexander Lin, **Andrew H. Song**, and Demba Ba, *Mixture Model Auto-Encoders: Deep Clustering through Dictionary Learning*, Submitted to IEEE ICASSP, 2021

Each chapter contains the thesis-adapted version of the manuscripts. In between the chapters, we also provide a brief insight on how each chapter relates to other chapters or the central topic.

Chapter 2

Time-frequency analysis with second-order state-space model

Primer This chapter is adapted from the following work
- Andrew H. Song*, Sourish Chakravarty*, and Emery N. Brown, *A Smoother State Space Multitaper Spectrogram*, IEEE EMBC, 2018

Frequency domain analysis of time series data with cyclic signature is an informative approach extensively used across multiple disciplines encompassing science, engineering and medicine. Due to the noiseness of the time series in general, it is often modeled as a realization of a stochastic process. Consequently, accurate *estimation* of the unknown true spectrum from sequential observations becomes vital. When the time series data can be assumed to be realizations of a second order stationary (s.o.s.) stochastic process, one can define the Power Spectral Density (PSD) as the Fourier transform of the autocovariance of the s.o.s. process [61]. In practice, estimation based on the Fourier transform on finite data sequences results in a tradeoff between bias and variance of the PSD estimate [62]. In this regard, the multitaper spectral (MT) analysis method [63] provides PSD estimators with optimal bias-variance tradeoffs.

In practical applications, a stochastic process may demonstrate non-stationarity, such as the electroencephalogram (EEG) recording from a patient undergoing anesthesia [62]. Nevertheless, for short time windows, such as a few seconds as in the anesthesia example, it is reasonable to assume s.o.s., allowing the MT to be applied to each window. To derive a dynamic description of the spectrum (*spectrogram*) it is a common practice to calculate the spectra for consecutive locally s.o.s. data segments which may or may not be overlapping. The choice of the overlap duration is somewhat arbitrary and, moreover, this approach does not allow for statistical inference on the entire time series. The State-Space Multitaper (SSMT) spectral analysis framework was recently introduced to address this gap [19]. The SSMT method is based on a state-space (SS) model formulation comprising: (1) the observations which are Fourier transforms of the multiple tapered versions of each locally s.o.s. segment, and (2) multiple mutually-independent 1-dimensional discrete time states (one for each Fourier coefficient) following a first-order random walk, thus imposing stochastic continuity across consecutive s.o.s. segments.

In this work, we assume that each state underlying the observations are discretely sampled version of an Integrated Wiener Process (IWP). Specifically, we choose a simple model where the time derivative of each state follows a Wiener Process. The imposition of a stochastic continuity on the first-order time derivatives in this framework (referred henceforth as IWP-SSMT) leads to smooth estimates. We augment each state from SSMT with its time-derivative and adapt the parameters in the SS framework accordingly. The rest of the paper is organized as follows: Section 2.1 provides an overview of the SSMT framework. Section 2.2 introduces the IWP-SSMT method. In sections 2.3 and 2.4 , the results and conclusions are presented, respectively.

2.1 Overview of the SSMT paradigm

Consider a sequence of real-valued scalar data $\{y_1, y_2 \cdots , y_T\}$, sampled at a rate F_s (in Hz), from a non-stationary time series $\{x_1, x_2 \cdots , x_T\}$ such that, $y_t = x_t + \varepsilon_t$ where, ε_t is a Gaussian white noise process with zero-mean and constant variance σ^2_ε and x_t is assumed to be a zero-mean, locally s.o.s. Gaussian process. We assume that J observations within each non-overlapping window are s.o.s., but not necessarily across K windows ($T = KJ$). A vectorized representation of the time sequences is,

$$Y_k = X_k + E_k, \tag{2.1}$$

where the j-th member of the column vectors are: $Y_{k,j} = y_{J(k-1)+j}$, $X_{k,j} = x_{J(k-1)+j}$ and $E_{k,j} = \varepsilon_{J(k-1)+j}$ for $k = 1, 2 \cdots, K$ and $j = 1, 2, \cdots, J$. Furthermore, X_k can be decomposed into orthogonal sequences in the frequency domain by virtue of the Spectral Representation Theorem [61], $X_k = W\Delta Z_k$,where, $W \in \mathbb{C}^{J\times J}$ and $W_{lj} = J^{-1/2}\exp(i(l-1)j(2\pi/J))$, $\omega_j = 2\pi(j-1)/J$ and $\Delta Z_{k,j} \equiv \Delta Z_k(\omega_j)$ is a sequence of complex-valued orthogonal Gaussian increments $\Delta Z_k = [\Delta Z_k(w_1), \cdots, \Delta Z_k(w_J)]^T$. By taking a discrete Fourier transform (DFT) of Eq. (2.1),

$$Y_k^{(F)} = \Delta Z_k + E_k^{(F)}, \tag{2.2}$$

where, $Y_k^{(F)} = W^{-1}Y_k$ and $E_k^{(F)} = W^{-1}E_k$. Using the principles of MT algorithm applied to data vector Y_k, one can write the following equations

$$Y_k^{(m,F)} = \Delta Z_k^{(m)} + E_k^{(m,F)} \tag{2.3}$$

where, $Y_k^{(m,F)}$ is the DFT of Y_k tapered with the m-th Slepian taper, $m = \{1, 2, \cdots, M\}$. The MT estimate is given by,

$$f_k^{(MT)}(\omega_j) = \frac{1}{M}\sum_{m=1}^{M} \|Y_k^{(m,F)}(\omega_j)\|^2 \tag{2.4}$$

The MT estimator in Eq. (2.4) has an optimal narrow-band vs. broad-band bias tradeoff and has lower variance than the single tapered estimates ([62, 63]). Note that $E_k^{(m,F)}$ in Eq. (2.3) is a complex-valued Gaussian random vector with a covariance matrix $\Sigma_E^{(m,F)} = (\sigma_\varepsilon^{(m)})^2 I_J$, where I_J denotes an $J \times J$ identity matrix and $\sigma_\varepsilon^2 = \sum_{m=1}^{M}(\sigma_\varepsilon^{(m)})^2/M$. The SSMT assumes that the increments $\Delta Z_k^{(m)}$ follows a first-order random walk per, $\Delta Z_k^{(m)} = \Delta Z_{k-1}^{(m)} + V_k^{(m)}$ where, $V_k^{(m)}$ is a complex valued Gaussian random vector with a diagonal covariance matrix $\Sigma_V^{(m)}$, with the (j, j)-th component as $\sigma_{v,j}^{(m)}$. This setup allows for a SS description independently for each frequecy bin and for each taper per,

$$\Delta Z_{k,j}^{(m)} = \Delta Z_{k-1,j}^{(m)} + v_{k,j}^{(m)} \tag{2.5}$$

$$Y_{k,j}^{(m,F)} = \Delta Z_{k,j}^{(m)} + \varepsilon_{k,j}^{(m,F)} \tag{2.6}$$

where, $v_{k,j}^{(m)} \sim CN(0, (\sigma_{v,j}^{(m)})^2)$ and $\varepsilon_{k,j}^{(m,F)} \sim CN(0, (\sigma_{\varepsilon}^{(m)})^2)$ [1]. SS formulation allows the user to estimate the states $\Delta Z_{k,j}^{(m)}$ recursively from sequential observations $Y_{k,j}^{(m,F)}$ using MJ independent Kalman filters. The SSMT spectral estimate from the k-th time window is,

$$f_{k|k}^{(SSMT)}(\omega_j) = \sum_{m=1}^{M} \|\Delta Z_{k|k}^{(m)}(w_j)\|^2/M \tag{2.7}$$

where, $x_{i|j}$ refers to the estimate of the process at the i-th window, based on the observations up to the j-th window.

2.2 SSMT with $\Delta Z(t)$ modeled as an IWP

2.2.1 Motivation

As an extension to SSMT, we model ΔZ_k as a discrete process sampled regularly from a continuous process, $\Delta Z(t)$, where $\Delta Z(t)$ is assumed to follow an IWP. This is a different modeling assumption from SSMT, where ΔZ_k was assumed to obey a first order random walk (a discretized representation of the Wiener Process). The main motivation for IWP comes from a connection between the IWP and the smoothing spline [64] - if a continuous process is an IWP, then the maximum aposteriori estimate of the process conditioned on observations (perturbed by Gaussian noise) is asymptotically equivalent to discrete samples from a smoothing spline fitted to the same observations.

[1] $x \sim CN(\mu_x, \sigma_x^2)$ refers to a circularly symmetric complex normal distribution. $x \in \mathbb{C}$

2.2.2 State-space formulation

We assume that the underlying continuous process for $Re\{\Delta Z^{(m)}_{k,j}\}$ and $Im\{\Delta Z^{(m)}_{k,j}\}$ each follow simple IWP *independently.* It is a reasonable assumption as the real and imaginary DFT coefficients are *asymptotically* independent [61, Appendix C]. By simple IWP, we mean $d(\Delta Z) = W(t)dt$, where $W(t)$ is a Wiener process. The discrete process ΔZ_k, sampled from $\Delta Z(t)$ following a simple IWP, can be formulated into SS framework via Laplace transform [65]. In the ensuing discussion, we will use the following matrices, where $\tau = J/F_s$ is a stationary interval length,

$$\Phi = \begin{pmatrix} 1 & \tau \\ 0 & 1 \end{pmatrix}, \quad Q = \begin{pmatrix} \tau^3/3 & \tau^2/2 \\ \tau^2/2 & \tau \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 \end{pmatrix}. \tag{2.8}$$

The state and observation equations, respectively, are

$$\begin{pmatrix} \Delta Z^{(m)}_{k,j} \\ \Delta \dot{Z}^{(m)}_{k,j} \end{pmatrix} = \Phi \begin{pmatrix} \Delta Z^{(m)}_{k-1,j} \\ \Delta \dot{Z}^{(m)}_{k-1,j} \end{pmatrix} + v^{(m)}_{k,j}, \text{ where,}$$

$$v^{(m)}_{k,j} \sim CN\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, (\sigma^{(m)}_{v,j})^2 Q\right), \text{and} \tag{2.9}$$

$$Y^{(m,F)}_{k,j} = A \begin{pmatrix} \Delta Z^{(m)}_{k,j} \\ \Delta \dot{Z}^{(m)}_{k,j} \end{pmatrix} + \varepsilon^{(m,F)}_{k,j} \text{ where,}$$

$$\varepsilon^{(m,F)}_{k} \sim CN(0, (\sigma^{(m)}_{\varepsilon})^2). \tag{2.10}$$

Similar to Eq. (2.7), we define the IWP-SSMT spectral estimate as,

$$f^{(IWP\text{-}SSMT)}_{k|k}(\omega_j) = \sum_{m=1}^{M} \|\Delta Z^{(m)}_{k|k}(w_j)\|^2 / M \tag{2.11}$$

where, $\Delta Z^{(m)}_{k|k}(w_j)$ is estimated using a Kalman filter based on the SS model per Eqs. (2.9) and (2.10). For notational convenience, we define a new random state vector $g^{(m)}_{k,j} = [\Delta Z^{(m)}_{k,j}, \Delta \dot{Z}^{(m)}_{k,j}]^T$. We indicate the conditional mean by $\widehat{g}_{k,j|l} = E[g_{k,j}|Y^{(F)}_{1,j}, \cdots, Y^{(F)}_{l,j}]$ and its associated error covariance by $\Gamma_{k,j|l} = E[(g_{k,j} - \widehat{g}_{k,j|l})(g_{k,j} - \widehat{g}_{k,j|l})^H|Y^{(F)}_{1,j}, \cdots, Y^{(F)}_{l,j}]$ where $(\cdot)^H$ indicates complex conjugate transpose. The filtered

($\widehat{g}_{k,\,j|k}^{(m)}$ and $\Gamma_{k,j|k}^{(m)}$) and smoothed ($\widehat{g}_{k,\,j|K}^{(m)}$ and $\Gamma_{k,j|K}^{(m)}$) estimates can be calculated using standard Kalman filtering and smoothing equations [61]. Here, the Kalman Gain (KG), $C_{k,j}^{(m)}$, is

$$C_{k,j}^{(m)} = \Gamma_{k,j|k-1}^{(m)} A^T \left(A\Gamma_{k,j|k-1}^{(m)} A^T + (\sigma_{\varepsilon}^{(m)})^2 \right)^{-1} \tag{2.12}$$

We define a *roughness* metric, $R_{j,m} = \sum_{k=2}^{K} r_k/(K-1)$, where $r_k \equiv E\left[(x_k - x_{k-1})^2 | Y_{1:K,j}^F\right]$

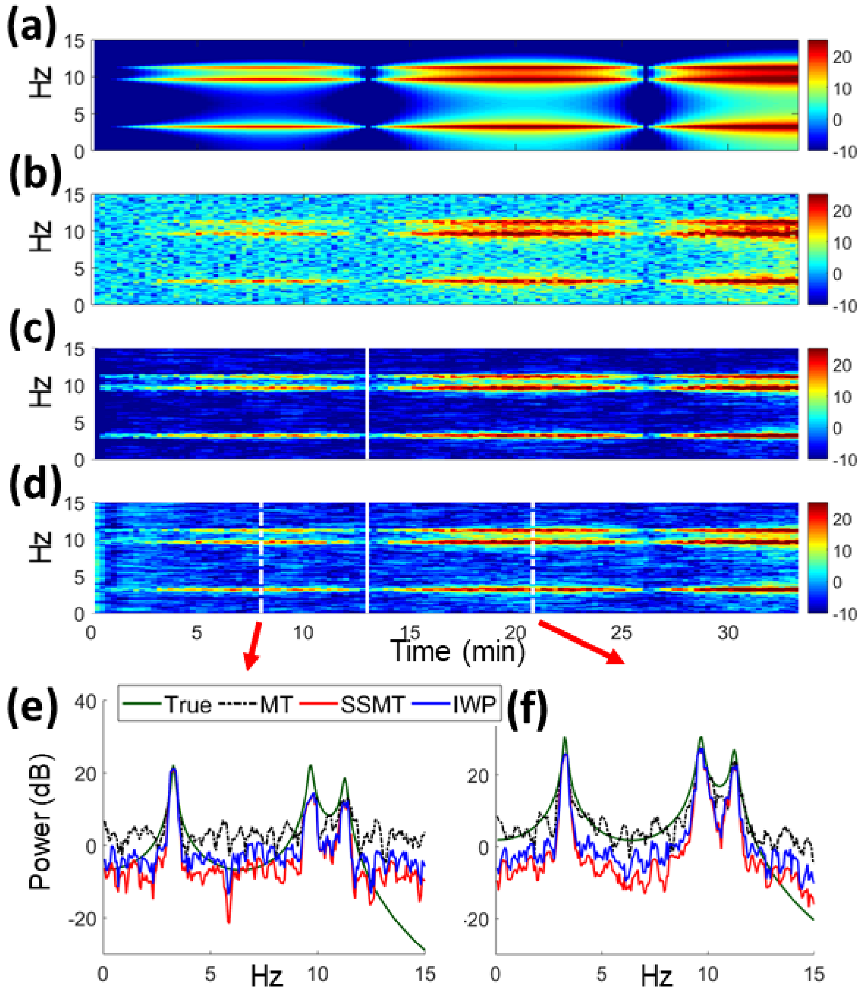

Figure 2-1: Spectrograms (dB scale) from synthetic data. (a) True spectrogram (dB) for an AR process $x_t = 3.9515x_{t-1} - 7.885x_{t-2} + 9.7340x_{t-3} - 7.7435x_{t-4} + 3.8078x_{t-5} - 0.9472x_{t-6} + \varepsilon_t \sim N(0, ((k/K)\sin(k/K))^2)$ for $t \in ((k-1)J, kJ]$ and $y_t = x_t + \varepsilon_o \sim N(0, 10^2)$, and spectrograms estimated by (b) MT (c) SSMT (d) IWP-SSMT methods. The calibration data for both SSMT and IWP-SSMT is taken from $t = 0$ to the time indicated by the solid white vertical line. (e, f) Comparing spectra at two time points. $F_s = 64Hz$, $J = 1024$, $K = 125$, $w_r = 0.25Hz$, $M = 3$, and calibration length = 13 min.

and x_k can denote either $Re(\Delta Z_{k,j}^{(m)})$ or $Im(\Delta Z_{k,j}^{(m)})$. Roughness metric quantifies the expected value of differences between the consecutive states conditioned on the entire data. The lower the metric, the smoother the spectral estimates across time.

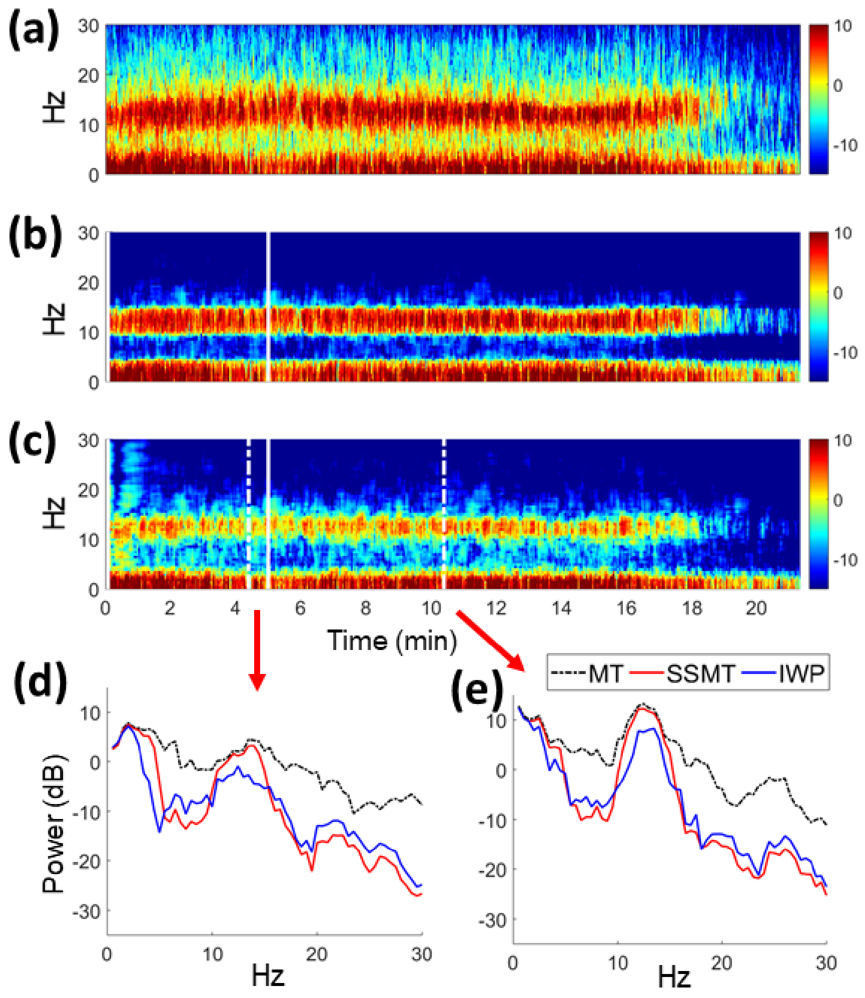

Figure 2-2: Spectrograms (dB scale) from EEG data based on (a) MT, (b) SSMT, (c) IWP-SSMT methods. The calibration data for both SSMT and IWP-SSMT is taken from $t = 0$ to the time indicated by the solid white vertical line, $t = 5$ min. (d) Comparing spectra at two time points. Parameters: $F_s = 250Hz$, $J = 500$, $N = 670$, $w_r = 2Hz$, $K = 4$.

2.2.3 Parameter estimation

To estimate the parameters in the IWP-SSMT framework we apply an Expectation Maximization (EM) algorithm [66] on a short calibration dataset. In the E-step, we compute the expectation of the complete data log-likelihood based on the parameter estimates from the previous EM step and on the filtered/smoothed state estimates and error covariances. In the M-step, the expectation is maximized with respect to the parameters of interest (Eqs. (2.13) and (2.14)). These steps are iterated until some convergence crtieria is reached.

$$\widehat{\sigma}^2_\varepsilon = \frac{1}{JK}\sum_{j,k=1}^{J,K}\Big(\|Y^F_{k,j}\|^2 - 2Re\Big\{\Big(Y^F_{k,j}\Big)^*\Delta Z_{k,j|K}\Big\}$$
$$+E\Big[\|\Delta Z_{k,j|K}\|^2|Y^F_{1:K,j}\Big]\Big) \tag{2.13}$$
$$\widehat{\sigma}^2_{v,j} = tr\Big\{Q^{-1}\Big(S_{11} - \Phi S_{10}^H - S_{10}\Phi^T + \Phi S_{00}\Phi^T\Big)\Big\}/2K \tag{2.14}$$

where, $S_{11} = \sum_{k=1}^{K}\widehat{g}_{k,j|K}\widehat{g}^H_{k,j|K} + \Gamma_{k,j|K}$, $S_{10} = \sum_{k=1}^{K}\widehat{g}_{k,j|K}\widehat{g}^H_{k-1,j|K} + \Gamma_{(k,k-1),j|K}$, and $S_{00} = \sum_{k=1}^{K}\widehat{g}_{k-1,j|K}\widehat{g}^H_{k-1,j|K} + \Gamma_{k-1,j|K}$

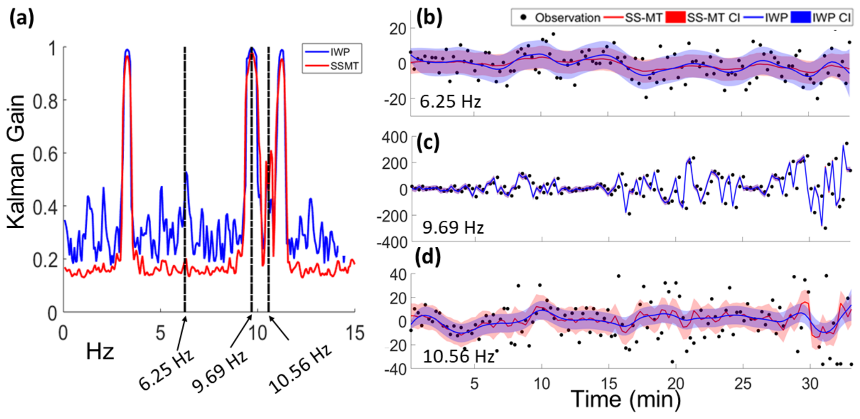

Figure 2-3: Comparing SSMT and IWP-SSMT from synthetic data. (a) Kalman gain vs. frequency for the first taper at time point $t = 12$ min (for IWP-SSMT we plot first component of the KG matrix). (b-d) $Re(\Delta Z^{(1)}_{k|K})$ with 95% confidence interval for 3 frequencies.

2.3 Results

We compare spectrograms among MT, SSMT [19] and IWP-SSMT, applied to two cases. (1) A synthetic data (an autoregressive (AR) process) whose true and MT spectrograms are presented in Fig. 2-1(a, b). (2) A brief snippet of EEG recording obtained via a frontal scalp electrode channel sampled at 250 Hz using a Sedline monitor (Masimo Corp.) during general anesthesia induced by sevoflurane (MT spectrogram is presented in Fig. 2-2(a)). The EEG recording is part of de-identified data collected from patients at Massachusetts General Hospital (MGH) as a part of a MGH Human Research Committee-approved protocol.

Figures 2-1(c, d) and 2-2(b, c), represent SSMT and IWP-SSMT filtered estimates from noisy observations. With the ground truth spectra available for the AR example, Fig. 2-1(a), we infer that both SSMT (Fig. 2-1(c)) and IWP-SSMT (Fig. 2-1(d)) are able to estimate the dynamics of the true spectrogram within the peak power frequencies. Furthermore, we perform a simple quantitative comparison as such: we consider the frequencies with PSD $> 0\,dB$, in Figs. 2-1(e) and (f), which account for $> 97\%$ of the power. For these frequencies, we find that the relative error [2] in the PSDs (MT, SSMT, IWP-SSMT) relative to the true PSD are given by (64%, 65%, 64%) and (68%, 70%, 70%) for Figs. 2-1(e) and (f), respectively. From these point estimates, we infer that the spectral estimates in Fig. 2-1(e, f), indicate that both SSMT and IWP-SSMT approximate the spectrum with same level of accuracy as MT.

Relative to MT, the SSMT and IWP-SSMT spectrograms appear to have higher contrast (Figs. 2-1(b - d) and 2-2(a - c)). This contrast-enhancing effect of SSMT and IWP-SSMT is a consequence of the KG (Fig. (2-3)(a) and Fig. 2-4(a)). KG reflects the degree to which a given SS model believes that there is a smooth underlying process at specific frequencies. For a given model, if KG is close to 1 then it trusts the observations more than the underlying process and hence assigns more weight on the observations ($Y_k^{(F)}$). On the other hand, lower KG indicates that the model trusts the underlying process more. MT approach is blind to the underlying process and simply follows the observations (equivalent to KG=1 for SS models) across all frequencies.

For a given model, SSMT or IWP-SSMT, if KG is nearly 1 (estimates dominated by observations) then one can expect the same degree of smoothness (average separation between consecutive estimates) as the observations, $Y_k^{(F)}$, themselves as seen in (Fig. (2-3)(c) and Fig 2-4(b)). If KG is low (model trusts process more), the

[2] e(x) = $100\|x - x^\star\|/\|x\|$ where $x^\star$ indicates reference truth.

Table 2.1: Roughness metric for the first taper.

	Simulation (Fig. 2-1)		EEG (Fig. 2-2)		
freq.	SSMT	IWP	freq.	SSMT	IWP
6.25	4.52	3.10	1.5	4110	3640
9.69	13330	13000	8.5	5.21	4.38
10.56	46.05	4.42	12.5	1050	8.05

estimates will reflect the smoothness property of the underlying process (both models in Fig. (2-3)(b, d) and Fig 2-4(c) and only IWP-SSMT in Fig 2-4(d)). Thus low KG seems to be indicative of a smoother estimate from a given model.

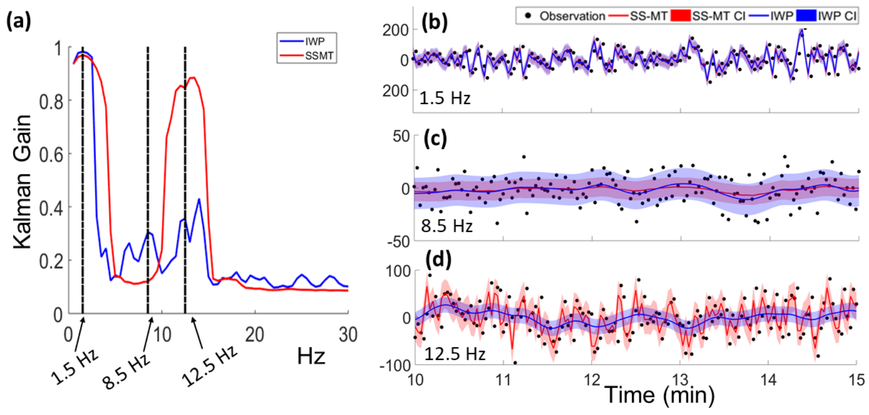

Figure 2-4: Comparing SSMT and IWP-SSMT from EEG data. (a) Kalman gain vs. frequency for the first taper at time point $t = 10$ min. (b-d) $Re(\Delta Z^{(1)}_{k|K})$ with 95% confidence interval for 3 frequencies.

Nevertheless, one needs to be cautious when using just KG to compare smoothness at specific frequencies across different models. For instance, even if KG is nearly identical for values lower than 1 the degree of smoothness could be different as seen in Fig. 2-3(d). Moreover, even if KG is quite different, the degree of smoothness could be similar as seen in Figs. 2-3(b) and 2-4(c). Therefore, to compare smoothness across models, KG by itself is inadequate which necessitates the roughness metric $R_{j,m}$ defined earlier. This roughness metric is appropriate for smoothness comparison across models as it is agnostic to the model choice (SSMT vs. IWP-SSMT). The inference drawn from smoothness comparison between the two models based on visual inspection is corroborated by the roughness metric reported in Table 2.1.

2.4 Conclusion

In this work, we have explored a variant of the SSMT [19] with a more complex description of the process dynamics. In this variant, the Fourier increments in each local s.o.s. time segment are considered discrete samples of the IWP. Both models are able to provide spectrogram estimates that are denoised compared to the MT method for both AR and EEG data analyzed here. Through this work, we have shown that IWP-SSMT can generate smoother estimates of the Fourier coefficients. Since anesthesiologists use EEG to gauge the level of sedation [67], IWP-SSMT and SSMT can potentially be used to track the gradual changes in EEG-based biomarkers (due to gradual changes in amount of drug in the system producing this response) and can thus aid EEG-based drug-effect modeling studies, e.g., [68]. More work is required to analyze the full consequences of the IWP-assumption in the IWP-SSMT. We believe IWP-SSMT, SSMT [19] and variants of this general SS framework can coexist within a prospective model selection toolkit for analysing non-stationary time-series.

Chapter 3

Time-frequency analysis with time-varying state-space model

Primer This chapter is adapted from the following work
- Andrew H. Song*, Seong-eun Kim*, and Emery N. Brown, *Adaptive State-Space Multitaper Spectral Estimation*, Submitted to IEEE Signal Processing Letters, 2021

3.1 Introduction

Nonstationary time series with time-varying probability structures are ubiquitous. Some examples include radar emissions [69], speech and image recordings [70, 71], oceanographic and seismic signals [72], neural spike trains [17], and electroencephalogram (EEG) [73]. We are interested in analyzing the nonstationary data through the lens of the time-varying spectral dynamics, which yields valuable information on the underlying system. The traditional approach has been to segment the data into *independent* overlapping or non-overlapping windows, assuming local stationarity [74] within each window, and to apply Fourier or wavelet transform for spectral analysis [75, 76].

Despite the popularity, the windowing approach suffers from spectral estimates of high variance within each window, due to finite window-length [77] and the restrictive independence assumption for different windows. The state-space multitaper (SSMT) framework [78] and its extensions [79, 80, 56, 20] have been proposed as the solutions, by positing a *time-invariant* latent state-space model in the time-frequency domain, with each state representing the Fourier coefficients in each window. Since the states are linked by stochastic continuity prior across the windows, the spectral estimates are not independent.

However, the use of *time-invariant* parameters limits the capacity of these frameworks to track strong nonstationarity, characterized by strong power fluctuations due to state transitions of the system, which are quite common in systems neuroscience [81]. Although the time-varying state-space model provides more flexibility, this comes at a price, especially for real-time applications. Model parameters for non-stationary process need to be re-estimated with every incoming batch of observations with expectation-maximization (EM) algorithm [82], which requires a prohibitive computational cost.

We propose the time-varying extension of SSMT with an adaptive parameter estimation scheme, termed adaptive SSMT (ASSMT). Based on the data-driven metric to track nonstationarity, theoretically motivated from the generative model, ASSMT adaptively switches between the time-variant and time-invariant state-space model. The adaptive estimation scheme only requires a single pass through the data, making ASSMT more suited for real-time application, compared to other frameworks that require multiple passes through the data.

The rest of the paper is organized as follows. In Section II, the SSMT method is reviewed. In Section III, the adaptive model parameter estimation is proposed. In Sections IV and V, experimental results and conclusion are presented.

3.2 Review of State-Space Multitaper Method

We first review the SSMT algorithm in [78]. Consider a nonstationary time series y_t sampled at frequency f_s as

$$y_t = x_t + \widetilde{\varepsilon}_t, \quad t = 1, \ldots, T \tag{3.1}$$

where x_t is a locally stationary latent Gaussian process [83] with the measurement noise $\widetilde{\varepsilon}_t$ follows $\widetilde{\varepsilon}_t \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Leveraging the local stationary property, we divide these signals into K nonoverlapping stationary intervals of J samples, such that $T = KJ$. The segmented vectors for interval k are denoted as $X_k, Y_k, \varepsilon_k \in \mathbb{R}^J$, with the jth element as $Y_{k,j} = y_{J(k-1)+j}$ for $k = 1, ..., K$ and $j = 1, ..., J$.

To perform time-frequency analysis, we introduce the latent $Z_k = (Z_{k,1}, ..., Z_{k,J}) \in \mathbb{C}^J$, where $Z_{k,j}$ is a complex Gaussian variable with the magnitude corresponding to the power at the normalized frequency $\omega_j = 2\pi(j-1)/J$ and interval k [84]. To model the evolution of spectra across the windows, we assume that $\{Z_k\}_{k=1}^K$ follow a random walk prior

$$Z_k = Z_{k-1} + v_k, \tag{3.2}$$

where v_k is a complex Gaussian noise with a diagonal covariance $I(\sigma_{v,j}^2)$, i.e., $v_k \sim \mathcal{CN}(0, I(\sigma_{v,j}^2))$. This prior encodes two important properties of the latent process. First, the state variance $\sigma_{v,j}^2$ controls the smoothness of the process, with large value indicative of non-smooth or fluctuating process. Second, $Z_{k,j}$ and $Z_{k,j'}$ for $j \neq j'$ are independent *a-priori*.

We can link the time-frequency process $\{Z_k\}_{k=1}^K$ with time-domain observation Y_k, using the Fourier matrix $\mathbf{F} \in \mathbb{C}^{J \times J}$ with $(F)_{j,l} = J^{-1/2} \exp(-i2\pi(l-1)j/J)$ and the inverse-Fourier matrix $\mathbf{W} \in \mathbb{C}^{J \times J}$, such that $\mathbf{FW} = I$

$$\begin{aligned} Y_k &= X_k + \varepsilon_k = \mathbf{W} Z_k + \varepsilon_k \\ \Rightarrow Y_k^F &= \mathbf{F} Y_k = \mathbf{FW} Z_k + \mathbf{F}\varepsilon_k = Z_k + \varepsilon_k^F, \end{aligned} \tag{3.3}$$

where $\varepsilon_k^F \sim \mathcal{N}(0, I(\sigma_\varepsilon^2))$, since $FI(\sigma_\varepsilon^2)F^* = I(\sigma_\varepsilon^2)$.

Along with the state-space model, we incorporate the data tapers to further reduce the variance of the spectral estimates. Specifically, we use M Slepian tapers, leading to the multitaper (MT) method that optimally balances the bias-variance trade-off via bandwidth adjustment [77, 85]. This essentially produces M *independent* set of state-space models,

$$Y_k^{(m),F} = Z_k^{(m)} + \varepsilon_k^{(m),F} \tag{3.4}$$

$$Z_k^{(m)} = Z_{k-1}^{(m)} + v_k^{(m)}, \tag{3.5}$$

where $v_k^{(m)} \sim \mathcal{CN}(0, I(\sigma_{v,j}^{2,(m)}))$, $Y_k^{(m)}$ corresponds to the m^{th} taper applied to Y_k, $Y_k^{(m),F} = \mathbf{F}Y_k^{(m)}$, and $Z_k^{(m)}$ represents the mth spectral eigen-coefficient of Z_k. The extensions of SSMT and similar frameworks modify Eqs. 3.4 or 3.5 [79, 80, 56, 20].

Based on Eqs. 3.4 and 3.5, we derive a Kalman filter algorithm for the estimation of $Z_{k,j}^{(m)}$ using Kalman gain $C_{k,j}^{(m)}$

$$Z_{k|k,j}^{(m)} = (1 - C_{k,j}^{(m)})Z_{k-1|k-1,j}^{(m)} + C_{k,j}^{(m)} Y_{k,j}^{(m),F} \tag{3.6}$$

$$\sigma_{k|k,j}^{2,(m)} = (1 - C_{k,j}^{(m)})(\sigma_{k-1|k-1,j}^{2,(m)} + \sigma_{v,j}^{2,(m)}), \tag{3.7}$$

where we focus on ω_j for simplicity and $C_{k,j}^{(m)}$ is given as

$$C_{k,j}^{(m)} = \frac{\sigma_{k-1|k-1,j}^{2,(m)} + \sigma_{v,j}^{2,(m)}}{\sigma_{\varepsilon}^{2,(m)} + \sigma_{k-1|k-1,j}^{2,(m)} + \sigma_{v,j}^{2,(m)}}. \tag{3.8}$$

The notation $k|s$ denotes the estimate on interval k given the data observed up to interval s. Finally, the SSMT spectrogram estimate at frequency ω_j on interval k is

$$f_k^{\text{SSMT}}(\omega_j) = M^{-1} \sum_{m=1}^{M} \|Z_{k|k,j}^{(m)}\|^2. \tag{3.9}$$

The parameters $\{\sigma_{v,j}^{(m),2}, \sigma_{\varepsilon}^{(m),2}\}_{m=1}^{M}$ are estimated via EM algorithm [82]. SSMT tracks nonstationarity with the *time-invariant* parameters $\sigma_{v,j}^{2,(m)}$, since $Z_{k|k,j}^{(m)} \neq Z_{k-1|k-1,j}^{(m)}$ and thus $f_k^{\text{SSMT}}(\omega_j) \neq f_{k-1}^{\text{SSMT}}(\omega_j)$.

3.3 Adaptive SSMT

Although SSMT can model slowly time-varying spectral dynamics, it is restrictive for highly fluctuating spectral dynamics. Such strong nonstationarity (or fluctuation) is common in EEG with *external intervention* to the brain or with change of the brain state during anesthesia/sleep [81]. Fig. 3-1 shows a snippet of human anesthesia EEG in which SSMT (Fig. 3-1(b)) cannot track the apparent dynamics shown in MT approach (Fig. 3-1(a)). However, the proposed ASSMT (Fig. 3-1(c)) is able to emphasize the spectral dynamics and denoise non-relevant background noise.

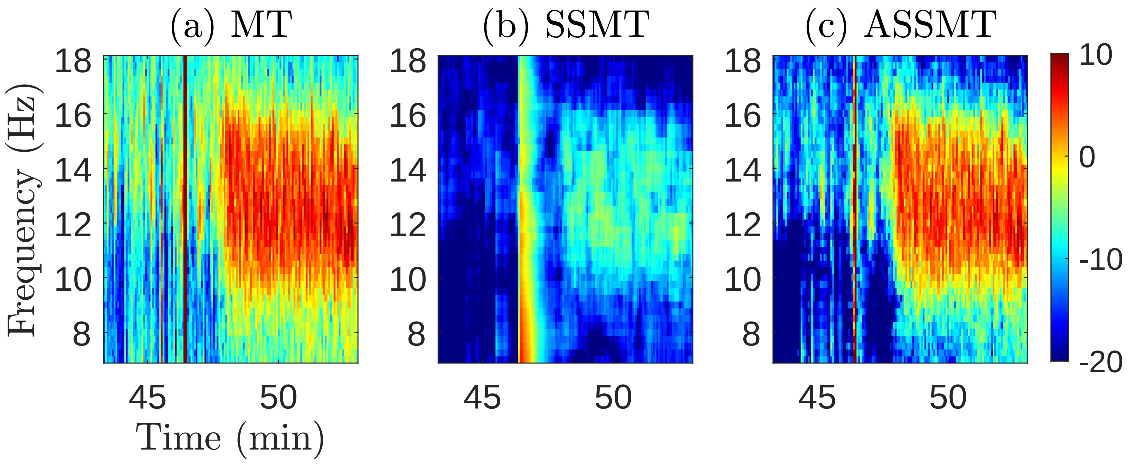

Figure 3-1: A spectrogram snippet estimated with (a) MT (b) SSMT (c) ASSMT.

The failure of SSMT is due to the time-invariant parameters, as the Kalman gain quickly converges to a steady-state value $C_{\infty,j}^{(m)}$ [86]. Denoting the observation prediction error as $\Delta Y_{k,j}^{(m)} = Y_{k,j}^{(m),F} - Z_{k-1|k-1,j}^{(m),F}$ and the change in the latent estimate as $\Delta Z_{k,j}^{(m)} = Z_{k|k,j}^{(m)} - Z_{k-1|k-1,j}^{(m)}$, we can express Eq. 3.6 as $\Delta Z_{k,j}^{(m)} = C_{\infty,j}^{(m)} \Delta Y_{k,j}^{(m)}$. Therefore, if $C_{\infty,j}^{(m)}$ is low as in SSMT for Fig. 3-1(b), f_k^{SSMT} cannot reliably track the spectral dynamics as $Z_{k|k,j}^{(m)}$ fails to reflect information in $Y_{k,j}^{(m),F}$.

To resolve this issue, adaptive SSMT (ASSMT) posits a time-varying state-space model, to allow the adaptive change of $\sigma_{v,j}^{(m),2}$. We start by rewriting Eq. 3.5 as

$$Z_{k,j}^{(m)} = Z_{k-1,j}^{(m)} + v_{k,j}^{(m)}, \tag{3.10}$$

where $v_{k,j}^{(m)} \sim \mathcal{CN}(0, I(\sigma_{v,k,j}^{2,(m)}))$, to indicate the state variance's dependence on time. We still use constant σ_ε^2 since we assume stationary background noise.

With the modified generative model, we now address *when* and *how* ASSMT tracks varying degrees of nonstationarity. We first quantify the notion of nonstationarity, and then propose an adaptive parameter estimation approach.

3.3.1 Measure of nonstationarity

We define $\|\widetilde{Y}_{k,j}^{(m),F}\|^2 = \mathbb{E}\|Y_{k,j}^{(m),F} - Y_{k-1,j}^{(m),F}\|^2$, the *expected* observation difference, as the measure of nonstationarity at window k. Intuitively, we use high fluctuation as a proxy for high nonstationarity. Specifically, we utilize $\|\widetilde{Y}_{k,j}^{(m),F}\|^2$ as 1) an indicator of high nonstationarity, i.e., when $\|\widetilde{Y}_{k,j}^{(m),F}\|^2$ exceeds a frequency-dependent threshold β_j and 2) a component in the parameter estimation procedure.

To estimate $\|\widetilde{Y}_{k,j}^{(m),F}\|^2$, we use *exponential moving average* (EMA), often used as a simple, yet effective approach to estimate the expectation in filtering literature [87]

$$\|\widetilde{Y}_{k,j}^{(m),F}\|^2 = (1-\alpha)\|\widetilde{Y}_{k-1,j}^{(m),F}\|^2 + \alpha\|Y_{k,j}^{(m),F} - Y_{k-1,j}^{(m),F}\|^2, \tag{3.11}$$

where $0 \leq \alpha \leq 1$ is a smoothing factor. We use it as a heuristic indicator to detect the presence of strong nonstationarity, independent from the estimation procedure of $Z_{k,j}^{(m)}$. The choice of α reflects the belief on the impulsiveness of nonstationarity and the volatile nature of the state transitions. With large α, $\|\widetilde{Y}_{k,j}^{(m),F}\|^2$ is sensitive to instantaneous fluctuation.

3.3.2 Estimation of parameters & adaptive thresholding

We now examine how to use $\|\widetilde{Y}_{k,j}^{(m),F}\|^2$ to set β_j and subsequently estimate the parameters. Using Eq. 3.4, we have

$$\begin{aligned}\|\widetilde{Y}_{k,j}^{(m),F}\|^2 &= \mathbb{E}\|Y_{k,j}^{(m),F} - Y_{k-1,j}^{(m),F}\|^2 \\ &= \mathbb{E}\|(Z_{k,j}^{(m)} - Z_{k-1,j}^{(m)}) + (\varepsilon_k^F - \varepsilon_{k-1}^F)\|^2 \\ &= \mathbb{E}\|Z_{k,j}^{(m)} - Z_{k-1,j}^{(m)}\|^2 + \mathbb{E}\|\varepsilon_k^F - \varepsilon_{k-1}^F\|^2,\end{aligned} \tag{3.12}$$

where we used the uncorrelatedness of the two differences. Next, we use the fact that 1) $\mathbb{E}\|Z_{k,j}^{(m)} - Z_{k-1,j}^{(m)}\|^2 = \sigma_{v,j}^{2,(m)}$ from Eq. 3.5 and 2) ε_k^F and ε_{k-1}^F are independent with variance $\sigma_\varepsilon^{2,(m)}$, hence $\mathbb{E}\|\varepsilon_k^F - \varepsilon_{k-1}^F\|^2 = 2\sigma_\varepsilon^2$, which leads to

$$\mathbb{E}\|Y_{k,j}^{(m),F} - Y_{k-1,j}^{(m),F}\|^2 = \sigma_{v,k,j}^{2,(m)} + 2\sigma_\varepsilon^{2,(m)}. \tag{3.13}$$

This establishes the connection between the nonstationarity metric, $\|\widetilde{Y}_{k,j}^{(m),F}\|^2$, and the two sources of variance.

With Eq. 3.11 and Eq. 3.13, we can now estimate $\sigma_{v,k,j}^{2,(m)}$. For σ_ε^2, we utilize $\widehat{\sigma}_{\varepsilon,j}^{2,(m)}$ estimated from SSMT. This leads to

$$\widehat{\sigma}_{v,k,j}^{2,(m)} = \|\widetilde{Y}_{k,j}^{(m),F}\|^2 - 2\hat{\sigma}_\varepsilon^{2,(m)}. \tag{3.14}$$

We further lower bound the $\widehat{\sigma}_{v,k,j}^{2,(m)}$ for two reasons. First, we impose that the signal-to-noise ratio (SNR), $\gamma_{k,j} = \sigma_{v,k,j}^{2,(m)}/\sigma_\varepsilon^2$, is greater than a minimum baseline SNR, i.e., $\gamma_{k,j} \geq \gamma_{k,j}^{\min}$. Second, we require nonnegative $\widehat{\sigma}_{v,k,j}^{2,(m)}$. Since SSMT estimates the baseline properties of the nonstationary data through EM, we naturally set $\gamma_{k,j}^{\min} = \widehat{\sigma}_{v,j}^{2,(m),\text{SSMT}}/\widehat{\sigma}_\varepsilon^2$, which yields

$$\widehat{\sigma}_{v,k,j}^{2,(m)} = \max(\|\widetilde{Y}_{k,j}^{(m),F}\|^2 - 2\hat{\sigma}_\varepsilon^{2,(m)}, \hat{\sigma}_{v,j}^{2,(m),\text{SSMT}}). \tag{3.15}$$

This procedure obviates the need for EM beyond the initial phase for estimating $\hat{\sigma}_\varepsilon^2$ and $\hat{\sigma}_{v,j}^{2,(m),\text{SSMT}}$. Although EM can be used for estimating the time-varying parameters, it requires multiple forward/backward passes through the entire data, which is computationally expensive. In addition, with every new observation, $\widehat{\sigma}^{2,(m)}_{v,k,j}$ in the past data need to re-estimated.

3.3.3 Estimation of nonstationary spectra

We use Kalman filter with $\widehat{\sigma}^{2,(m)}_{v,k,j}$ to estimate the spectrogram $f_k^{\text{ASSMT}}(\omega_j)$. ASSMT therefore operates with two different modes depending on $\beta_j = 2\hat{\sigma}^{2,(m)}_{\varepsilon} + \hat{\sigma}^{2,(m),\text{SSMT}}_{v,j}$. If $\|\widetilde{Y}^{(m),F}_{k,j}\|^2 \geq \beta_j$, ASSMT adaptively uses larger state variance to track high nonstationarity. Given $\sigma^{2,(m)}_{k-1|k-1,j}$ and fixed $\widehat{\sigma}^2_{\varepsilon}$ in Eq. 3.8, we observe that the *increase* in $\widehat{\sigma}^{2,(m)}_{v,k,j}$ leads to the *increase* in $C^{(m)}_{k,j}$. This agrees with our intuition, since we want the Kalman gain to *increase* such that $\Delta Z^{(m)}_{k,j}$ explains a greater portion of $\Delta Y^{(m)}_{k,j}$. For $\|\widetilde{Y}^{(m),F}_{k,j}\|^2 < \beta_j$, ASSMT simply uses the *baseline* $\hat{\sigma}^{2,(m),\text{SSMT}}_{v,j}$. This explains how ASSMT with adaptive state variance is able to track strong nonstationarity.

3.4 Results

We apply ASSMT to two datasets: 1) nonstationary simulated data and 2) human EEG data under propofol anesthesia. We compare the spectrogram estimates between MT, SSMT, and ASSMT. All spectrograms are in dB scale.

3.4.1 Application to Simulated Data

We simulate the data as a superposition of amplitude-modulated process $y_{t,1}$ and frequency-modulated processes $y_{t,2}$ with high dynamic range. The process $y_{t,1}$ is generated from an AR(6) process centered at 11 Hz and modulated by a cosine function of $f_0 = 0.02$ Hz. The process $y_{t,2}$ is generated from ARMA(6,4) with varying pole loci. More details on $y_{t,1}, y_{t,2}$ can be found in [80]. The observations are given by $y_t = y_{t,1}\cos(2\pi f_0 t) + y_{t,2} + \sigma v_t$, where $v_t \sim \mathcal{N}(0,1)$ and σ is chosen to achieve an SNR of 30 dB [80]. For MT, we use 6-second windows with 50% overlap and $M = 3$ Slepian tapers, and 6-second non-overlapping windows for both SSMT and ASSMT.

For SSMT, we use the entire data to estimate the parameters. For ASSMT, we use initial 300 seconds of the data to compute the baseline parameters and use $\alpha = 0.95$.

Fig. 3-2 shows the ground truth and the estimated spectrograms. Although MT captures the spectral dynamics reasonably well, it picks up background noise and spectral artifacts (i.e., vertical lines), and induces mixing of adjacent frequency bands due to low resolution. SSMT (Fig. 3-2 (c)) resolves these issues with sharper spectral localization and removal of spectral artifacts, benefitting from the state-space prior.

ASSMT also shares the artifact rejection and noise reduction properties of SSMT. Moreover, ASSMT performs better denoising, as evident in $5 \sim 20$ Hz frequency band. The Itakura-Saito divergence (IS) [88] between the ground truth and the spectrogram estimate also confirms this observation, with ASSMT attaining the lowest value ($\text{IS}^{\text{MT}} = 6.51$, $\text{IS}^{\text{SSMT}} = 3.16$, $\text{IS}^{\text{ASSMT}} = \mathbf{2.75}$). We attribute this difference to SSMT's fixed high $\widehat{\sigma}_{v,j}^{2,(m),\text{SSMT}}$ and consequently high Kalman gain across $5 \sim 20$ Hz. This is because for any given frequency component, there exists strong nonstationary regime within the parameter estimation window (the entire data), which inevitably leads to high $\widehat{\sigma}_{v,j}^{2,(m),\text{SSMT}}$, $\forall j$. However, since ASSMT does not commit to a fixed value, it can adaptively change the parameter at different regimes of the data.

3.4.2 Application to anesthesic EEG data

The EEG was recorded ($f_s = 250$ Hz) from a volunteer receiving propofol administered with increasing rate, followed by the decreasing rate [81]. This setup induces altering states of unconsciousness (or brain states), resulting in varying levels of nonstationarity. We used $M = 5$ Slepian tapers, $J = 1,000$ samples. For SSMT, we estimate the spectrogram based on the parameters estimated from 1) the initial 4 minutes of data (Fig. 3-3 (b)) and 2) the entire data (Fig. 3-3 (c)). For ASSMT, we use the initial 4 minutes to compute the baseline parameters. Fig. 3-3 shows the estimated spectrograms.

SSMT vs. ASSMT SSMT based on the initial 4 minutes of the data (Fig. 3-3 (b)) produces low $\widehat{\sigma}_{v,j,k}^{2,(m)}$ estimates due to absence of spectral dynamics for the baseline state. Although it denoises the background noise well as a result, it misses most of

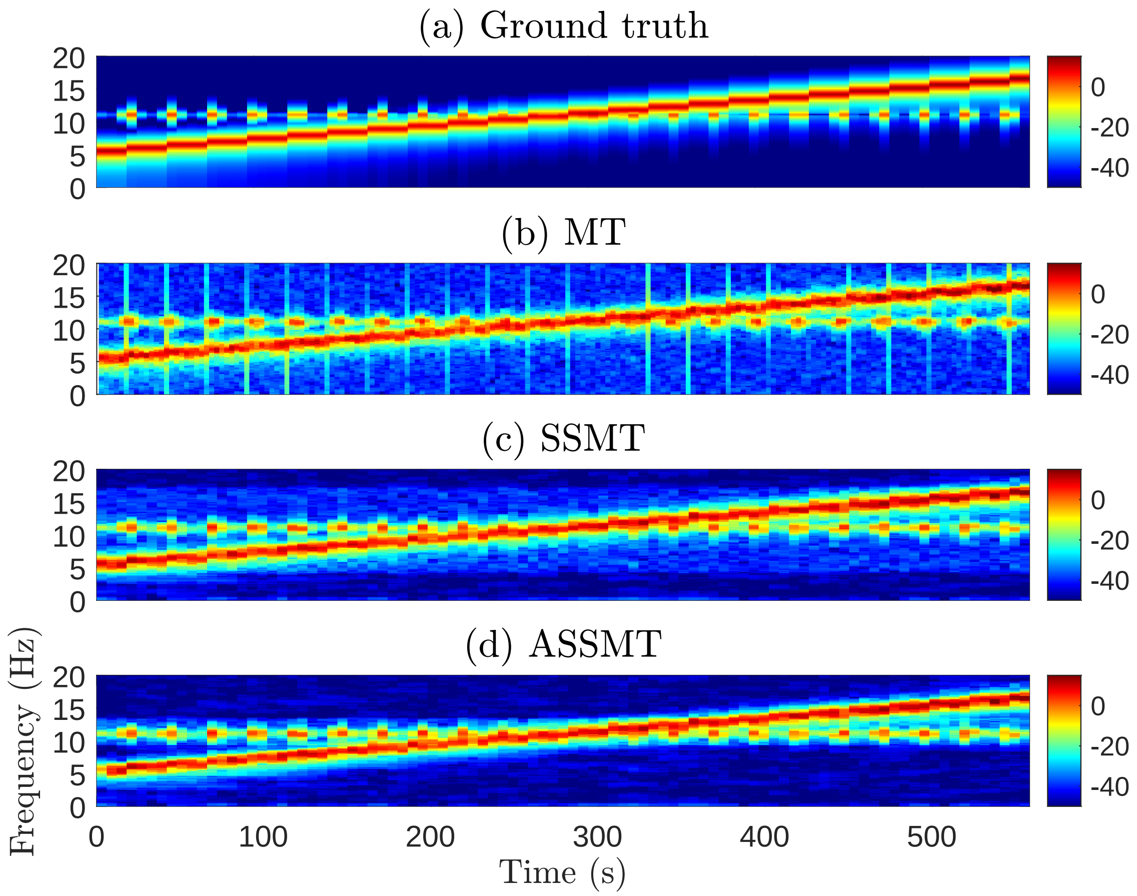

Figure 3-2: Spectrograms for simulated data (a) ground truth (b) MT (c) SSMT (d) ASSMT with $\alpha = 0.95$.

the strong spectral fluctuation, as evident in extreme denoising of the spectra from 40 min to 120 min.

This can be mitigated by applying EM to a different section of the data, or as in our case, to the entire data. Due to high $\widehat{\sigma}^{2,(m)}_{v,j,k}$ estimates, SSMT is better equipped at capturing the strong nonstationarity (Fig. 3-3 (c)). However, it fails to denoise the baseline state ($0 \sim 40$ min) due to high Kalman gain, as similarly observed in the simulation. These results identifies a drawback of the time-invariant paradigm, as different $\widehat{\sigma}^{2,(m)}_{v,j,k}$ estimated from different sections could yield significantly different spectrogram estimates.

In contrast, ASSMT adaptively denoises the spectrogram (Fig. 3-3 (d-e)) even with the initial baseline parameters, the same setting for which SSMT failed (Fig. 3-3

]

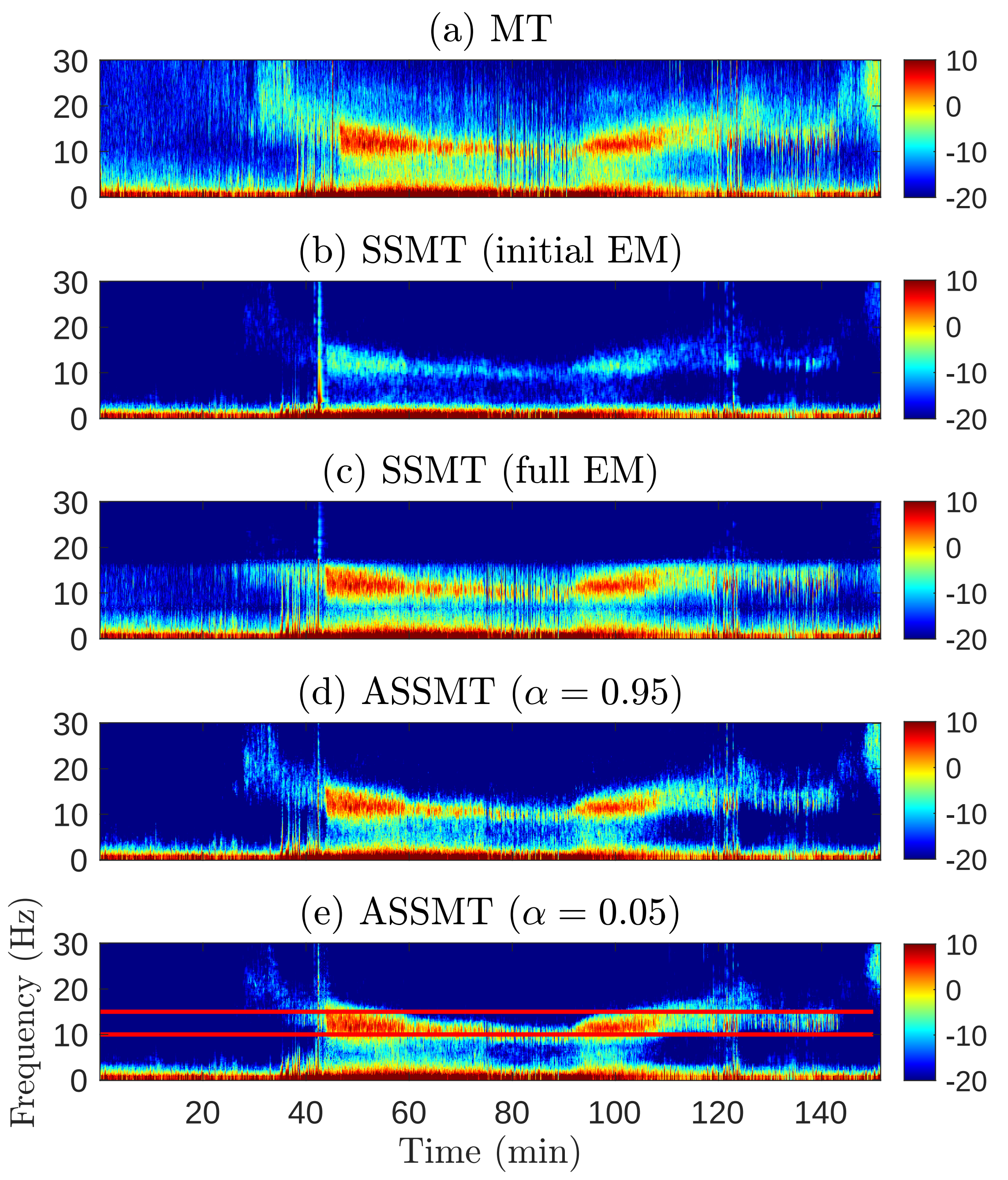

Figure 3-3: Propofol anesthesia EEG spectrograms (a) MT (b) SSMT with initial 4-min EM window (c) SSMT with full data EM window (d) ASSMT with $\alpha = 0.95$ (e) ASSMT with $\alpha = 0.05$. Both ASSMT use initial 4-min EM window. Red horizontal lines correspond to 10 and 15 Hz.

(b)). The time-varying nature of the model allows it to switch between low $\widehat{\sigma}^{2,(m)}_{v,j,k}$, excelling at denoising the background noise, and high $\widehat{\sigma}^{2,(m)}_{v,j,k}$, excelling at capturing the spectral fluctuation, without fully committing to either parameters.

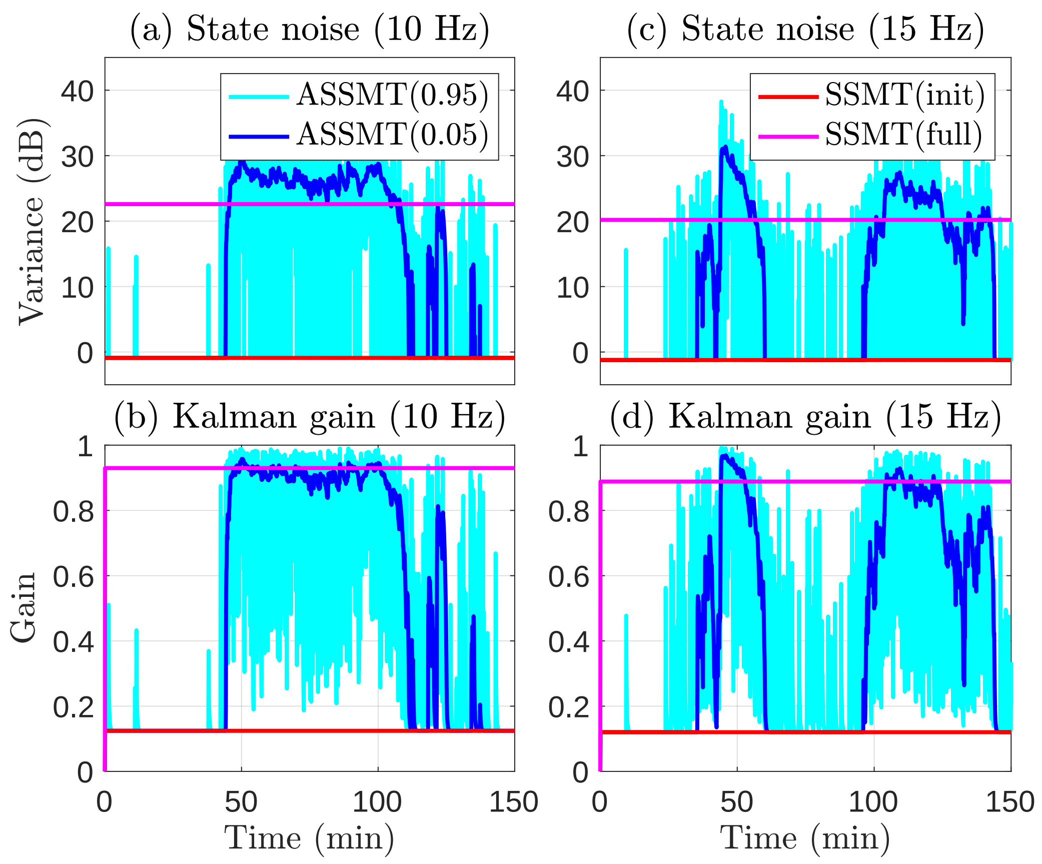

Figure 3-4: Kalman gain and state noise variance for SSMT and ASSMT for the first taper $m = 1$. (a) Kalman gain and (b) State noise variance at 10 Hz. (c) Kalman gain and (d) state noise variance at 15 Hz.

Another important difference is the computation time. The inference procedure of EM on the entire data & Kalman filtering is $\sim$ 400 seconds. For ASSMT, however, the procedure takes $\sim$ 6 seconds, boasting huge computational time reduction, making it more suitable for real-time application.

Effect of Kalman gain To further understand ASSMT, we analyze the evolution of state variance and the Kalman gain across time at the representative frequency bands (10 and 15 Hz), shown in Fig. 3-4. For both frequency bands, we observe that the state variance and consequently the Kalman gain is increased above the threshold, in tandem with the large spectral fluctuation. As discussed previously, a large Kalman gain is imperative for this purpose. The SSMT Kalman gain stays fixed at either 0.1 (for initial 4-min estimation, red in Fig. 3-4) or 0.9 (for entire data estimation, magenta in Fig. 3-4).

Effect of $\boldsymbol{\alpha}$ We observe that the denoising performance of ASSMT is robust towards the choice of α. To further understand how α affects the filtering/denoising operation, we analyze how $\widehat{\sigma}^{2,(m)}_{v,j,k}$ and Kalman gain change over time. ASSMT with $\alpha = 0.95$ (light-blue) is dominated by the heavy fluctuation, $\|Y^{(m),F}_{k,j} - Y^{(m),F}_{k-1,j}\|^2$. In contrast, ASSMT with small $\alpha = 0.05$ (blue) shows smoother state variance and Kalman gain. In this case, ASSMT starts adapting when there is significant evidence for nonstationarity. This explains why ASSMT with $\alpha = 0.05$ has a superior denoising effect, as it is less susceptible to instantaneous fluctuations (40 min., $0 \sim 10$ Hz) and background noise ($75 \sim 95$ min., $2 \sim 10$ Hz).

3.5 Conclusion

We introduced an adaptive state-space multitaper (ASSMT) framework, a state-space model for adaptively estimating spectral dynamics in the nonstationarity time series. By relaxing the time-invariant parameters assumption and proposing an adaptive parameter estimation scheme, we demonstrated that ASSMT was able to capture strong power fluctuations much more reliably compared to SSMT.

Chapter 4

Time-frequency analysis with Gaussian Process

Primer To impose smoothness on the latent process, the previous chapters focused on using stochastic continuity on the Fourier coefficients across different windows. While these frameworks exhibit superior denoising performance in the time-frequency domain compared to the baseline MT spectrograms, there is no guarantee that this continuity is maintained in the time domain. As we will see in this chapter, it does not. Consequently, the recovered time-domain estimates, obtained by performing inverse Fourier transform on estimated Fourier coefficients for each window, are discontinuous and distorted around the window boundaries. To resolve this issue, we propose a new *time-domain* generative model, the Piecewise Locally Stationary Oscillation (PLSO) framework, which guarantees stochastic continuity in both the time and time-frequency domain. This chapter is adapted from the following work

- Andrew H. Song, Demba Ba, and Emery N. Brown, *PLSO: A generative framework for decomposing nonstationary timeseries into piecewise stationary oscillatory components*, Uncertainty in Artificial Intelligence (UAI), 2021

4.1 Introduction

With the collection of long time-series now common, in areas such as neuroscience and geophysics, it is important to develop an *inference* framework for data where the stationarity assumption is too restrictive. We restrict our attention to data 1) with spectral properties that change slowly over time and 2) for which decomposition into several oscillatory components is warranted for interpretation, often the case in electroencephalogram (EEG) or electrophysiology recordings. One can use bandpass filtering [89] or methods such as the empirical mode decomposition [90, 91] for these purposes. However, due to the *absence* of a generative model, these methods lack a framework for performing inference. Another popular approach is to perform inference in the time-frequency (TF) domain on the short-time Fourier transform (STFT) of the data, assuming stationarity within small intervals. This has led to a rich literature on inference in the TF domain, such as [92]. A drawback is that most of these methods focus on estimates for the power spectral density (PSD) and lose important phase information. To recover the time-domain estimates, additional algorithms are required [93].

This motivates us to explore time-domain generative models that allow time-domain inference and decomposition into oscillatory components. We can find examples based on the stationarity assumption in the signal processing/Gaussian process (GP) communities. A superposition of stochastic harmonic oscillators, where each oscillator corresponds to a frequency band, is used in the processing of speech [94] and neuroscience data [95, 96]. In GP literature [46], the spectral mixture (SM) kernel [97, 98] models the data as samples from a GP, whose kernel consists of the superposition of localized and frequency-modulated kernels.

These time-domain models can be applied to nonstationary data by partitioning them into stationary intervals and performing time-domain inference within each interval. However, we are faced with a different kind of challenge. As the inference is localized within each interval, the time-domain estimates in different intervals are independent conditioned on the data and do not reflect the dependence across intervals. This also causes discontinuity/distortion of the time-domain estimates near the interval boundaries, and consequently any quantities derived from these estimates.

To address these shortcomings, we propose a generative framework for data with slow time-varying spectra, termed the Piecewise Locally Stationary Oscillation (PLSO) framework[1]. The main contributions are:

Generative model for piecewise stationary, oscillatory components PLSO models time-series as the superposition of piecewise stationary, oscillatory components. This allows time-domain inference on each component and estimation of the time-varying spectra.

Continuity across stationary intervals The state-space model that underlies PLSO strikes a balance between ensuring time-domain continuity across piecewise stationary intervals and stationarity within each interval. Moreover, by imposing stochastic continuity on the interval-level, PLSO learns underlying smooth time-varying spectra accurately.

Inference procedure We propose a two-stage inference procedure for the time-varying spectra and the time-series. By leveraging the Markovian dynamics, the algorithm combines Kalman filter theory [99] and inexact accelerated proximal gradient approach [100].

In Section 6.2 we introduce necessary background, followed by the PLSO framework in Section 4.3. In Section 6.3, we discuss inference for PLSO. In Section 4.5, we discuss how PLSO relates to other frameworks. In Section 6.5, we present experimental results and conclude in Section 4.7.

4.2 Background

4.2.1 Notation

We use $j \in \{1, \ldots, J\}$ and $k \in \{1, \ldots, K\}$ to denote frequency and discrete-time sample index, respectively. We use $\omega \in [-\pi, \pi]$ for normalized frequency. The j^{th}

[1]Code is available at https://github.com/andrewsong90/plso.git

latent process centered at $\omega = \omega_j$ is denoted as $\mathbf{z}_j \in \mathbb{C}^K$, with $\mathbf{z}_{j,k} \in \mathbb{C}$ denoting the k^{th} sample of $\mathbf{z}_j$ and $\mathbf{z}_{j,k}^{\Re}$, $\mathbf{z}_{j,k}^{\Im}$ its real and imaginary parts. We also represent $\mathbf{z}_{j,k}$ as a $\mathbb{R}^2$ vector, $\widetilde{\mathbf{z}}_{j,k} = [\mathbf{z}_{j,k}^{\Re}, \mathbf{z}_{j,k}^{\Im}]^{\mathrm{T}}$. The elements of $\mathbf{z}_j$ are denoted as $\mathbf{z}_{j,k:k'} = [\mathbf{z}_{j,k}, \ldots, \mathbf{z}_{j,k'}]^{\mathrm{T}}$. The state covariance matrix for $\mathbf{z}_{j,k}$ is defined as $\mathbf{P}_k^j = \mathbb{E}[\widetilde{\mathbf{z}}_{j,k}\,(\widetilde{\mathbf{z}}_{j,k})^{\mathrm{T}}]$. To express an enumeration of variables, we use $\{\cdot\}$ and drop first/last index for simplicity, e.g. $\{\mathbf{z}_j\}_j$ instead of $\{\mathbf{z}_j\}_{j=1}^J$.

We use $\mathbf{y}_k$ and $\mathbf{z}_{j,k}$ for the discrete-time counterpart of the continuous observation and latent process, $y(t)$ and $z_j(t)$. With the sampling frequency $f_s = 1/\Delta$, we have $\mathbf{y}_k = y(k\Delta)$, $\mathbf{z}_{j,k} = z_j(k\Delta)$, and $T = K\Delta$.

4.2.2 Piecewise local stationarity

The concept of *piecewise local stationarity* (PLS) for nonstationary time-series with slowly time-varying spectra [83] plays an important role in PLSO. A stationary process has a constant mean and a covariance function which depends only on the difference between two time points.

For our purposes, it suffices to understand the following on PLS: 1) It includes local stationary [101, 102] and amplitude-modulated stationary processes. 2) A PLS process can be approximated as a piecewise stationary (PS) process (Theorem 1 of [83])

$$z(t) = \sum_{m=1}^{M} \mathbf{1}(u_m \le t < u_{m+1}) \cdot z^m(t), \tag{4.1}$$

where $z^m(t)$ is a continuous stationary process and the boundaries are $0 = u_1 < \cdots < u_{M+1} = T$. Note that Eq. 4.1 does not guarantee continuity across different PS intervals,

$$\lim_{t \to u_m^-} z(t) = \lim_{t \to u_m^-} z^{m-1}(t) \neq \lim_{t \to u_m^+} z^m(t) = \lim_{t \to u_m^+} z(t).$$

4.3 The PLSO model and its mathematical properties

Building on the Theorem 1 of [83], PLSO models nonstationary data as PS processes. It is a superposition of J different PS processes $\{\mathbf{z}_j\}_j$, with $\mathbf{z}_j$ corresponding to an oscillatory process centered at frequency ω_j. PLSO also guarantees stochastic continuity across PS intervals. We show that piecewise stationarity and continuity across PS intervals are two competing objectives and that PLSO strikes a balance between them, as discussed in Section 4.3.2.

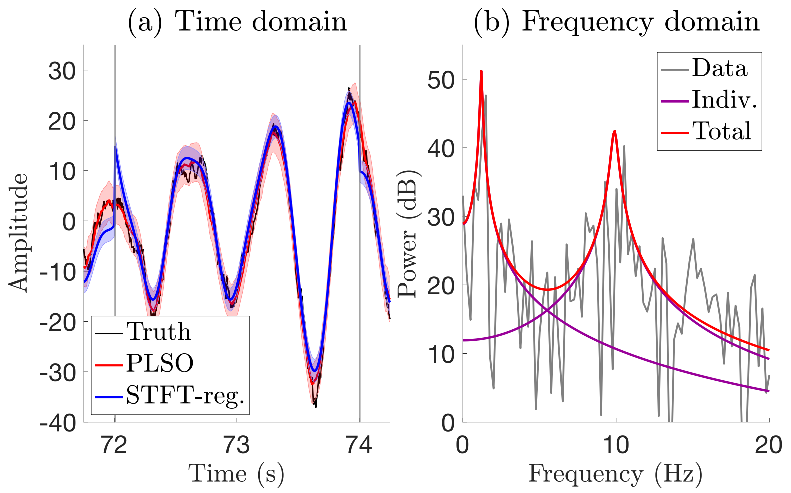

Figure 4-1: A simulated example. (a) Time domain. Data (black) around boundaries (gray) and inferred oscillatory components using PLSO (red) and regularized STFT (blue). (b) Frequency domain. Spectrum of the data (gray), PLSO components for $J = 2$ (purple) and their sum (red).

Fig. 4-1 shows an example of the PLSO framework applied to simulated data. In the time domain, the oscillation inferred using the regularized STFT (blue) [19], which imposes stochastic continuity on the STFT coefficients, suffers from discontinuity/distortion near window boundaries, whereas that inferred by PLSO (red) does not. In the frequency domain, each PLSO component corresponds to a localized spectrum $S_j(\omega)$, the sum of which is the PSD $\gamma(\omega)$, and is fit to the data STFT (or periodogram) $I(\omega)$. We start by introducing the PLSO model for a single window.

4.3.1 PLSO for stationary data

As a building block for PLSO, we use the discrete stochastic harmonic oscillator for a stationary time series [103, 94, 95]. The data $\mathbf{y}$ are assumed to be a superposition of J *independent* zero-mean components

$$\begin{aligned} \widetilde{\mathbf{z}}_{j,k} &= \rho_j \mathbf{R}(\omega_j)\widetilde{\mathbf{z}}_{j,k-1} + \varepsilon_{j,k} \\ \mathbf{y}_k &= \sum_{j=1}^{J} \mathbf{z}_{j,k}^{\Re} + \nu_k, \end{aligned} \tag{4.2}$$

where $\mathbf{R}(\omega_j) = \begin{pmatrix} \cos(\omega_j) & -\sin(\omega_j) \\ \sin(\omega_j) & \cos(\omega_j) \end{pmatrix}$, $\varepsilon_{j,k} \sim \mathcal{N}(0, \alpha_j \mathbf{I}_{2\times 2})$, and $\nu_k \sim \mathcal{N}(0, \sigma_\nu^2)$, correspond to the rotation matrix, the state noise, and the observation noise, respectively. The imaginary component $\mathbf{z}_{j,k}^{\Im}$, which does not directly contribute to $\mathbf{y}_k$, can be seen as the auxiliary variable to write $\mathbf{z}_j$ in recursive form using $\mathbf{R}(\omega_j)$ [104]. We assume $\mathbf{P}_1^j = \sigma_j^2 \cdot \mathbf{I}_{2\times 2}\ \forall j$.

We reparameterize ρ_j and α_j, using lengthscale l_j and power σ_j^2, such that $\rho_j = \exp(-\Delta/l_j)$ and $\alpha_j = \sigma_j^2(1-\rho_j^2)$. Theoretically, this establishes a connection to 1) the stochastic differential equation [105, 106], detailed in **Appendix A**, and 2) a superposition of frequency-modulated and localized GP kernels, similar to SM kernel [97]. Practically, this ensures that $\rho_j < 1$, and thus stability of the process.

Given Eq. 4.2, we can readily express the frequency spectra of PLSO in each interval, through the autocovariance function. The autocovariance of $\mathbf{z}_j$ is given as $Q_j(n') = \mathbb{E}[\mathbf{z}_{j,k}^{\Re}\mathbf{z}_{j,k+n'}^{\Re}] = \sigma_j^2 \cos(\omega_j n') \exp(-n'\Delta/l_j)$. It can also be thought of as an *exponential* kernel, frequency-modulated by ω_j. The spectra for $\mathbf{z}_j$, denoted as $S_j(\omega)$, is obtained by taking the Fourier transform (FT) of $Q_j(n')$

$$\begin{aligned} S_j(\omega) &= \sum_{n'=-\infty}^{\infty} Q_j(n') \exp(-i\omega n') = \varphi_j(\omega) + \varphi_j(-\omega) \\ \varphi_j(\omega) &= \frac{\sigma_j^2 \left(1 - \exp(-2\Delta/l_j)\right)}{1 + \exp(-2\Delta/l_j) - 2\exp(-\Delta/l_j)\cos(\omega - \omega_j)}, \end{aligned}$$

with the detailed derivation in **Appendix B**.

Given $S_j(\omega)$, we can show that PSD $\gamma(\omega)$ of the entire process $\sum_{j=1}^J \mathbf{z}_j$ is simply $\gamma(\omega) = \sum_{j=1}^J S_j(\omega)$. First, since $\mathbf{z}_j$ is independent across j, the autocovariance can be simplified, i.e., $\mathbb{E}[\sum_j \mathbf{z}_{j,k}^{\Re} \sum_j \mathbf{z}_{j,k+n'}^{\Re}] = \sum_j \mathbb{E}[\mathbf{z}_{j,k}^{\Re}\mathbf{z}_{j,k+n'}^{\Re}]$. Next, using the linearity of FT, we can conclude that $\gamma(\omega)$ is a superposition of individual spectra.

4.3.2 PLSO for nonstationary data

If $\mathbf{y}$ is nonstationary, we can still apply stationary PLSO of Eq. 4.2 for the time-domain inference. However, this implies constant spectra for the entire data ($S_j(\omega)$ and $\gamma(\omega)$ do not depend on k), which is not suitable for nonstationary time-series for which we want to track spectral dynamics. This point is further illustrated in Section 6.5.

We therefore segment $\mathbf{y}$ into M non-overlapping PS intervals, indexed by $m \in \{1, \ldots, M\}$, of length N, indexed by $n \in \{1, \ldots, N\}$, such that $K = MN$. We then apply the stationary PLSO to each interval, with additional Markovian dynamics imposed on $\sigma_{j,m}^2$,

$$\begin{aligned} \log(\sigma_{j,m}^2) &= \log(\sigma_{j,m-1}^2) + \eta_{j,m} \\ \widetilde{\mathbf{z}}_{j,mN+n} &= \rho_j \mathbf{R}(\omega_j)\widetilde{\mathbf{z}}_{j,mN+(n-1)} + \varepsilon_{j,mN+n} \\ \mathbf{y}_{mN+n} &= \sum_{j=1}^J \mathbf{z}_{j,mN+n}^{\Re} + \nu_{mN+n}, \end{aligned} \tag{4.3}$$

where $\varepsilon_{j,mN+n} \sim \mathcal{N}(0, \sigma_{j,m}^2(1-\rho_j^2)\mathbf{I}_{2\times 2})$, $\eta_{j,m} \sim \mathcal{N}(0, 1/\sqrt{\lambda})$ and $\nu_{mN+n} \sim \mathcal{N}(0, \sigma_\nu^2)$. We define $\mathbf{P}_{m,n}^j$ as the covariance of $\widetilde{\mathbf{z}}_{j,mN+n}$, with $\mathbf{P}_{1,1}^j = \sigma_{j,1}^2\mathbf{I}_{2\times 2}, \forall j$. The graphical model is shown in Fig. 4-2.

We can understand PLSO as providing a *parameterized* spectrogram defined by $\theta = \{\lambda, \sigma_\nu^2, \{l_j\}_j, \{\omega_j\}_j\}$ and $\{\sigma_{j,m}^2\}_{j,m}$ of the time-domain generative model. The lengthscale l_j controls the bandwidth of the j^{th} process, with *larger* l_j corresponding to *narrower* bandwidth. The variance $\sigma_{j,m}^2$ controls the power of $\mathbf{z}_j$ and changes across different intervals, resulting in time-varying spectra $S_j^{(m)}(\omega)$ and PSD $\gamma^{(m)}(\omega)$. The center frequency ω_j, and $-\omega_j$, at which $S_j^{(m)}(\omega)$ is maximized, controls the modulation frequency.

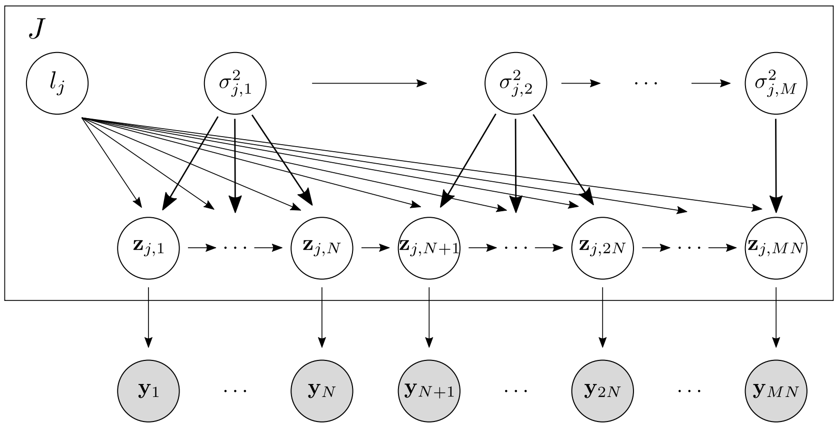

Figure 4-2: The graphical model for PLSO.

As discussed previously, the segmentation approach for nonstationary time-series produces distortion/discontinuity artifacts around interval boundaries - The PLSO, as described by Eq. 4.3, resolves these issues gracefully. We now analyze two mathematical properties, *stochastic continuity* and *piecewise stationarity*, to gain more insights on how PLSO accomplishes this.

Stochastic continuity

We discuss two types of stochastic continuity, 1) across the interval boundaries and 2) on $\{\sigma^2_{j,m}\}$.

Continuity across the interval boundaries In PLSO, the state-space model (Eq. 4.3) provides stochastic continuity across different PS intervals. The following proposition rigorously explains stochastic continuity for PLSO.

Proposition 1. *For a given m, as $\Delta \to 0$, the samples on either side of the interval boundary, which are $\widetilde{\mathbf{z}}_{j,(m+1)N}$ and $\widetilde{\mathbf{z}}_{j,(m+1)N+1}$, converge to each other in mean square,*

$$\lim_{\Delta \to 0} \mathbb{E}[\Delta\widetilde{\mathbf{z}}_{j,(m+1)N}\Delta\widetilde{\mathbf{z}}^T_{j,(m+1)N}] = 0,$$

where we use $\Delta\widetilde{\mathbf{z}}_{j,(m+1)N} = \widetilde{\mathbf{z}}_{j,(m+1)N+1} - \widetilde{\mathbf{z}}_{j,(m+1)N}$.

Proof. We use the connection between PLSO, which is a discrete-time model, and its continuous-time counterpart. It suffices to show that $\lim_{\Delta\to 0}\exp(\mathbf{F}\Delta) = \mathbf{I}_{2\times 2}$ and

$\lim_{\Delta\to 0}\mathbb{E}[\varepsilon_{j,(m+1)N+1}\varepsilon^{\mathrm{T}}_{j,(m+1)N+1}] = \mathbf{0}$. We have,

$$\lim_{\Delta\to 0}\exp(\mathbf{F}\Delta) = \mathbf{I}_{2\times 2} + \lim_{\Delta\to 0}\sum_{k=1}^{\infty}\frac{\Delta^k}{k!}\mathbf{F}^k = \mathbf{I}_{2\times 2}$$
$$\lim_{\Delta\to 0}\mathbb{E}[\varepsilon_{j,(m+1)N+1}\varepsilon^{\mathrm{T}}_{j,(m+1)N+1}]/\sigma^2_{j,m+1}$$
$$= \lim_{\Delta\to 0}\int_0^{\Delta}\exp\left(\mathbf{F}(\Delta-\tau)\right)\exp\left(\mathbf{F}(\Delta-\tau)\right)^{\mathrm{T}}d\tau = \mathbf{0}.$$

Since this implies $\lim_{\Delta\to 0}\mathbb{E}[\Delta\tilde{\mathbf{z}}_{j,(m+1)N}\Delta\tilde{\mathbf{z}}^{\mathrm{T}}_{j,(m+1)N}] = 0$, we have convergence in mean square. □

This matches our intuition that as $\Delta \to 0$, the adjacent samples from the same process should coverge to each other. For PS approaches without the sample-level continuity, even with the interval-level constraint [107, 19, 56, 108, 109], convergence is not guaranteed.

We can interpret the continuity in the context of posterior for $\mathbf{z}_j$. For PS approaches without continuity, we have

$$p(\{\mathbf{z}_j\}_j\,|\mathbf{y}) \propto \prod_{m=1}^{M} p(\left\{\mathbf{z}_{j,(m-1)N+1:mN}\right\}_j\,|\mathbf{y}_{(m-1)N+1:mN}), \tag{4.4}$$

where $\{\sigma^2_{j,m}\}_{j,m}$ and θ are omitted for notational ease. This is due to $p(\{\mathbf{z}_j\}_j) = \prod_{m=1}^{M} p(\{\mathbf{z}_{j,(m-1)N+1:mN}\}_j)$, as a result of *absence* of continuity across the intervals. Consequently, the inferred time-domain estimates are conditionally independent across the intervals. On the contrary, in PLSO, the time-domain estimates depend on the entire $\mathbf{y}$, not just on a subset.

Continuity on $\sigma^2_{j,m}$ For a given j, we impose stochastic continuity on $\log\sigma^2_{j,m}$. Effectively, this pools together estimates of $\{\sigma^2_{j,m}\}_m$ to 1) prevent overfitting to the noisy data spectra and 2) estimate smooth dynamics of $\{\sigma^2_{j,m}\}_m$. The use of $\log\sigma^2_{j,m}$ ensures that $\sigma^2_{j,m}$ is non-negative.

The choice of λ dictates the smoothness of $\{\sigma^2_{j,m}\}_m$, with the two extremes corresponding to the familiar dynamics. If $\lambda \to 0$, we treat each window independently. If $\lambda \to \infty$, we treat the data as stationary, as the constraint forces $\sigma^2_{j,m} = \sigma^2_j$, $\forall m$.

Practically, the smooth regularization prevents artifacts in the spectral analysis, arising from sudden motion or missing data, as demonstrated in Section 6.5.

Piecewise stationarity

For the m^{th} window to be piecewise stationary, the initial state covariance matrix $\mathbf{P}^j_{m,1}$ should be the *steady-state* covariance matrix for the window, denoted as $\mathbf{P}^j_{m,\infty}$.

The challenge is transitioning from $\mathbf{P}^j_{m,\infty}$ to $\mathbf{P}^j_{m+1,\infty}$. Specifically, to ensure $\mathbf{P}^j_{m+1,1} = \mathbf{P}^j_{m+1,\infty}$, given that $\mathbf{P}^j_{m,N} = \mathbf{P}^j_{m,\infty} \neq \mathbf{P}^j_{m+1,\infty}$, the *variance* of the process noise between the two samples, $\varepsilon_{j,(m+1)N+1}$, has to equal $\mathbf{P}^j_{m+1,\infty} - \exp\left(-2\Delta/l_j\right)\mathbf{P}^j_{m,\infty}$. However, this is infeasible. If $\mathbf{P}^j_{m+1,\infty} < \mathbf{P}^j_{m,\infty}$, the variance is negative. Even if it were positive, the limit as $\Delta \to 0$ does not equal zero, i.e., $\mathbf{P}^j_{m+1,\infty} - \mathbf{P}^j_{m,\infty}$. As a result, the Proposition 1 no longer holds and the trajectory is discontinuous.

In summary, there exists a *trade-off* between maintaining piecewise stationarity and continuity across intervals. PLSO maintains continuity across the intervals while ensuring that the state covariance quickly transitions to the steady-state covariance. We quantify the speed of transition in the following proposition.

Proposition 2. *Assume $l_j \ll N\Delta$, such that $\mathbf{P}^j_{m,N} = \mathbf{P}^j_{m,\infty}$. In Eq. 4.3, the difference between $\mathbf{P}^j_{m,\infty}$ and $\mathbf{P}^j_{m+1,\infty}$ decays exponentially fast as a function of n,*

$$\mathbf{P}^j_{m+1,n} = \mathbf{P}^j_{m+1,\infty} + \exp\left(-\frac{2n\Delta}{l_j}\right)\left(\mathbf{P}^j_{m,\infty} - \mathbf{P}^j_{m+1,\infty}\right).$$

Proof. We prove this result by induction. We first obtain the steady-state covariance $\mathbf{P}^j_{m,\infty}$, similar to **Appendix B**. Since we assume $\mathbf{P}^j_{1,1} = \sigma^2_{j,1}\mathbf{I}_{2\times2}$, we can show that $\forall m, n$, $\mathbf{P}^j_{m,n}$ is a diagonal matrix, noting that $\mathbf{R}(\omega_j)\mathbf{R}^{\mathrm{T}}(\omega_j) = \mathbf{I}_{2\times2}$. Denoting $\mathbf{P}^j_{m,\infty} = \alpha\mathbf{I}_{2\times2}$, we now use the discrete Lyapunov equation

$$\begin{aligned}
\mathbf{P}^j_{m,\infty} &= \exp(-2\Delta/l_j)\mathbf{R}(\omega_j)\mathbf{P}^j_{m,\infty}\mathbf{R}^{\mathrm{T}}(\omega_j) \\
&\quad + \sigma^2_{j,m}\left(1 - \exp\left(-2\Delta/l_j\right)\right)\mathbf{I}_{2\times2} \\
&\Rightarrow \alpha = \exp(-2\Delta/l_j)\alpha + \sigma^2_{j,m}\left(1 - \exp\left(-2\Delta/l_j\right)\right) \\
&\Rightarrow \mathbf{P}^j_{m,\infty} = \sigma^2_{j,m}\mathbf{I}_{2\times2}.
\end{aligned}$$

We now prove the proposition by induction. For fixed j and m, and for $n = 1$,

$$\begin{aligned}\mathbf{P}^j_{m+1,1} &= \exp(-2\Delta/l_j)\mathbf{R}(\omega_j)\mathbf{P}^j_{m,N}\mathbf{R}^{\mathrm{T}}(\omega_j) + \sigma^2_{j,m+1}\left(1 - \exp\left(-2\Delta/l_j\right)\right)\mathbf{I}_{2\times 2}\\ &= \left\{\sigma^2_{j,m+1} + \exp\left(-2\Delta/l_j\right)\left(\sigma^2_{j,m} - \sigma^2_{j,m+1}\right)\right\}\mathbf{I}_{2\times 2}.\end{aligned}$$

Assuming the same holds for $n = n' - 1$, we have for $n = n'$,

$$\begin{aligned}\mathbf{P}^j_{m+1,n'} &= \exp(-2\Delta/l_j)\mathbf{R}(\omega_j)\mathbf{P}^j_{m,n'-1}\mathbf{R}^{\mathrm{T}}(\omega_j) + \sigma^2_{j,m+1}\left(1 - \exp\left(-2\Delta/l_j\right)\right)\mathbf{I}_{2\times 2}\\ &= \exp(-2\Delta/l_j)\sigma^2_{j,m+1}\mathbf{I}_{2\times 2} + \exp\left(-2n'\Delta/l_j\right)\left(\sigma^2_{j,m} - \sigma^2_{j,m+1}\right)\mathbf{I}_{2\times 2}\\ &\qquad + \sigma^2_{j,m+1}\left(1 - \exp\left(-2\Delta/l_j\right)\right)\mathbf{I}_{2\times 2}\\ &= \left\{\sigma^2_{j,m+1} + \exp\left(-2n'\Delta/l_j\right)\left(\sigma^2_{j,m} - \sigma^2_{j,m+1}\right)\right\}\mathbf{I}_{2\times 2}.\end{aligned}$$

By the principle of induction, Eq. 2 holds for $1 \le n \le N$.

$\square$

This implies that, except for the transition portion at the beginning of each window, we can assume stationarity. In practice, we additionally impose an upper bound on l_j during estimation and also use a reasonably-large N. Empirically, we observe that the transition period has little impact.

4.4 Inference

Given the generative model in Eq. 4.3, our goal is to perform inference on the posterior distribution

$$\begin{aligned}&p(\{\mathbf{z}_j\}_j, \{\sigma^2_{j,m}\}_{j,m} \mid \mathbf{y}, \theta)\\ &= \underbrace{p(\{\sigma^2_{j,m}\}_{j,m} \mid \mathbf{y}, \theta)}_{\text{window-level posterior}} \cdot \underbrace{p(\{\mathbf{z}_j\}_j \mid \{\sigma^2_{j,m}\}_{j,m}, \mathbf{y}, \theta)}_{\text{sample-level posterior}}.\end{aligned} \tag{4.5}$$

We can learn θ or fix the parameters to specific values informed by domain knowledge, such as the center frequency or bandwidth of the processes. The posterior distribution factorizes into two terms as in Eq. 4.5, the *window-level* posterior $p(\{\sigma^2_{j,m}\}_{j,m}|\mathbf{y}, \theta)$ and the *sample-level* posterior $p(\{\mathbf{z}_j\}_j|\{\sigma^2_{j,m}\}_{j,m}, \mathbf{y}, \theta)$. Accordingly, we break the inference into two stages.

Stage 1 We minimize the *window-level* negative log-posterior, with respect to θ and $\{\sigma^2_{j,m}\}_{j,m}$. Specifically, we obtain maximum likelihood (ML) estimate $\hat{\theta}_{\mathrm{ML}}$ and maximum a posteriori (MAP) estimate $\{\widehat{\sigma}^2_{j,m,\mathrm{MAP}}\}_{j,m}$. We drop subscripts ML and MAP for notational simplicity.

Stage 2 Given $\{\widehat{\sigma}^2_{j,m}\}_{j,m}$ and $\hat{\theta}$, we perform inference on the *sample-level* posterior. This includes computing the mean $\widehat{\mathbf{z}}_j = \mathbb{E}[\mathbf{z}_j|\{\widehat{\sigma}^2_{j,m}\}_{j,m}, \mathbf{y}, \hat{\theta}]$ and credible intervals, which quantifies the uncertainty of the estimates, or any statistical quantity derived from the posterior distribution.

4.4.1 Optimization of $\{\sigma^2_{j,m}\}_{j,m}$ and θ

We factorize the posterior, $p(\{\sigma^2_{j,m}\}_{j,m} \mid \mathbf{y}, \theta) \propto p(\mathbf{y} \mid \{\sigma^2_{j,m}\}_{j,m}, \theta) \cdot p(\{\sigma^2_{j,m}\}_{j,m} \mid \theta)$. As the exact inference is intractable, we instead minimize the negative log-posterior, $-\log p(\{\sigma^2_{j,m}\}_{j,m} \mid \mathbf{y}, \theta)$. This is an *empirical Bayes* approach [110], since we estimate $\{\sigma^2_{j,m}\}_{j,m}$ using the marginal likelihood $p(\mathbf{y} \mid \{\sigma^2_{j,m}\}_{j,m}, \theta)$. The smooth hyperprior provides the MAP estimate for $\{\sigma^2_{j,m}\}_{j,m}$.

We use the *Whittle likelihood* [111], defined for stationary time-series in the frequency domain, for the log-likelihood $f(\{\sigma^2_{j,m}\}_{j,m}; \theta) = \log p(\mathbf{y} \mid \{\sigma^2_{j,m}\}_{j,m}, \theta)$,

$$f(\{\sigma^2_{j,m}\}_{j,m}; \theta) = -\frac{1}{2} \sum_{m,n=1}^{M,N} \log(\gamma^{(m)}(\omega_n) + \sigma^2_\nu) + \frac{I^{(m)}(\omega_n)}{\gamma^{(m)}(\omega_n) + \sigma^2_\nu}, \tag{4.6}$$

where the log-likelihood is the sum of the Whittle likelihood computed for each interval, with discrete frequency $\omega_n = 2\pi n/N$, and data STFT (periodogram) $I^{(m)}(\omega_n) = \left|\sum_{n'=1}^{N} \exp\left(-2\pi i(n'-1)(n-1)/N\right) \mathbf{y}_{mN+n'}\right|^2$. The Whittle likelihood, which is non-convex, enables frequency-domain parameter estimation as a computationally more efficient alternative to the time domain estimation [112]. The concave log-prior $g(\{\sigma^2_{j,m}\}_{j,m}; \theta) = \log p(\{\sigma^2_{j,m}\}_{j,m} \mid \theta)$, which arises from the continuity on $\{\sigma^2_{j,m}\}_{j,m}$, is given as

$$g(\{\sigma^2_{j,m}\}_{j,m}; \theta) = -\frac{\lambda}{2} \sum_{j=1}^{J} \sum_{m=1}^{M} \left(\log \sigma^2_{j,m} - \log \sigma^2_{j,m-1}\right)^2. \tag{4.7}$$

This yields the following nonconvex problem

$$\min_{\{\sigma_{j,m}^2\}_{j,m},\theta} -\log p\left(\{\sigma_{j,m}^2\}_{j,m} \mid \mathbf{y}, \theta\right) = \min_{\{\sigma_{j,m}^2\}_{j,m},\theta} -f(\{\sigma_{j,m}^2\}_{j,m};\theta) - g(\{\sigma_{j,m}^2\}_{j,m};\theta). \tag{4.8}$$

We optimize Eq. 4.8 by block coordinate descent [113] on $\{\sigma_{j,m}^2\}_{j,m}$ and $\{\sigma_\nu^2, \{l_j\}_j, \{\omega_j\}_j\}$. For σ_ν^2, $\{l_j\}_j$, and $\{\omega_j\}_j$, we minimize $-f(\{\sigma_{j,m}^2\}_{j,m};\theta)$, since $g(\{\sigma_{j,m}^2\}_{j,m};\theta)$ does not affect them. We perform conjugate gradient descent on $\{l_j\}_j$ and $\{\omega_j\}_j$.

Estimation for σ_ν^2

There are two possible ways to estimate the observation noise variance σ_ν^2. The first approach is to perform maximum likelihood estimation of $f(\{\sigma_{j,m}^2\}_{j,m};\theta)$ with respect to σ_ν^2. The second approach, which we found to work *better* in practice and use throughout the manuscript, is to directly estimate it from the Fourier transform of the data. Given a cutoff frequency ω_c, informed by domain knowledge, we take the average power of the Fourier transform of $\mathbf{y}$ in $[\omega_c, f_s/2]$. For instance, it is widely known that the spectral content below 40 Hz in anesthesia EEG dataset is physiologically relevant and we use $\omega_c \simeq 40$ Hz.

Optimization of $\{\sigma_{j,m}^2\}_{j,m}$

We introduce an algorithm to compute a local optimal solution of $\{\sigma_{j,m}^2\}_{j,m}$ for the nonconvex optimization problem in Eq. 4.8, by leveraging the regularized temporal structure of $\{\sigma_{j,m}^2\}_{j,m}$. It extends the inexact accelerated proximal gradient (APG) method [100], by solving the proximal step with a Kalman filter/smoother [99]. This follows since computing the proximal operator for $g(\{\sigma_{j,m}^2\}_{j,m};\theta)$ is equivalent to MAP estimation for J independent 1-dimensional linear Gaussian state-space models

$$\begin{aligned}\{\log \sigma_{j,m}^{(l+1),2}\}_{j,m} &= \mathrm{prox}_{-\alpha^{(l)}g}(\mathbf{v}^{(l)}) \\ &= \arg\min_{\{\sigma_{j,m}^2\}_{j,m}} \underbrace{\frac{\sum_{j,m}^{J,M}(\mathbf{v}_{j,m}^{(l)} - \log \sigma_{j,m}^2)^2}{2\alpha^{(l)}} - g(\{\sigma_{j,m}^2\}_{j,m})}_{\sum_{j=1}^{J} q_j}\end{aligned} \tag{4.9}$$

where $\alpha^{(l)} > 0$ is a step-size for the l^{th} iteration, $q_j = \sum_{m=1}^{M} \frac{(\mathbf{v}_{j,m}^{(l)} - \log \sigma_{j,m}^2)^2}{2\alpha^{(l)}} + \frac{\lambda}{2}(\log \sigma_{j,m}^2 - \log \sigma_{j,m-1}^2)^2$, and $\mathbf{v}_{j,m}^{(l)} = \log \sigma_{j,m}^{(l),2} + \alpha^{(l)} \frac{\partial f(\{\sigma_{j,m}^2\}_{j,m})}{\partial \log \sigma_{j,m}^2}\Big|_{\{\sigma_{j,m}^{(l),2}\}_{j,m}}$.

The j^{th} optimization problem, $\min_{\{\sigma_{j,m}^2\}_m} q_j$, is equivalent to estimating the mean of the posterior for $\{\log \sigma_{j,m}^2\}_m$ in a linear Gaussian state-space model with observations $\{\mathbf{v}_{j,m}^{(l)}\}_m$, observation noise variance $\alpha^{(l)}$, and state variance $1/\lambda$. Therefore, the solution can efficiently be computed with J 1-dimensional, Kalman filters/smoothers, with the computational complexity of $O(JM)$.

Note that Eq. 4.9 holds for all non-negative λ. If $\lambda = 0$, the proximal operator is an identity operator, as $\log \sigma_{j,m}^{(l+1),2} = \mathbf{v}_{j,m}^{(l)}$. This is a gradient descent with a step-size rule. If $\lambda \to \infty$, we have $\log \sigma_{j,m}^2 = \log \sigma_{j,m-1}^2$, $\forall m$, which leads to $\log \sigma_{j,m}^{(l+1),2} = (1/M) \sum_{m=1}^{M} \mathbf{v}_{j,m}^{(l)}$. The algorithm is guaranteed to converge when $\alpha^{(l)} < 1/C$, where C is the Lipschitz constant for $f(\{\sigma_{j,m}^2\}_{j,m}; \theta)$. In practice, we select $\alpha^{(l)}$ according to the step-size rule [114]. In **Appendix F & G**, we present the full algorithm for optimizing $\{\sigma_{j,m}^2\}_{j,m}$ and a derivation for C.

4.4.2 Inference for $p(\{\mathbf{z}_j\}_j \mid \{\sigma_{j,m}^2\}_{j,m}, \mathbf{y}, \theta)$

We perform inference on the posterior distribution $p(\{\mathbf{z}_j\}_j \mid \{\widehat{\sigma}_{j,m}^2\}_{j,m}, \mathbf{y}, \hat{\theta})$. Since this is a Gaussian distribution, the mean trajectories $\{\widehat{\mathbf{z}}_j\}_j$ and the credible intervals can be computed analytically. Moreover, Eq. 4.3 is a linear Gaussian state-space model, we can use Kalman filter/smoother for efficient computation with computational complexity $O(J^2K)$, further discussed In **Appendix H**. Since we use the point estimate $\{\widehat{\sigma}_{j,m}^2\}_{j,m}$, the credible interval for $\{\widehat{\mathbf{z}}_j\}_j$ will be *narrower* compared to the full Bayesian setting which accounts for all values of $\{\sigma_{j,m}^2\}_{j,m}$.

Monte Carlo Inference

We can also obtain posterior samples and perform Monte Carlo (MC) inference on any posterior-derived quantity. To generate the MC trajectory samples, we use the forward-filter backward-sampling (FFBS) algorithm [115]. Assuming S number of MC samples, the computational complexity for FFBS is $O(SJ^2K)$, since for each sample, the algorithm uses Kalman filter/smoother for sampling. This is different from generating samples with the interval-specific posterior in Eq. 4.4. In the latter case, the FFBS algorithm is run M times, the samples of which have to be concatenated to form an entire trajectory. With PLSO, the trajectory sample is conditioned on the entire observation and is continuous across the intervals.

One quantity of interest is the *phase*. We obtain the phase as $\phi_{j,k} = \tan^{-1}(\mathbf{z}^{\Im}_{j,k}/\mathbf{z}^{\Re}_{j,k})$. Since $\tan^{-1}(\cdot)$ is a non-linear operation, we compute the mean and credible interval with MC samples through the FFBS algorithm. Given the posterior-sampled trajectories $\{\mathbf{z}^{(s)}_j\}_{j,s}$, where $s \in \{1, \dots, S\}$ denotes MC sample index, we estimate $\widehat{\phi}_{j,k} = (1/S)\sum_{s=1}^{S} \tan^{-1}(\mathbf{z}^{\Im,(s)}_{j,k}/\mathbf{z}^{\Re,(s)}_{j,k})$, and use empirical quantiles for the associated credible interval.

4.4.3 Choice of J and λ

We choose J that minimizes the Akaike Information Criterion (AIC) [116], defined as

$$\mathrm{AIC}(J) = -(2/M) \cdot \log p(\mathbf{y} \mid \{\widehat{\sigma}^2_{j,m}\}_{j,m}, \hat{\theta}) + 2 \cdot 3 \cdot J, \tag{4.10}$$

where $3 \cdot J$ corresponds to the number of parameters $(\{l_j\}_j, \{\omega_j\}_j, \{\sigma^2_{j,m}\}_j)$. The regularization parameter λ is determined through a two-fold cross-validation, where each fold is generated by aggregating even/odd sample indices [18].

4.4.4 Choice of window length N

The choice of window length N presents the tradeoff between 1) spectral resolution and 2) the temporal resolution of the spectral dynamics [89]. For a shorter window, the estimated spectral dynamics have a *finer* temporal resolution, *coarser* spectral resolution, and *higher* variance. For a longer window, these trends are reversed. This suggests that the choice is application-dependent. For electrophysiology data, a window on the order of seconds is used, as scientific interpretations are made on the basis of fine spectral resolution (< 1Hz). For audio signal processing [117], short windows ($10 \sim 100$ ms) are used, since audio data is highly nonstationary and thus requires fine temporal resolution for processing. A survey of window lengths used in different applications can be found in the **supporting information** of [19].

4.4.5 Initialization

For a given number of components J, we first construct the spectrogram of the data using STFT and identify the frequency bands with prominent power, i.e., frequency bands whose average power exceeds pre-determined threshold. The center frequencies of these bands serve as the initial center frequencies $\{\omega_j^{\text{init}}\}_j$, which are either fixed throughout the algorithm or further refined through the estimation algorithm in the main text. If J exceeds the number of identified frequency bands from the spectrogram, 1) we first place $\{\omega_j^{\text{init}}\}_j$ in the prominent frequency bands and 2) we then place the remaining components uniformly spread out in $[0, \omega_c]$, where ω_c is a cutoff frequency to be further determined in the next section. As for $\{l_j^{\text{init}}\}_j$, we set it to be a certain fraction of the corresponding $\{\omega_j^{\text{init}}\}_j$. We then fit $\{\sigma_{j,m}^2\}_{j,m}$ and θ with $\lambda = 0$, through the procedure explained in Stage 1. We finally use these estimates as initial values for other values of λ.

4.5 Related works

We examine how PLSO relates to other nonstationary frameworks.

STFT/Regularized STFT In STFT, the harmonic basis is used, whereas quasi-periodic components are used for PLSO, which allows capturing of broader spectral content. Recent works regularize STFT coefficients with stochastic continuity across the windows, to infer smooth spectral dynamics [18, 19]. However, this regularization leads to discontinuities at window boundaries.

Piecewise stationary GP GP regression and parameter estimation are performed within each interval [118, 119]. Consequently, the recovered trajectories are discontinuous. Also, the inversion of covariance matrix leads to an expensive inference. For example, the time-complexity of mean trajectory estimation is $O(MN^3) = O(N^2K)$, whereas the time-complexity for PLSO is $O(J^2K)$. Considering that the typical sampling frequency for electrophysiology data is $\sim 10^3$ (Hz) and windows are several seconds, which leads to $N \sim 10^3$, PLSO is computationally more efficient. In the **Appendix I**, we confirm this through an experiment.

Time-varying Autoregressive model (TVAR) The TVAR model [120] is given as

$$\mathbf{y}_k = \sum_{p=1}^{P} a_{p,k}\mathbf{y}_{k-p} + \varepsilon_k,$$

with the time-varying coefficients $\{a_{p,k}\}_p$. Consequently, it does not suffer from discontinuity issues. TVAR can also be *approximately* decomposed into oscillatory components via eigen-decomposition of the transition matrix [121]. However, since the eigen-decomposition changes at every sample, this leads to an ambiguous interpretation of the oscillations in the data, as we discuss in Section 6.5.

RNN frameworks Despite the popularity of recurrent neural networks (RNN) for time-series applications [122], we believe PLSO is more appropriate for time-frequency analysis for two reasons.

1. RNNs operate in the time-domain with the goal of prediction/denoising and consequently less emphasis is placed on local stationarity or estimation of second-order statistics. Performing time-frequency analysis requires segmenting the RNN outputs and applying the STFT, which yields noisy spectral estimates.

2. RNN is not a generative framework. Although variational framework can be combined with RNN [123, 55], the use of variational lower bound objective could lead to suboptimal results. On the other hand, PLSO is a generative framework that maximizes the true log-posterior.

4.6 Experiments

We apply PLSO to three settings: 1) A simulated dataset 2) local-field potential (LFP) data from the rat hippocampus, and 3) EEG data from a subject under anesthesia.

We use PLSO with $\lambda = 0$, $\lambda \to \infty$, and λ determined by cross-validation, λ_{CV}. We use interval lengths chosen by domain experts. As baselines, we use 1) regularized STFT (STFT-reg.) and 2) Piecewise stationary GP (GP-PS). For GP-PS, we use the same $\{\widehat{\sigma}^2_{j,m}\}_{j,m}$ and $\widehat{\theta}$ as PLSO with $\lambda = 0$, so that the estimated PSD of GP-PS and PLSO are identical. This lets us explain differences in time-domain estimates by the fact that PLSO operates in the time-domain.

4.6.1 Simulated dataset

We simulate from the following model for $1 \leq k \leq K$

$$\mathbf{y}_k = 10\left(\frac{K-k}{K}\right)\mathbf{z}^{\Re}_{1,k} + 10\cos^4(2\pi\omega_0 k)\mathbf{z}^{\Re}_{2,k} + \nu_k,$$

where $\mathbf{z}_{1,k}$ and $\mathbf{z}_{2,k}$ are as in Eq. 4.2, with $(\omega_0, \omega_1, \omega_2) = (0.04, 1, 10)$ Hz, $f_s = 200$ Hz, $T = 100$ seconds, $l_1 = l_2 = 1$, and $\nu_k \sim \mathcal{N}(0, 25)$. This stationary process comprises two amplitude-modulated oscillations, namely one modulated by a slow-frequency sinusoid and the other a linearly-increasing signal [18]. We simulate 20 realizations and train on each realization, assuming 2-second PS intervals. For PLSO, we use $J = 2$. Additional details are provided in the **Appendix J**.

Results We use two metrics: 1) Mean squared error (MSE) between the mean estimate $\hat{\mathbf{z}}_j$ and the ground truth $\mathbf{z}^{\text{True}}_j$ and 2) jump($\mathbf{z}_j$). The averaged results are shown in Table 4.1. We define $\text{jump}(\mathbf{z}_j) = \frac{1}{M-1}\sum_{m=1}^{M-1}|\hat{\mathbf{z}}_{j,mN+1} - \hat{\mathbf{z}}_{j,mN}|$ to be the level of discontinuity at the interval boundaries. If jump($\mathbf{z}_j$) greatly exceeds jump($\mathbf{z}^{\text{True}}_j$), this implies the existence of large discontinuities at the boundaries.

Fig. 6-2 shows the true data in the time domain and spectrogram results. Fig. 6-2(c) shows that although the regularized STFT detects activities around 1 and 10 Hz, it fails to delineate the time-varying spectral pattern. Fig. 6-2(d) shows that PLSO with stationarity ($\lambda \to \infty$) assumption is too restrictive. Fig. 6-2(e), (f) show that both PLSO with independent window assumption ($\lambda = 0$) and PLSO with cross-validated λ ($\lambda = \lambda_{\text{CV}}$) are able to capture the dynamic pattern, with the latter being more effective in recovering the smooth dynamics across different PS intervals.

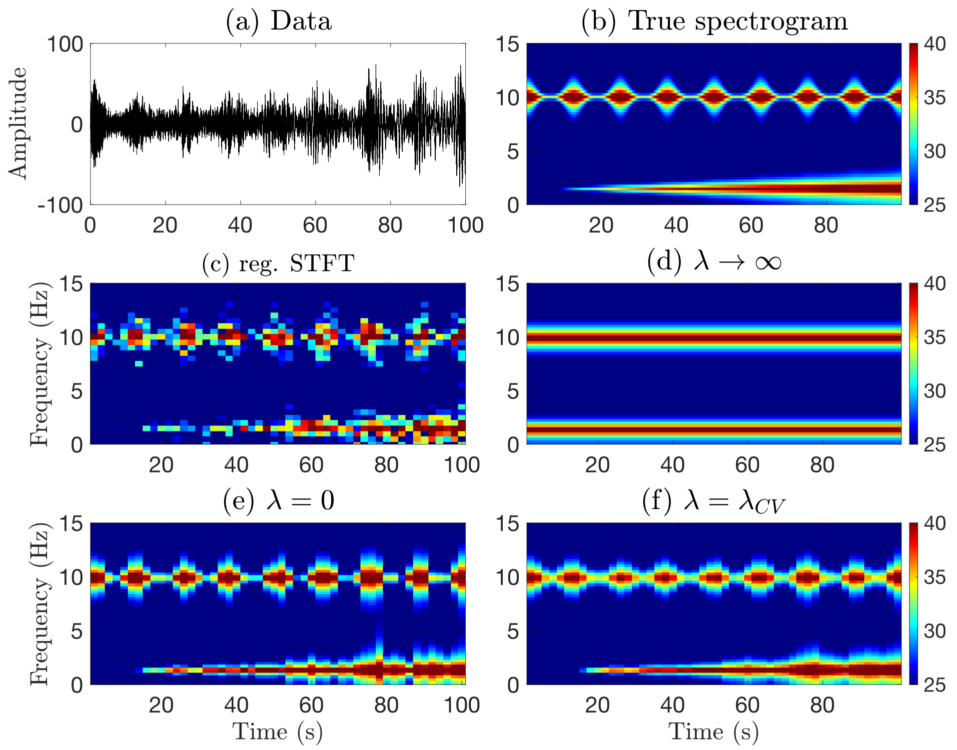

Figure 4-3: Spectrograms for simulation (in dB). (a) True data (b) True spectrogram (c) regularized STFT (d) PLSO with $\lambda \to \infty$ (e) PLSO with $\lambda = 0$ (f) PLSO with $\lambda = \lambda_{\text{CV}}$.

Table 4.1: Simulation results. For jump($\mathbf{z}_j$) and MSE, left/right metrics correspond to $\mathbf{z}_1/\mathbf{z}_2$, respectively.

	jump($\mathbf{z}_j$)	MSE	IS div.
Truth	0.95/12.11	0/0	0
$\lambda = 0$	0.26/10.15	2.90/3.92	4.08
$\lambda \to \infty$	0.22/10.32	3.26/4.53	13.78
$\lambda = \lambda_{\text{CV}}$	0.25/10.21	**2.88/3.91**	**3.93**
STFT-reg.	49.59/81.00	6.89/10.68	N/A
GP-PS	16.99/23.28	3.00/4.04	4.08

For GP-PS and STFT-reg., jump($\mathbf{z}_j$) exceeds jump($\mathbf{z}_j^{\text{True}}$), indicating discontinuities at the boundaries. An example is in Fig. 4-1. PLSO produces a similar jump metric as the ground truth metric, indicating the absence of discontinuities. We attribute the lower value to Kalman smoothing. For the TF domain, we use Itakura-Saito (IS) divergence [88] as a distance measure between the ground truth spectra and the PLSO

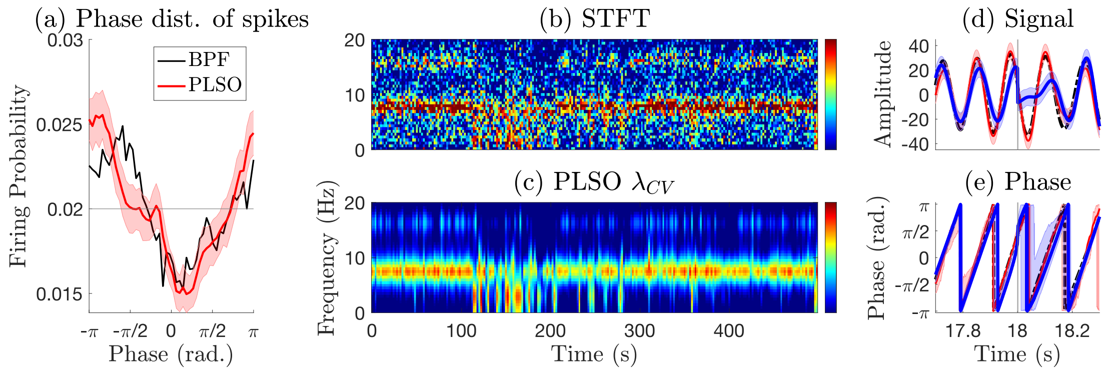

Figure 4-4: Result of analyses of hippocampal data. (a) Theta phase distribution of population neuron spikes, computed with bandpass-filtered LFP (black), PLSO estimate of $\hat{\mathbf{z}}_2$ with credible intervals estimated from 200 posterior samples (red). Horizontal gray line indicates the uniform distribution. (b-c) Spectrogram (in dB) for 500 seconds (b) STFT (c) PLSO with λ_{CV}. Learned frequencies are $(\widehat{\omega}_1, \widehat{\omega}_2, \widehat{\omega}_3) = (2.99, 7.62, 15.92)$ Hz, with $\widehat{\omega}_4 \sim \widehat{\omega}_5 > 25$ Hz. (d-e) Time-domain results. (d) Reconstructed signal (e) phase for $\hat{\mathbf{z}}_2$ and interval boundary (vertical gray), with bandpass-filtered data (dotted black), STFT-reg. (blue), and PLSO (red). Shaded area represents 95% credible interval from $S = 200$ sample trajectories.

estimates. That the highest divergence is given by $\lambda \to \infty$ indicates the inaccuracy of the stationarity assumption.

4.6.2 LFP data from the rat hippocampus

We use LFP data collected from the rat hippocampus during open field tasks [124], with $T = 1,600$ seconds and $f_s = 1,250$ Hz[2]. The theta neural oscillation band ($5 \sim 10$ Hz) is believed to play a role in coordinating the firing of neurons in the entorhinal-hippocampal system and is important for understanding the local circuit computation.

We fit PLSO with $J = 5$, which minimizes AIC as shown in Table 4.2, with 2-second PS interval. The estimated $\widehat{\omega}_2$ is 7.62 Hz in the theta band. To obtain the phase for non-PLSO methods, we perform the Hilbert transform on the theta-band reconstructed signal. With no ground truth, we bandpass-filter (BPF) the data in the theta band for reference.

[2]We use channel 1 of mouse ec013.528 for the LFP. The population spikes were simultaneously recorded.

Table 4.2: AIC as a function of J for Hippocampus data

J	1	2	3	4	**5**	6
AIC	2882	2593	2566	2522	**2503**	2505

Spike-phase coupling Fig. 4-4(a) shows the theta phase distribution of population neuron spikes in the hippocampus. The PLSO-estimated distribution (red) confirms the original results analyzed with bandpass-filtered signal (black) [124]–the hippocampal spikes show a strong preference for a specific phase, π for this dataset, of the theta band. Since PLSO provides posterior sample-trajectories for the entire time-series, we can compute as many realizations of the phase distribution as the number of MC samples. The resulting credible interval excludes the uniform distribution (horizontal gray), suggesting the statistical significance of strong phase preference.

Denoised spectrogram Fig. 4-4(b-c) shows the estimated spectrogram. We observe that PLSO denoises the spectrogram, while retaining sustained power at $\widehat{\omega}_2 = 7.62$ Hz and weaker bursts at $(\widehat{\omega}_1, \widehat{\omega}_3) = (2.99, 15.92)$ Hz.

Time domain discontinuity Fig. 4-4(d-e) show a segment of the estimated signal and phase near a boundary for $\widehat{\omega}_2$. While the estimates from STFT-reg. (blue) and PLSO (red) follow the BPF result closely, the STFT-reg. estimates exhibit discontiunity/distortion near the boundary. In Fig. 4-4(e), the phase jump at the boundary is 38.4 degrees. We also computed jump(ϕ_2) in degrees/sample. Considering that the theta band roughly progresses 2.16 ($= 7.5(\text{Hz}) \times 360/1250\,(\text{Hz})$) degrees/sample, we observe that BPF (2.23), as expected, and PLSO (λ_{CV} : 2.40, $\lambda \to \infty$: 2.66) are not affected by the boundary effect. This is not the case for STFT-reg. (26.83) and GP-PS (25.91).

Comparison with TVAR Fig. 4-5(a-b) shows a segment of TVAR inference results. Specifically, Fig. 4-5(a) and (b) shows a time-varying center frequency $\widehat{\omega}_1$ and the corresponding reconstruction, for the lowest frequency component. Note that the eigenvalues, which correspond to $\{\omega_j\}_j$, and the eigenvectors, which are used for oscillatory decomposition, are derived from the *estimated* TVAR transition

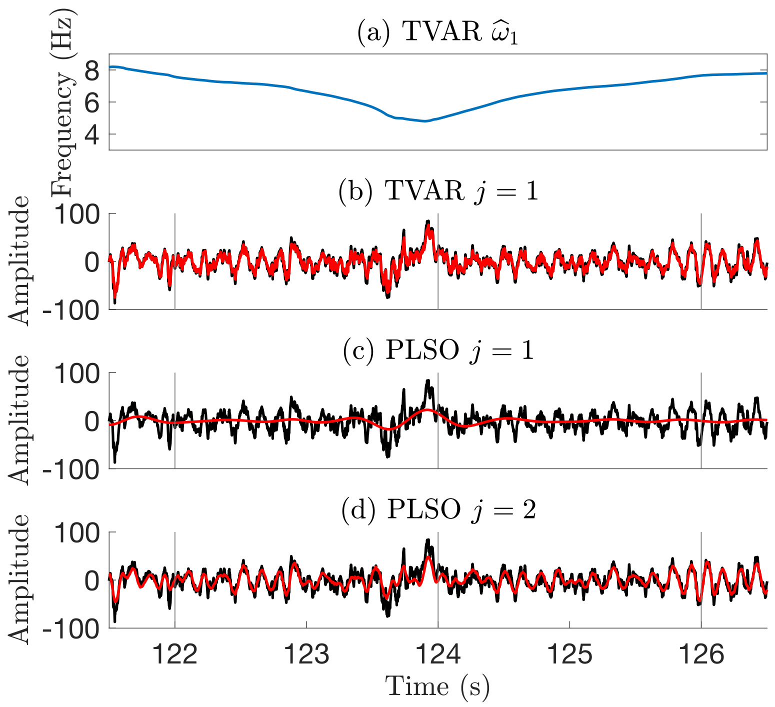

Figure 4-5: Hippocampus data. (a) time-varying $\widehat{\omega}_1$ for TVAR. (b-d) Inferred mean trajectory (red) for (b) TVAR $j = 1$, (c) PLSO $j = 1$, and (d) PLSO $j = 2$, with raw data (black).

matrix. Consequently, we cannot explicitly control $\{\omega_j\}_j$, as shown in Fig. 4-5(a), the bandwidth of each component, as well as the number of components J. This is further complicated by the fact that the transition matrix changes every sample. Fig. 4-5(b) shows that this ambiguity results in the lowest-frequency component of TVAR explaining *both* the slow ($0.1 \sim 2$ Hz) and theta components. With PLSO, on the contrary, we can explicitly specify or learn the parameters. Fig. 4-5(c-d) demonstrates that PLSO is able to delineate the slow/theta components without any discontinuity.

4.6.3 EEG data from the human brain under propofol anesthesia

We apply PLSO to the EEG data from a subject anesthetized with propofol anesthetic drug, to assess whether PLSO can leverage regularization to recover smooth spectral dynamics, which is widely-observed during propofol-induced unconsciousness[3] [26]. The data last $T = 2{,}300$ seconds, sampled at $f_s = 250$ Hz. The drug infusion starts at $t = 0$ and the subject loses consciousness around $t = 260$ seconds. We use $J = 6$ and assume a 4-second PS interval.

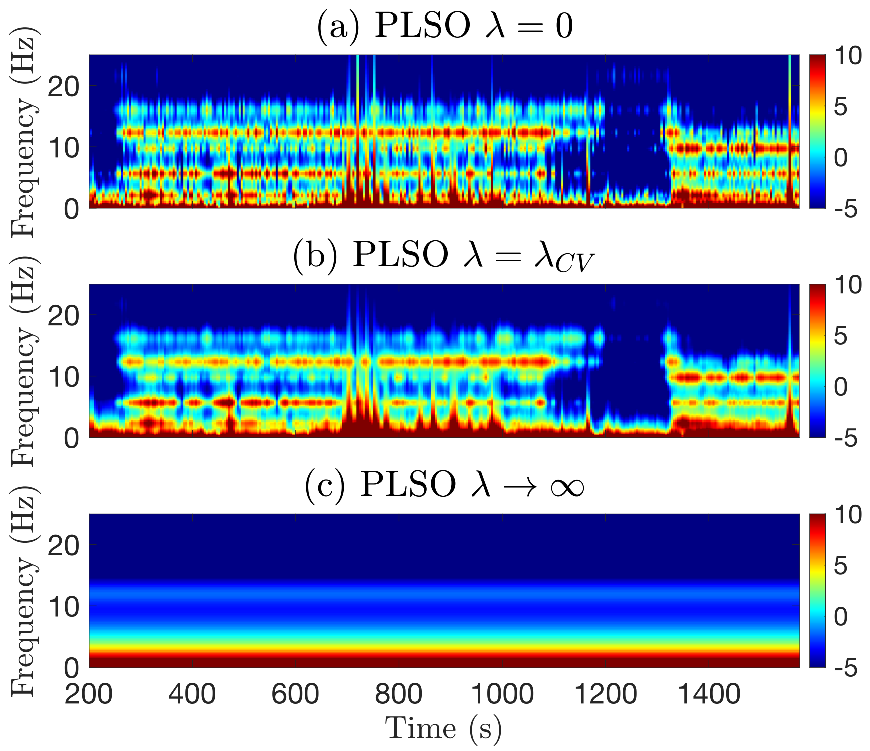

Figure 4-6: Spectrogram (in dB) under propofol anesthesia. PLSO with (a) $\lambda = 0$ (b) $\lambda = \lambda_{\mathrm{CV}}$ (c) $\lambda \to \infty$.

Smooth spectrogram Fig. 4-6(a-b) shows a segment of the PLSO-estimated spectrogram with $\lambda = 0$ and $\lambda = \lambda_{\mathrm{CV}}$. They identify strong slow ($0.1 \sim 2$ Hz) and alpha oscillations ($8 \sim 15$ Hz), both well-known signatures of propofol-induced unconsciousness. We also observe that the alpha band power diminishes between 1,200 and 1,350 seconds, suggesting that the subject regained consciousness before becoming unconscious again. PLSO with $\lambda = 0$ exhibits PSD fluctuation across windows, since $\{\sigma^2_{j,m}\}_{j,m}$ are estimated independently. The stationary PLSO ($\lambda \to \infty$) is restrictive and fails to capture spectral dynamics (Fig. 4-6(c)). In contrast, PLSO with λ_{CV} exhibits smooth dynamics by pooling together estimates from the neighboring windows.

[3]The EEG recording is part of de-identified data collected from patients at Massachusetts General Hospital (MGH) as a part of a MGH Human Research Committee-approved protocol.

The regularization also helps remove movement-related artifacts, shown as vertical lines in Fig. 4-6(a), around 700 $\sim$ 800/1,200 seconds, and spurious power in 20 $\sim$ 25 Hz band. In summary, PLSO with regularization enables smooth spectral dynamics estimation and spurious noise removal.

Fig. 4-7 is another example of PLSO in action for different propofol anesthesic EEG data. Similar to the previous example, we observe the benefits of applying PLSO to the nonstationary timeseries data.

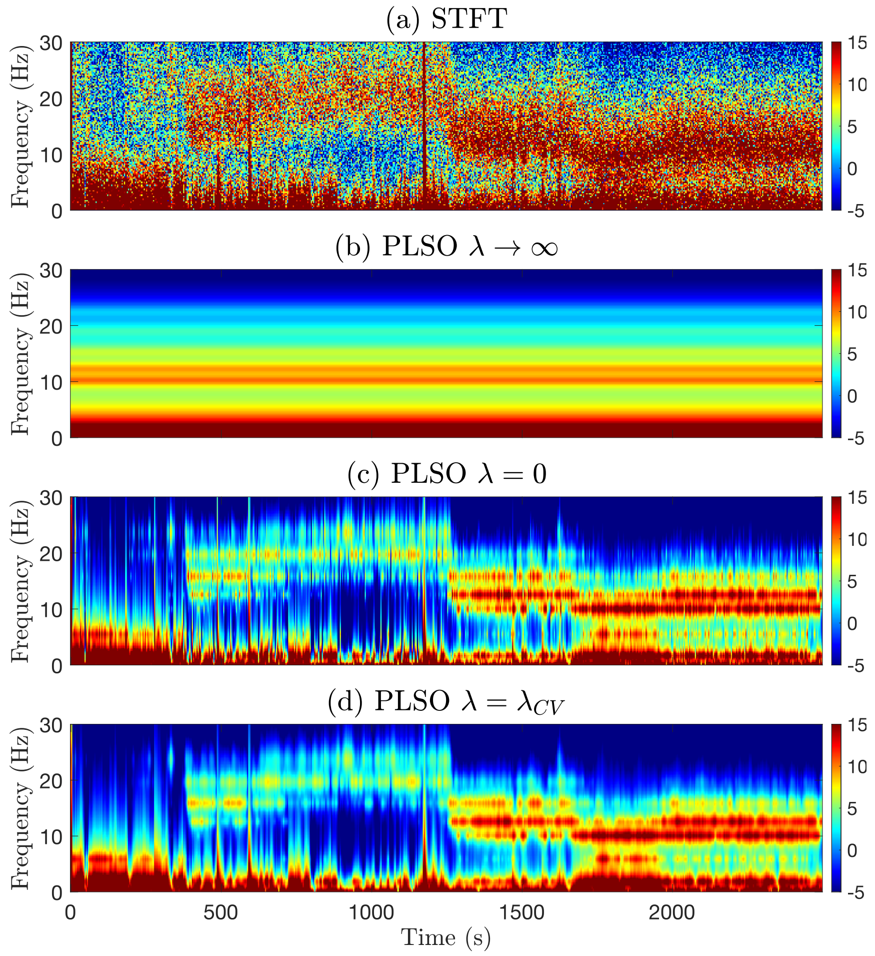

Figure 4-7: Spectrogram (in dB) under propofol anesthesia. (a) STFT of the data (b) PLSO with $\lambda \to \infty$ (c) PLSO with $\lambda = 0$ (d) PLSO with $\lambda = \lambda_{\text{CV}}$.

4.7 Conclusion

We presented the Piecewise Locally Stationary Oscillatory (PLSO) framework to model nonstationary time-series data with slowly time-varying spectra, as the superposition of piecewise stationary (PS) oscillatory components. PLSO strikes a balance between stochastic continuity of the data across PS intervals and stationarity within each interval. For inference, we introduce an algorithm that combines Kalman theory and nonconvex optimization algorithms. Applications to simulated/real data show that PLSO preserves time-domain continuity and captures time-varying spectra. Future directions include 1) the automatic identification of PS intervals and 2) the expansion to higher-order autoregressive models and diverse priors on the parameters that enforce continuity across intervals.

Chapter 5

Convolutional Dictionary learning with grid refinement

Primer We now turn our attention to a completely different topic from the time-frequency analysis. We study how we can incorporate the smoothness of the dictionary elements into the convolutional dictionary learning (CDL) framework. This chapter is adapted from the following work

- Andrew H. Song, Francisco Flores, and Demba Ba, *Convolutional Dictionary Leraning with Grid Refinement*, IEEE Transactions on Signal Processing, 2020

5.1 Introduction

In recent years, the problem of decomposing an observed signal into a sparse linear combination of elements drawn from a known dictionary, often referred to as sparse approximation [125], has been of great interest to the signal processing community. Specifically, representing the signal as the superposition of shifted (or shift-invariant) templates with local support has received special attention [41]. This is due to the observation that many examples of real-world signals can be modeled in this manner. For instance, signals arising from electrophysiological recordings of neural activity can be modeled as the sum of distinct action potentials produced by the neurons near recording electrodes [126]. In studies involving electroencephalography (EEG), there is growing evidence that the signal should be studied in terms of the aggregation of transient events with specific templates [127]. In microscopy imaging of single molecules, an image of photoactivated single molecules can be modeled as a point spread function (PSF) placed at each molecule's location [128].

A generative model for these signals is the convolution, on a continuous domain, between templates, a collection of which is referred to as *dictionary*, and a set of scaled and shifted delta functions, referred to as *codes*. For one-dimensional signals, the amplitude and the location of each code correspond to the magnitude and the time when an event occurs, respectively. Given an observed signal, the goal of *Convolutional Dictionary Learning* (CDL) frameworks [129] is to estimate the templates and the codes under the generative model, with sparsity constraints on the codes. These typically alternate between two steps, a *Convolutional Sparse Coding* (CSC) step to estimate the codes, and a *Convolutional Dictionary Update* (CDU) step to estimate the dictionary.

One of the drawbacks of existing CDL frameworks is the assumption that the domain of the signal of interest is discrete when, in fact, the underlying signal occurs on a continuous domain. For one-dimensional signals, the discrete approximation of the generative model introduces errors known as time-quantization errors [130]. Specifically, if an event in the continuous-time model were to occur at a time that does not coincide with any point on the discrete-time sampling grid, the CSC step would inaccurately identify the event as occurring at a time on the grid. Consequently, the dictionary estimate would be inaccurate in the CDU step.

Recently, sampling-grid-free methods have been introduced to address the sensitivity of CSC to the discrete approximation of the continuous generative model. Continuous Basis Pursuit (CBP) [131], a convex sparse regression framework, uses as new set of templates, derived from the original dictionary, to approximate the subspace of continuous-time-shifted copies of the original templates. With this new dictionary, it solves an ℓ_1-regularized convex regression problem, which yields continuous estimates of the codes. Another line of works employs a greedy continuous framework, based on the Frank-Wolfe algorithm (conditional gradient method) [132], [133], [134], to estimate continuous codes. This method first greedily searches for likely neighborhoods of the codes, and then performs local optimization within these to identify accurate off-the-grid code locations. Despite both families of methods being more accurate compared to approaches that operate on the sampling grid, they do not scale well with the size of modern datasets. More importantly, they do not include a CDU step, and thus cannot be considered CDL frameworks.

To address the aforementioned drawbacks, we propose a CDL framework that 1) is scalable and efficient and 2) estimates a dictionary in a manner that accounts for events occurring off the discrete sampling grid. The main principle behind our framework is grid refinement: different from the above-mentioned grid-free methods, all our operations occur on a refined grid several times finer than the original grid. Specifically, we use smoothly-interpolated versions of the templates on the original grid to obtain templates on the refined grid. Performing CSC and CDU on the fine-resolution grid is more accurate than on the original grid. Grid refinement has two additional advantages. First, it allows us to extend CSC approaches based on greedy methods, which are known to be less computationally demanding than basis pursuit and ℓ_1-regularized methods [135, 136]. An efficient greedy implementation significantly offsets the increase in computational cost due to the refined sampling grid. Second, it allows us to learn the dictionary while incorporating the amplitudes and the times of events that occur off the grid.
Our contributions are the following:

- **A dictionary update framework that handles non-integer delays** For the first time, we introduce a CDU algorithm that accounts for estimates of the sparse codes from the CSC step that correspond to events off the grid. When compared to conventional dictionary update algorithms, the templates learned from our approach are more accurate (Figs. 5-5, 5-9).

- **A fast convolutional greedy pursuit algorithm** We propose an efficient algorithm for convolutional greedy pursuit under a discrete generative model. When the events from the continuous model occur on the sampling grid, we term this algorithm Convolutional Orthogonal Matching Pursuit (COMP). COMP is much faster than grid-free methods (Table 5.4). LocOMP [137] is a related algorithm that differs from ours in one of the steps.

- **A CSC framework that handles non-integer delays** We introduce a discrete generative model that accounts for events that do not occur on the original sampling grid, but rather on a finer grid, with tunable precision. This model is inspired by the concept of smooth interpolation in digital signal processing. To perform CSC, we extend COMP and call the resulting algorithm COMP-INTERP. Compared to the conventional grid-based CSC frameworks, COMP-INTERP is more accurate in identifying the locations where off-the-grid events occur and achieves an accuracy similar to grid-free methods (Fig. 5-4).

The rest of our treatment begins in Section 5.2, where we introduce the generative model and the CDL objective functions. In Section 6.2, we review existing work. In Section 5.4, we introduce COMP-INTERP, an efficient CSC algorithm that accounts for events that occur off the grid. In Section 5.5, we introduce a CDU step that can handle events off the grid. We use simulated and real datasets to show the performance of our algorithms in Section 6.5. We conclude in Section 6.6.

5.2 Generative Model and Problem setup

The framework we propose for CDL with grid refinement applies to signals with an arbitrary *finite-dimensional* domain. Our mathematical exposition focuses on signals with a one-dimensional domain, namely time, with the understanding that a generalization to signals whose domain is of higher dimension is simply a matter of replacing one-dimensional convolutions with multi-dimensional ones. We assume that the shift-invariant templates occur fully *within* the domain of the signal, i.e., we do not consider border scenarios.

5.2.1 Notation

Table 5.1 summarizes our notation. We introduce additional notation as necessary, at the beginning of the section that uses it first. We use the expressions *event off the grid* and *event with a non-integer delay* interchangeably. We define convolution, $*$, and cross-correlation, $\star$, between $\mathbf{h}$ and $\mathbf{y}$ as follows

$$\begin{aligned}(\mathbf{h} * \mathbf{y})[n] &= \sum_m \mathbf{h}[m]\mathbf{y}[n-m],\\ (\mathbf{h} \star \mathbf{y})[n] &= \sum_m \mathbf{h}[m]\mathbf{y}[n+m].\end{aligned} \tag{5.1}$$

We treat $\mathbf{h}$ and $\mathbf{y}$ as discrete-time signals–instead of vectors–which can adopt zero or negative indices.

Symbol	Description
$\mathbf{H}$	Matrix
$\mathbf{h}$	Vector
$\mathcal{S}$	Set
$\mathbf{H}_i$	i^{th} column from $\mathbf{H}$
$\mathbf{H}^c$	c^{th} block from $\mathbf{H}$
$\mathbf{h}[j]$	j^{th} entry from $\mathbf{h}$
$\mathcal{S}_i$	i^{th} element from set $\mathcal{S}$
$\mathcal{S}^j$	j^{th} set
$\mathbf{I}_{L\times L}$	Identity matrix of size $L \times L$
$\mathbf{r}^{(t)}$	$\mathbf{r}$ at t^{th} iteration
$\mathbf{0}_L$	a length-L vector with all entries equal to 0
$n^c_{j,i}$	Location of i^{th} event from source c in j^{th} window
N^c_j	Number of events from source c in j^{th} window
$\lVert\cdot\rVert_p$	ℓ_p norm

Table 5.1: Notational conventions.

5.2.2 Continuous and discrete-time generative models

Let $y(t)$ be a continuous-time signal observed in the interval $(0, T]$ and $\{h_c(t)\}_c$ be templates (filters) from C sources. We assume that the templates each have the same length and are localized in time. The shift-invariant continuous generative model expresses $y(t)$ as

$$y(t) = \sum_{c=1}^{C} \sum_{i=1}^{N^c} x_i^c h_c(t - \tau_i^c) + \varepsilon(t), \tag{5.2}$$

where N^c denotes the number of events from source c, τ_i^c and x_i^c denote the position and the amplitude of the i^{th} event from source c, respectively. The variable $\varepsilon(t)$ denotes i.i.d. white noise.

To formulate a discrete-time analogue of Eq. 5.2, let Δ denote the length of a sampling interval and $f_s = \frac{1}{\Delta}$ the associated sampling frequency. The number of intervals of size Δ in $(0, T]$ is $N = \lfloor \frac{T}{\Delta} \rfloor$. Further let $n = 1, \cdots, N \in \mathbb{N}^+$ be the discrete-time index, and n_i^c denote the discrete-time approximation of τ_i^c, such that $n_i^c \Delta \leq \tau_i^c < (n_i^c + 1)\Delta$. Finally, we denote by $\mathbf{h}_c \in \mathbb{R}^L$ the discrete time analogue of $h_c(t)$, which we assume is normalized such that $\|\mathbf{h}_c\|_2 = 1, \forall c$. Using this notation, we can obtain discrete-time samples $\mathbf{y}[n] = y(n\Delta)$ of $y(t)$ that satisfy

$$\mathbf{y}[n] = \sum_{c=1}^{C} \Big(\mathbf{x}^c * \mathbf{h}_c \Big)[n] + \boldsymbol{\varepsilon}[n], \tag{5.3}$$

where $\mathbf{x}^c[n] = \sum_{i=1}^{N^c} x_i^c \delta[n - n_i^c]$, for $n = 1, \cdots, N - L + 1$, and $\mathbf{x}^c = \Big[\mathbf{x}^c[1], \cdots, \mathbf{x}^c[N - L + 1]\Big]^{\mathrm{T}}$, are referred to as the *code* and the *code vector*, respectively.

We can express Eq. 5.3 in linear-algebraic form as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \boldsymbol{\varepsilon}, \tag{5.4}$$

where $\mathbf{H} = \Big[\mathbf{H}^1 \Big| \cdots \Big| \mathbf{H}^C\Big] \in \mathbb{R}^{N \times C(N-L+1)}$ is a block-Toeplitz matrix with c^{th} block $\mathbf{H}^c \in \mathbb{R}^{N \times (N-L+1)}$ for $c = 1, \cdots, C$, and $\mathbf{x} = [(\mathbf{x}^1)^{\mathrm{T}}, \cdots, (\mathbf{x}^C)^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{C(N-L+1)}$. The columns of the Toeplitz matrix $\mathbf{H}^c$ represent delayed versions (time-shifts) of $\mathbf{h}_c$, with integer delay between 0 and $N - L$, that have been zero-padded to length N. For each c, the non-zero entries of $\mathbf{x}^c$ represent the discrete-time indices $\{n_i^c\}_{i=1}^{N^c}$ when source c appears in the signal $\mathbf{y}$. Fig. 5-1 illustrates Eq. 5.4.

In practice, we divide the signal $\mathbf{y}$ into J non-overlapping windows of length W, such that $N = WJ$. We assume that $L \ll W \ll N$, so that the filters from each source are localized within a window. We denote by $\mathbf{Y} \in \mathbb{R}^{W \times J}$ the matrix whose

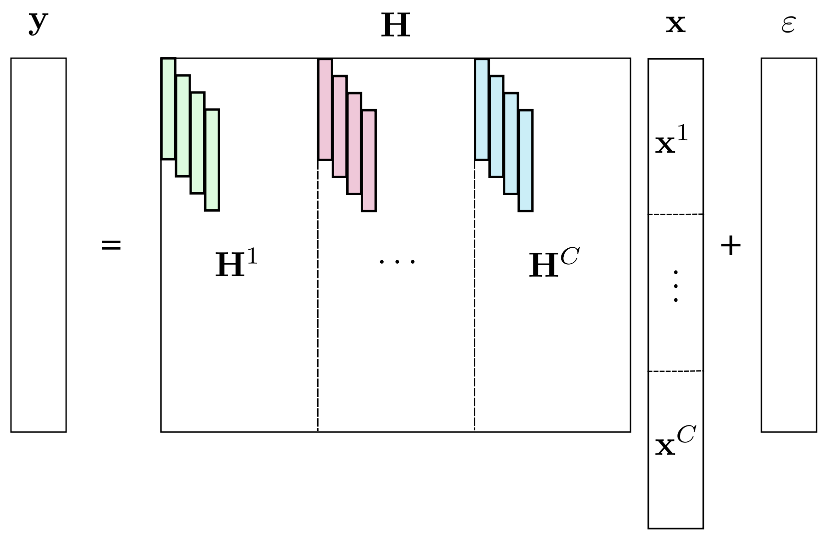

Figure 5-1: A schematic of Eq. 5.4. The Toeplitz matrix $\mathbf{H}^c$ represents all possible time-shifts of $\mathbf{h}_c$ with integer delays. The non-zero elements of each block $\mathbf{x}^c$ from $\mathbf{x}$ are the times when source c appears in the signal $\mathbf{y}$, $c = 1, \ldots, C$.

j^{th} column is $\mathbf{Y}_j = \Big[\mathbf{y}[(j-1)W+1], \cdots, \mathbf{y}[jW]\Big]^{\mathrm{T}} \in \mathbb{R}^W$, namely the j^{th} window from $\mathbf{y}[n]$. Similarly, we denote by $\mathbf{X} \in \mathbb{R}^{C(W-L+1)\times J}$ the coefficient matrix whose j^{th} column $\mathbf{X}_j = [(\mathbf{x}_j^1)^T, \cdots, (\mathbf{x}_j^C)^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{C(W-L+1)}$ is the code vector associated with window j.

5.2.3 CDL from optimization perspective

The goal of *Convolutional Dictionary Learning* (CDL) is to estimate $\{\mathbf{h}_c\}_c$ and $\{\mathbf{X}_j\}_j$ that minimize the error of reconstructing $\mathbf{Y}_j$ in each window using its linear approximation $\mathbf{H}\mathbf{X}_j$. We impose a sparsity constraint on the total number of nonzero elements of $\{\mathbf{X}_j\}_j$ for two reasons. First, without additional constraints, the problem as posed leads to an under-determined system of equations that does not have a unique solution. Second, in many applications, the rate of occurrence of events from the sources of interest is small compared to T, implying that each block $\mathbf{x}_j^c$ of the vector $\mathbf{X}_j$ is sparse. For example, in electrophysiological recordings of neural activity, we expect a sparse number of action potentials from neurons due to their biophysical properties [43]. Following [138], we use the ℓ_0 quasi-norm $\|\mathbf{X}_j\|_0$, which counts the number of non-zero elements of a vector, to express the sparsity constraint. Using this notation, the CDL problem is

$$\min_{\{\mathbf{h}_c\}_c, \{\mathbf{X}_j\}_j} \sum_{j=1}^{J} \left\| \mathbf{Y}_j - \mathbf{H}\mathbf{X}_j \right\|_2^2 \text{ s.t. } \|\mathbf{X}_j\|_0 \leq \beta_1, \quad \forall j, \tag{5.5}$$

where β_1 is a pre-defined sparsity threshold. One limitation of this approach comes from approximating the continuous-domain generative model (Eq. 5.2) with the discrete-domain generative model (Eq. 5.3). This approximation results in time-quantization errors, which manifest themselves in two ways: 1) a mismatch between the continuous time when the event occurs and its approximation in discrete time, $\tau_{j,i}^c \neq n_{j,i}^c \Delta$ and 2) template mismatch, $\mathbf{h}_c[m - n_{j,i}^c] = h_c(m\Delta - n_{j,i}^c \Delta) \neq h_c(m\Delta - \tau_{j,i}^c)$. One of our contributions is to introduce, in Section 5.4, a discrete-time generative model that mitigates the effects of time-quantization errors.

CDL is a nonconvex optimization problem, due to the simultaneous optimization over $\{\mathbf{h}_c\}_c$ and $\{\mathbf{X}_j\}_j$, and the presence of the ℓ_0 penalty. A popular approach is to alternatively minimize the objective over one set of variables while the other is fixed, until convergence. At iteration $s+1$, $\mathbf{X}^{(s+1)}$ is computed based on $\mathbf{H}^{(s)}$ through a *sparse coding* step, after which $\mathbf{H}^{(s+1)}$ is computed using $\mathbf{X}^{(s+1)}$ through a *dictionary update* step. If $\mathbf{H}$ is a convolutional matrix, we refer to these steps as *Convolutional Sparse Coding* (CSC) and *Convolutional Dictionary Update* (CDU), respectively. CSC approaches fall into two categories based on how the sparsity contraint is enforced. One class of approaches, the one we follow in this work, uses greedy methods to tackle the original problem with the ℓ_0 quasi-norm. Another class relaxes the ℓ_0 quasi-norm to the ℓ_1 norm, which converts the CSC objective into a convex optimization problem [129], [139], [140]. The advantage of greedy approaches is that they are more efficient computationally [135, 136]. Existing CSC frameworks that reduce time-quantization errors use the ℓ_1 norm [131, 132, 133, 134]. In the next section, we review both classes of CSC approaches, as well as approaches to solve the CDU step.

5.3 Background for CDL

For notational simplicity, we use $\mathbf{x}$ instead of $\mathbf{X}_j$ and $\mathbf{y}$ instead of $\mathbf{Y}_j$.

5.3.1 Convolutional greedy pursuit

Matching Pursuit (MP) [141] and Orthogonal Matching Pursuit (OMP) [142] are greedy methods to tackle the CSC step. We introduce the methods first when $\mathbf{H}$ is an arbitrary matrix, and then discuss the convolutional case.

Classical greedy pursuit–MP and OMP

Both MP and OMP iteratively select columns from $\mathbf{H}$ to produce an approximation $\mathbf{Hx}$ of $\mathbf{y}$. At iteration $t'+1$, the column of $\mathbf{H}$ with the maximal absolute inner product with the residual $\mathbf{r}^{(t')}$ is selected and added to set $\mathcal{S}^{(t')}$ comprising indices of active columns. The initial conditions are $\mathbf{r}^{(0)} = \mathbf{y}$ and $\mathcal{S}^{(0)} = \emptyset$. The two methods differ in how they compute the coefficients of the chosen columns and the residuals. Let $\mathbf{h}^{(t')}$ denote the template chosen at iteration t'.

- **MP** The coefficient associated with $\mathbf{h}^{(t')}$ and the residual are given, respectively, by $\langle \mathbf{h}^{(t')}, \mathbf{r}^{(t')} \rangle$ and $\mathbf{r}^{(t'+1)} = \mathbf{r}^{(t')} - \langle \mathbf{h}^{(t')}, \mathbf{r}^{(t')} \rangle \mathbf{h}^{(t')}$.

- **OMP** The coefficients associated with $\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(t')}$ are those that minimize the squared error between $\mathbf{y}$ and its linear reconstruction using the columns. This is equivalent to projecting $\mathbf{y}$ onto the span of $\mathbf{h}^{(1)}, \cdots, \mathbf{h}^{(t')}$, and is called the projection step.

The projection step implies two key differences between MP and OMP. First, OMP is slower than MP, due to the matrix inversion and multiplication required in the former. Second, as the residual $\mathbf{r}^{(t')}$ in OMP is orthogonal to the span of previously selected columns, OMP selects a different column from $\mathbf{H}$ at every iteration. This is not the case for MP, which means that it can select the same column multiple times.

Convolutional extensions

The convolutional extensions of MP and OMP exploit the fact that the templates are localized and shorter than the signal (or residual). Convolutional Matching Pursuit (CMP) [141], [143] has enjoyed popularity in biomedical applications [2] and image recognition [144] due to its simplicity. As in OMP, a convolutional extension of OMP involves computationally expensive selection and projection steps. In Section 5.4, we introduce an efficient algorithm for convolutional OMP. LocOMP [137] is related to this algorithm. The two algorithms differ in the projection step (Section 5.4-C).

5.3.2 CSC algorithms for estimating continuous-time shifts

Continuous basis pursuit (CBP) CBP approximates each continuous-time template $h_c(t)$ and its continuous-time shifts as the linear combination of $p > 1$ basis elements $\{\boldsymbol{\psi}_{(c,p)}\}_p$. Popular choices for $\{\boldsymbol{\psi}_{(c,p)}\}_p$ include the Taylor basis ($p = 2$), comprising $h_c(t)$ and its derivative, and the Polar basis ($p = 3$) comprising triogonometric splines. An ℓ_1-regularized convex optimization objective is formulated with respect to a shift-invariant representation using $\{\boldsymbol{\psi}_{(c,p)}\}_p$. The coefficients for $\{\boldsymbol{\psi}_{(c,p)}\}_p$ are then mapped to the amplitudes and the times when $h_c(t)$ occurs, with the mapping dependent on the choice of basis $\{\boldsymbol{\psi}_{(c,p)}\}_p$. The estimates of the times when events occur are continuous and not confined to the discrete sampling grid. Recently, continuous OMP [145] has been proposed as a greedy alternative to CBP that penalizes the ℓ_0 quasi-norm.

Greedy continuous approaches A recent line of works uses greedy continuous approaches based on the Frank-Wolfe algorithm (conditional gradient method) [132], [133, 134] to solve the CSC problem in continuous time. At a high-level, methods in this line of work alternate between a global and a local optimization step. We focus on the Alternating Descent Conditional Gradient (ADCG) [133] method, as [132] and [134] mostly differ from it in the local optimization step. At each iteration, ADCG greedily selects a *coarse* location for a code, based on the inner product between templates and the gradient of the residual, derived from the Frank-Wolfe algorithm. In practice, this step relies on discrete grids to compute the inner product. *As a result, it is equivalent to the selection step in CMP/COMP.* Then, the codes in the active set are further optimized locally by block coordinate descent, alternating between their locations and amplitudes. This latter step is grid-free and thus the locations are continuous. Unlike CBP, ADCG requires knowledge of the value of the template and its gradient at every point. This aspect makes it hard to apply to real datasets for which the convolutional filters are not known a-priori and need to be learned.

Grid refinement approaches This class of methods operates entirely on discrete grids, finer than the original sampling grid [146], [147]. Specifically, in [146], the authors demonstrate that the accuracy of sparse recovery on very-fine grids is competitive with that of the off-the-grid methods [133, 134]. Existing work on grid refinement for sparse recovery uses the ℓ_1 norm. More importantly, it requires knowledge of values of the templates on the refined grid. The CDL framework we propose also performs CSC with grid refinement. Unlike existing approaches, it 1) uses the ℓ_0 penalty and 2) only requires specification of the values of the templates on the original sampling grid. Fig. 5-2 shows an application of CBP, ADCG, BP, COMP and COMP-INTERP (COMP with interpolated dictionary) to the estimation of the continuous-time shift and amplitude of a single event from one filter. CBP, ADCG, and COMP-INTERP, the approach we propose in Section 5.4, are able to estimate the continuous-time shift accurately. As we demonstrate in Section 6.5, COMP-INTERP is orders of magnitude faster than CBP and ADCG. BP and COMP cannot capture the continuous-time shift, as they are confined to the original sampling grid.

5.3.3 CDU frameworks

The majority of existing CDU frameworks estimate the templates $\{\mathbf{h}_c\}_c$ by minimizing the error of reconstructing $\mathbf{y}$ using its linear approximation $\mathbf{Hx}$. The key differences between existing approaches are the constraints imposed on the templates and the optimization methods used, as detailed in a recent survey [129]. To the best of our knowledge, existing CDU approaches do not address the problem of learning the templates when the events of interest may occur off the discrete sampling grid.

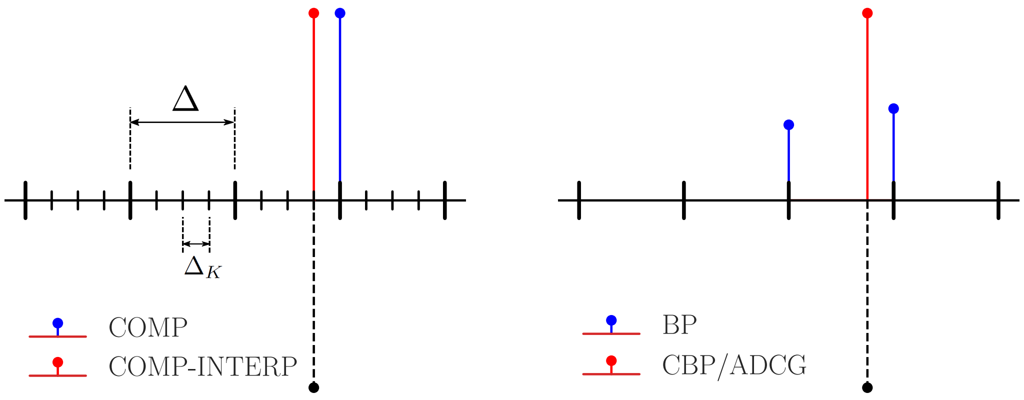

Figure 5-2: An application of CSC, where an event (black, flipped for clarity) occurs off the sampling grid. COMP and BP can only approximate the time of occurrence of the event on the grid, with spacing Δ. CBP, ADCG, and COMP-INTERP (COMP with interpolated dictionary) recover the time of occurrence accurately. COMP-INTERP uses grid refinement and operates on finer grid with resolution $\Delta_K \leq \Delta$.

5.4 Convolutional Orthogonal Matching Pursuit with Interpolated Dictionary

For the CSC step in the alternating-minimization approach to CDL, we introduce an algorithm for off-the-grid sparse coding called Convolutional OMP (COMP) with interpolated dictionary (COMP-INTERP). This is a convolutional greedy pursuit method that uses grid refinement to mitigate the effect of events that occur off the sampling grid. From a computational perspective, COMP-INTERP is an efficient alternative to the ℓ_1-based CSC frameworks described previously.

We use $\mathbf{x} \in \mathbb{R}^W$ and $\mathbf{y} \in \mathbb{R}^W$, instead of $\mathbf{X}_j$ and $\mathbf{Y}_j$, for notational simplicity. We use s and t', respectively, to denote an iteration of alternating-minimization and an iteration of COMP within it. Since the discussion in this section involves a single iteration of the alternating-minimization procedure, we drop s.

5.4.1 Non-integer delay through smooth interpolation

The discrete-time model from Eq. 5.3 is restrictive because events from the continuous-time generative model of Eq. 5.2 do not necessarily occur at multiples of the sampling

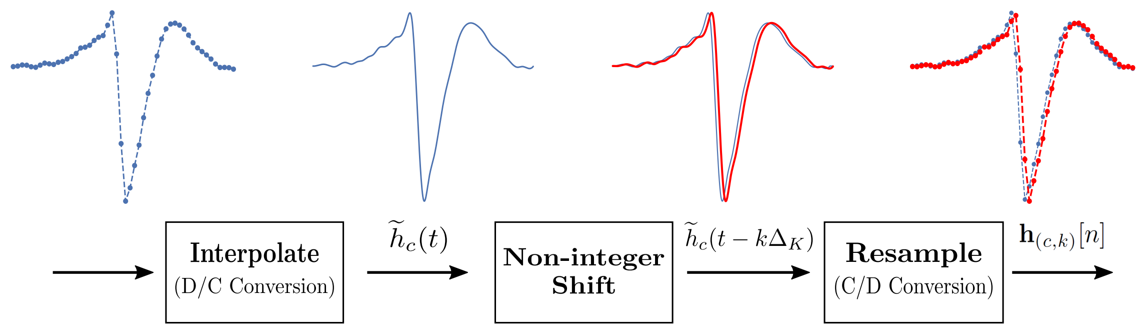

Figure 5-3: Illustration of the process for obtaining $\mathbf{h}_{(c,k)}$ (red) from $\mathbf{h}_c$ (blue). The two discrete-time templates in the rightmost panel highlight the difference between $\mathbf{h}_{(c,k)}$ and $\mathbf{h}_c$. The Interpolation and Resampling steps correspond, respectively, to D/C (Discrete-to-Continuous) conversion and C/D (Continuous-to-Discrete) conversion in digital signal processing theory.

interval Δ. We address this limitation by partitioning Δ into finer intervals of length $\Delta_K := \frac{1}{K}\Delta$, and modifying $\mathbf{H}$ and $\mathbf{x}$ accordingly. The resulting CSC framework, with finer resolution Δ_K, can approximate τ_i^c with $m\Delta + k\Delta_K$, where $m \in \mathbb{N}$ and $k = 0, \cdots, K-1$, rather than $m\Delta$, leading to a reduction in time-quantization error. That is, $|\tau_i^c - m\Delta| \geq |\tau_i^c - (m\Delta + k\Delta_K)|$. By definition, each template $\mathbf{h}_c$ corresponds to discrete-time samples of $h_c(t)$ with resolution Δ. Our challenge is to modify $\mathbf{H}$ to account for versions of $h_c(t)$ delayed by a non-integer amount $k\Delta_K$, that is not an integer multiple of Δ, *and* sampled at resolution Δ. Let $\mathbf{h}_{(c,k)} \in \mathbb{R}^L$ denote $h_c(t)$ delayed by a non-integer amount $k\Delta_K$ *and* sampled at resolution Δ. This definition motivates us to reformulate Eq. 5.3 to account for non-integer shifts of the templates at a finer scale Δ_K

$$\mathbf{y}[n] = \sum_{c=1}^{C}\sum_{k=0}^{K-1}\left(\mathbf{x}^{(c,k)} * \mathbf{h}_{(c,k)}\right)[n] + \boldsymbol{\varepsilon}[n], \tag{5.6}$$

where $\mathbf{x}^{(c,k)}$ denotes the code vector corresponding to $\mathbf{h}_{(c,k)}$. For notational simplicity, we let $\mathbf{h}_{(c,0)} = \mathbf{h}_c$. Note that $\mathbf{h}_c \neq \mathbf{h}_{(c,k)}$ for $k \neq 0$, as illustrated in Fig. 5-3. We discuss the systematic method for obtaining $\mathbf{h}_{(c,k)}$ from $\mathbf{h}_c$ in the next section. We use $\{\mathbf{h}_{(c,k)}\}_{c,k}$

to construct the interpolated convolutional dictionary $\mathbf{H}_{\text{INTERP}} \in \mathbb{R}^{W \times CK(W-L+1)}$

$$\begin{aligned}&\mathbf{H}_{\text{INTERP}} \\ &= \left[\mathbf{H}_{\text{INTERP}}^{(1,0)}\middle|\cdots\middle|\mathbf{H}_{\text{INTERP}}^{(1,K-1)}\middle|\cdots\middle|\mathbf{H}_{\text{INTERP}}^{(C,0)}\middle|\cdots\middle|\mathbf{H}_{\text{INTERP}}^{(C,K-1)}\right],\end{aligned} \tag{5.7}$$

where $\mathbf{H}_{\text{INTERP}}^{(c,k)}$ is the Toeplitz matrix whose columns consists of all integer shifts of $\mathbf{h}_{(c,k)}$. Note that when $K = 1$, we obtain the original convolutional dictionary, i.e., $\mathbf{H}_{\text{INTERP}} = \mathbf{H}$. In linear-algebraic form, we can write the generative model as $\mathbf{Y} = \mathbf{H}_{\text{INTERP}}\mathbf{X} + \boldsymbol{\varepsilon}$, where $\mathbf{X} \in \mathbb{R}^{CK(W-L+1)\times J}$.

5.4.2 Smooth interpolation of $\mathbf{h}_c$

We use the concept of *continuous-time operations on discrete-time signals* from digital signal processing theory [89] to obtain $\mathbf{h}_{(c,k)}$ from $\mathbf{h}_c$ via smooth interpolation. Smooth interpolation assumes that $h_c(t)$ is smooth, in the sense that the template does not change abruptly from one sample on the grid to the next. The process consists of three steps: interpolation, shifting by a non-integer amount, and resampling. These steps, illustrated in Fig. 5-3, perform

1. **Interpolation** Interpolate $\mathbf{h}_c$ with a smooth interpolator to obtain $\tilde{h}_c(t)$.

2. **Non-integer shift** Shift $\tilde{h}_c(t)$ to obtain $\tilde{h}_c(t - k\Delta_K)$.

3. **Resampling** Resample $\tilde{h}_c(t - k\Delta_K)$ with resolution Δ to obtain $\mathbf{h}_{(c,k)}$.

We ca write this three-step process concisely as a convolution. Let f be a generic smooth interpolator function, e.g. the cubic or the sinc interpolator. Then, $\tilde{h}_c(t)$ is given by

$$\tilde{h}_c(t) = \sum_{m=-\infty}^{\infty} \mathbf{h}_c[m] f(t - m\Delta), \tag{5.8}$$

from which we can obtain

$$
\begin{aligned}
\mathbf{h}_{(c,k)}[n] &= \tilde{h}_c(n\Delta - k\Delta_K) \\
&= \sum_{m=-\infty}^{\infty} \mathbf{h}_c[m] f(n\Delta - k\Delta_K - m\Delta) \\
&= \sum_{m=-\infty}^{\infty} \mathbf{h}_c[m] \overbrace{f\left((n-m)\Delta - k\Delta_K\right)}^{\mathbf{f}^k[n-m]} \\
&= \mathbf{h}_c * \mathbf{f}^k.
\end{aligned}
\tag{5.9}
$$

Effectively, $\mathbf{h}_{(c,k)}$ is the convolution between $\mathbf{h}_c$ and $\mathbf{f}^k[n] = f(n\Delta - k\Delta_K)$ for $n = -\frac{L-1}{2}, \dots, \frac{L-1}{2}$, where $\mathbf{f}^k \in \mathbb{R}^L$ is f shifted by a non-integer amount $k\Delta_K$ and resampled. Even though different interpolators have different lengths, we zero-pad and truncate $\mathbf{f}^k$, such that $\mathbf{f}^k$ is of length L.

We can express convolution in linear-algebraic form, which will be useful for the CDU step with the interpolated dictionary, detailed in Section 5.5: $\mathbf{h}_{(c,k)} = \mathbf{F}^k \mathbf{h}_c$, where $\mathbf{F}^k \in \mathbb{R}^{L \times L}$ is the Toeplitz matrix associated with $\mathbf{f}^k$ and defined as

$$
\mathbf{F}^k = \begin{pmatrix}
\mathbf{f}^k[0] & \mathbf{f}^k[-1] & \cdots & \mathbf{f}^k[-\frac{L-1}{2}] & \mathbf{0}^T_{(L-1)/2} \\
\mathbf{f}^k[1] & \mathbf{f}^k[0] & \cdots & \mathbf{f}^k[-\frac{L-1}{2}] & \mathbf{0}^T_{(L-3)/2} \\
\vdots & & \vdots & & \vdots \\
\mathbf{0}^T_{(L-1)/2} & \mathbf{f}^k[\frac{L-1}{2}] & \cdots & & \mathbf{f}^k[0]
\end{pmatrix}.
\tag{5.10}
$$

In general, $\tilde{h}_c(t)$ depends on the interpolator and does not necessarily coincide with the *true* template $h_c(t)$. Nevertheless, if $h_c(t)$ is sufficiently bandlimited, different interpolators should give $\tilde{h}_c(t)$ that are very similar to $h_c(t)$ [89]. In particular, if $h_c(t)$ is *bandlimited*, and the sampling frequency f_s is above its Nyquist rate, we can recover $h_c(t)$ from its discrete-time samples $\mathbf{h}_c$ using the sinc interpolator. This process, often referred to as *bandlimited interpolation* [89], guarantees that $h_c(t) = \tilde{h}_c(t)$. Using the sinc interpolator $f(t) = \mathrm{sinc}(t/\Delta) = \frac{\sin \pi(t/\Delta)}{\pi(t/\Delta)}$, we have

$$
\mathbf{f}^k[n] = \frac{\sin \pi(n - k/K)}{\pi(n - k/K)}.
\tag{5.11}
$$

In practice, since the sinc interpolator has infinite support, we use a truncated sinc interpolator, with truncation performed using a Kaiser window to mitigate truncation artifacts.

5.4.3 Efficient algorithm for COMP and COMP-INTERP

Matrix operations involving convolutional dictionaries $\mathbf{H}$ or $\mathbf{H}_{\text{INTERP}}$ are expensive both in terms of computation and storage requirements. This is because typical recordings can last on the order of minutes, if not hours, and sampling rates can be on the order of $\sim 10^4$ Hz for electrophysiology and 10^3 Hz for EEG, to name a few examples. Existing greedy algorithms for CSC can handle high-dimensional data in the selection step [143], [137]. Here, we focus instead on accelerating the projection step.

We explore efficient implementations of COMP and COMP-INTERP, focusing on the projection step for a convolutional matrix $\mathbf{H}$ (or $\mathbf{H}_{\text{INTERP}}$). For completeness, we describe the key principles of the efficient implementation of the selection step, proposed in [137]. For notational simplicity, we focus on $\mathbf{H}$, noting that the same argument holds for $\mathbf{H}_{\text{INTERP}}$.

Selection step The selection step requires the inner product between time-shifted $\mathbf{h}_c$ and $\mathbf{r}^{(t')}$, expressed as $\mathbf{H}^{\mathrm{T}}\mathbf{r}^{(t')}$. For large $\mathbf{H}$, explicit computation of the inner product is expensive. However, the convolutional structure of $\mathbf{H}$ lets us compute C cross-correlations instead,

$$\begin{aligned}\mathbf{H}^{\mathrm{T}}\mathbf{r}^{(t')} =&\Big[(\mathbf{h}_1 \star \mathbf{r}^{(t')})[1], \cdots, (\mathbf{h}_1 \star \mathbf{r}^{(t')})[W-L+1], \cdots, \\ &(\mathbf{h}_C \star \mathbf{r}^{(t')})[1], \cdots, (\mathbf{h}_C \star \mathbf{r}^{(t')})[W-L+1]\Big]^{\mathrm{T}}.\end{aligned} \tag{5.12}$$

This formulation has two computational benefits. First, we do not need to construct the convolutional matrix $\mathbf{H}$ explicitly. We only require $O(W)$ memory to store $\{\mathbf{h}_c\}_{c=1}^{C}$ and $\mathbf{r}^{(t')}$, as opposed to $O(CW^2)$ memory to store the matrix. Second, we can compute the C cross-correlation operations using the FFT, which is much more efficient than computing them by multiplication of $\mathbf{H}^{\mathrm{T}}$ and $\mathbf{r}^{(t')}$.

Furthermore, we can exploit the fact that the templates are much shorter than the residual, implying that $\mathbf{r}^{(t')}$ and $\mathbf{r}^{(t'-1)}$ only differ locally. Therefore, we only need to compute the cross-correlation for local segments in which the two differ. For CMP, the length of such segments is always fixed, whereas for COMP, the length can increase due to potential overlaps with other templates. The authors who introduced LocOMP [137] are arguably the first to recognize this. As pointed out in [137], in the regime of a large number of overlaps between templates, the selection step dominates the difference in computation time between CMP and COMP. Our focus is on the regime in which the number of overlaps is small. In this regime, the projection step is the dominating factor. The applications we consider in Section 6.5 fall into this regime.

Projection step In this step, we project the residual onto the span of $\mathbf{H}\big|_{t'}$, which requires the inversion of $\mathbf{H}\big|_{t'}^{\mathrm{T}}\mathbf{H}\big|_{t'}$. The matrix $\mathbf{H}\big|_{t'} \in \mathbb{R}^{W\times t'}$ refers to a convolutional dictionary restricted to columns that have been selected by COMP up to iteration t'. Consequently, the code $\mathbf{x}|_{t'} \in \mathbb{R}^{t'}$ refers to the nonzero coefficients from $\mathbf{x} \in \mathbb{R}^{W-L+1}$ corresponding to the columns $\mathbf{H}\big|_{t'}$. To avoid the computational cost of inversion in the projection step, [148] suggested an efficient method for computing the Cholesky factor $\mathbf{L}^{(t')}$, which is a lower triangular matrix such that $\mathbf{L}^{(t')}\left(\mathbf{L}^{(t')}\right)^{\mathrm{T}} = \mathbf{H}\big|_{t'}^{\mathrm{T}}\mathbf{H}\big|_{t'} \in \mathbb{R}^{t'\times t'}$. The key idea is that for OMP, $\mathbf{H}\big|_{t'-1}$ and $\mathbf{H}\big|_{t'}$ differ only by one column, which is the column selected by OMP at step t', and therefore $\mathbf{L}^{(t')}$ can be easily computed from $\mathbf{L}^{(t'-1)}$ as

$$\begin{gathered}\mathbf{L}^{(t')} = \begin{pmatrix} \mathbf{L}^{(t'-1)} & 0 \\ \mathbf{w}^{\mathrm{T}} & \sqrt{1-\|\mathbf{w}\|_2^2} \end{pmatrix} \\ \text{where } \mathbf{L}^{(t'-1)}\mathbf{w} = \mathbf{H}\big|_{t'-1}^{\mathrm{T}}\mathbf{h}^{(t')}, \end{gathered} \tag{5.13}$$

and $\mathbf{h}^{(t')} \in \mathbb{R}^{W}$ denotes the column of $\mathbf{H}$ selected at iteration t' of COMP. The code $\mathbf{x}\big|_{t'}$ is is the solution to $\mathbf{L}^{(t')}\left(\mathbf{L}^{(t')}\right)^{\mathrm{T}}\mathbf{x}|_{t'} = \mathbf{H}\big|_{t'}^{\mathrm{T}}\mathbf{y}$, which can be solved more efficiently than $\mathbf{H}\big|_{t'}^{\mathrm{T}}\mathbf{H}\big|_{t'}\mathbf{x}|_{t'} = \mathbf{H}\big|_{t'}^{\mathrm{T}}\mathbf{y}$.

We extend this idea to the convolutional case, noting that Eq. 5.13 still requires us to construct $\mathbf{H}\Big|_{t'}$ and to perform multiplications that are expensive in terms of memory and computation. We replace the multiplication operation involving $\mathbf{H}\Big|_{t'}^{\mathrm{T}}$ with a cross-correlation operation, as outlined in Algorithm 1. To keep track of the selected filters, we utilize two sets: 1) The set $\mathcal{S}$ of indices of active template chosen by COMP and defined in Section 6.2, and 2) the set $\mathcal{I}$ of times when events associated with each of the templates from $\mathcal{S}$ occur. At COMP iteration t', $\mathbf{h}_{\mathcal{S}_i^{(t')}}$ refers to the template selected at iteration i, where $i \leq t'$, and $\mathcal{I}_i^{(t')}$ refers to the time of occurrence of the corresponding template.

Algorithm 1: Cholesky factorization for CSC at iteration t'

```
   Input: $\mathbf{L}^{(t'-1)}$, $\mathcal{S}^{(t')}$, $\mathcal{I}^{(t')}$, $\{\mathbf{h}_c\}_c$, $\mathbf{y}$
   Output: $\mathbf{L}^{(t')}$, $\mathbf{x}|_{t'}$
1  Initialization: $\mathbf{v} = \mathbf{0}_{t'-1}$, $\boldsymbol{\alpha} \in \mathbb{R}^{t'}$
2  if $t' = 1$ then
3      $\mathbf{L}^{(t')} \leftarrow [1]$
4  else
5      for $i \leftarrow 1$ to $t' - 1$ do
6          if $\left|\mathcal{I}_i^{(t')} - \mathcal{I}_{t'}^{(t')}\right| \leq L$ then
7              $\mathbf{v}[i] = \left(\mathbf{h}_{\mathcal{S}_i^{(t')}} \star \mathbf{h}_{\mathcal{S}_{t'}^{(t')}}\right)\left[\mathcal{I}_i^{(t')} - \mathcal{I}_{t'}^{(t')}\right]$
8      Solve for $\mathbf{w} \in \mathbb{R}^{t'-1}$: $\mathbf{L}^{(t'-1)}\mathbf{w} = \mathbf{v}$
9      $\mathbf{L}^{(t')} \leftarrow \begin{pmatrix} \mathbf{L}^{(t'-1)} & 0 \\ \mathbf{w}^{\mathrm{T}} & \sqrt{1 - \|\mathbf{w}\|_2^2} \end{pmatrix}$
10 for $i \leftarrow 1$ to $t'$ do
11     $\boldsymbol{\alpha}[i] = (\mathbf{h}_{\mathcal{S}_i^{(t')}} \star \mathbf{y})\left[\mathcal{I}_i^{(t')}\right]$
12 Solve for $\mathbf{L}^{(t')}\left(\mathbf{L}^{(t')}\right)^{\mathrm{T}}\mathbf{x}|_{t'} = \boldsymbol{\alpha}$
```

With $\mathcal{S}$ and $\mathcal{I}$, neither the convolutional matrix $\mathbf{H}\Big|_{t'}$, nor the zero-padded filters are required. As we demonstrate in Section 6.5, the efficiency gain from the modified projection step reduces the performance gap betwen CMP and COMP.

Algorithm We summarize COMP-INTERP in Algorithm 2. When $K = 1$, COMP-INTERP is equivalent to COMP with the original dictionary. The INTERPOLATE function refers to the process of obtaining the interpolated templates. The CHOLESKY function refers to the efficient projection step. The superscript $(*)$ denotes the quantities at convergence. As in OMP, the convergence criterion can either be when the residual falls below a certain threshold or when a certain sparsity level is reached. We note that, as mentioned previously, we only need to compute the cross-correlation (line 5) for local segments where consecutive residuals differ.

Algorithm 2: COMP-INTERP

```
   Input: $\mathbf{y}$, $\{\mathbf{h}_c\}_c$, $\Delta_K$
   Output: $\mathbf{x}^{(*)}, \mathcal{S}^{(*)}, \mathcal{I}^{(*)}$
1  Initialization: $\mathbf{r}^{(0)} = \mathbf{y}$, $\mathbf{L}^{(0)} = 1$, $\mathcal{S}^{(0)}, \mathcal{I}^{(0)} = \emptyset$
2  $\{\mathbf{h}_{(c,k)}\}_{c,k} \leftarrow$ Interpolate($\{\mathbf{h}_c\}_c$, $\Delta_K$)
3  while $t' = 0$ to convergence do
4      (Selection step)
5      $c^*, k^*, i^* \leftarrow \arg\max_{c,k,i}\{\left|\mathbf{h}_{(c,k)} \star \mathbf{r}^{(t')}\right|[i]\}_{c,k,i}$
6      $\mathcal{S}^{(t'+1)} = \mathcal{S}^{(t')} \cup \{(c^*, k^*)\}$
7      $\mathcal{I}^{(t'+1)} = \mathcal{I}^{(t')} \cup \{i^*\}$
8
9      (Projection step)
10     $\mathbf{L}^{(t'+1)}, \mathbf{x}|_{t'+1} \leftarrow$ CHOLESKY($\theta$), where
11         $\theta = \left\{\mathbf{L}^{(t')}, \mathcal{S}^{(t'+1)}, \mathcal{I}^{(t'+1)}, \{\mathbf{h}_{(c,k)}\}_{c,k}, \mathbf{y}\right\}$
12     $\mathbf{r}^{(t'+1)} \leftarrow \mathbf{y}$
13     for $i \leftarrow 1$ to $t' + 1$ do
14         Subtract $\mathbf{h}_{\mathcal{S}_i^{(t'+1)}} \mathbf{x}|_{t'+1}$ from the segment of $\mathbf{r}^{(t'+1)}$ that starts at $\mathcal{I}_i^{(t'+1)}$
```

5.4.4 Equivalence between interpolating the template and interpolating the residual

As described thus far, COMP-INTERP mitigates errors from events that occur off the original sampling grid by introducing smoothly-interpolated versions of the templates on a refined grid, while preserving the signal of interest on the original grid. Because the algorithm relies on correlation operations between the interpolated templates and the signal, what is important in fact is the *relative* placement of the templates with respect to the signal $\mathbf{y}$ (or $\mathbf{r}$). This motivates an alternate approach that, instead, preserves the templates on the original grid and smoothly interpolates $\mathbf{y}$ (or $\mathbf{r}$) on the refined grid. This section examines the equivalence between these two approaches.

Specifically, we show that COMP-INTERP is *equivalent* to COMP using a smoothly-interpolated augmented data set (Theorem 1). We first develop two Lemmas showing the equivalence between certain cross-correlation and convolution operations. The Lemmas assume that the discrete interpolator $\mathbf{f}$ is *symmetric*, which holds for interpolators such as the linear, cubic, sinc, and Lanczos interpolators.

Lemma 1. *For a symmetric interpolator* $\mathbf{f}$ *and signal (or residual)* $\mathbf{r}$*, we have* $\mathbf{f}^k \star \mathbf{r} = \mathbf{f}^{-k} * \mathbf{r}$.

Proof.

$$\begin{aligned}\mathbf{f}^{-k} * \mathbf{r} &= \sum_m f\Big(m\Delta + k\Delta_K\Big)\mathbf{r}[n-m] \\ &= \sum_{m'} f\Big(-m'\Delta + k\Delta_K\Big)\mathbf{r}[n+m'] \\ &= \sum_{m'} f\Big(m'\Delta - k\Delta_K\Big)\mathbf{r}[n+m'] = \mathbf{f}^k \star \mathbf{r},\end{aligned} \tag{5.14}$$

where the second equality is a simple change of variables, and the third equality uses the fact that $\mathbf{f}$ is symmetric. □

The next Lemma applies Lemma 1 to the COMP-INTERP selection step.

Lemma 2. $\mathbf{h}_{(c,k)} \star \mathbf{r} = \mathbf{h}_c \star (\mathbf{f}^{-k} * \mathbf{r})$.

Proof.

$$\begin{aligned}\mathbf{h}_{(c,k)} \star \mathbf{r} &= (\mathbf{h}_c * \mathbf{f}^k) \star \mathbf{r} \\ &= \sum_m \Big(\sum_p \mathbf{h}_c[p]\mathbf{f}^k[m-p]\Big)\mathbf{r}[m+n] \\ &= \sum_p \mathbf{h}_c[p]\Big(\sum_m \mathbf{r}[m+n]\mathbf{f}^k[m-p]\Big) \\ &= \sum_p \mathbf{h}_c[p](\mathbf{f}^k \star \mathbf{r})[n+p] = \mathbf{h}_c \star (\mathbf{f}^k \star \mathbf{r})\end{aligned} \tag{5.15}$$

Using Lemma 1, we have $\mathbf{h}_{(c,k)} \star \mathbf{r} = \mathbf{h}_c \star (\mathbf{f}^{-k} * \mathbf{r})$. □

Lemma 2 states that performing the COMP-INTERP selection step with interpolated $\mathbf{h}_{(c,k)}$, on the *original* $\mathbf{r}$, is equivalent to performing a COMP selection step with the *original* $\mathbf{h}_c$, on $\mathbf{f}^{-k} * \mathbf{r}$, which is $\mathbf{r}$ delayed by $-k\Delta_K$.

Before stating the theorem, we introduce some notation. COMP-INTERP($\mathbf{y}, \{\mathbf{h}_{(c,k)}\}_{c,k}$) refers to COMP-INTERP applied to $\mathbf{y}$ (using the interpolated dictionary) and COMP($\{\mathbf{f}^{-k} * \mathbf{y}\}_k, \{\mathbf{h}_c\}_c$) refers to COMP on an augmented dataset denoted by $\{\mathbf{f}^{-k} * \mathbf{y}\}_k$. We define COMP on the augmented set, $\{\mathbf{f}^{-k} * \mathbf{y}\}_k$, as follows: At iteration t', in the selection step, compute $c^{(*,k)}$ and $i^{(*,k)}$ for each $\mathbf{f}^{-k} * \mathbf{r}^{(t')}$ with $k = 0, \cdots, K-1$, via the usual COMP selection step. Next, choose k^* such that

$$k^* = \arg\max_k \left\{ \left| \mathbf{h}_{c^{(*,k)}} \star (\mathbf{f}^{-k} * \mathbf{r}^{(t')}) \right| [i^{(*,k)}] \right\}_k . \tag{5.16}$$

Finally, we add the indices to the active sets, such that $\mathcal{S}^{(t'+1)} = \mathcal{S}^{(t')} \cup \{(c^{(*,k^*)}, k^*)\}$, $\mathcal{I}^{(t'+1)} = \mathcal{I}^{(t')} \cup \{i^{(*,k^*)}\}$ to complete the selection step. The projection step is same as that of COMP-INTERP($\mathbf{y}, \{\mathbf{h}_{(c,k)}\}_{c,k}$).

Theorem 1. *The two algorithms, COMP*($\{\mathbf{f}^{-k} * \mathbf{y}\}_k, \{\mathbf{h}_c\}_c$) *and COMP-INTERP*($\mathbf{y}, \{\mathbf{h}_{(c,k)}\}_{c,k}$), *are equivalent. That is,* $\mathcal{S}^{(*)}$, $\mathcal{I}^{(*)}$, *and* $\mathbf{r}^{(*)}$ *are same for both upon convergence.*

Proof. We proceed by induction. At iteration $t' = 0$, $\mathcal{S}^{(0)} = \emptyset$ and $\mathbf{r}^{(0)} = \mathbf{y}$. In the selection step, Lemma 2 implies that c^*, k^*, i^* from the COMP-INTERP selection step are same as $c^{(*,k^*)}, k^*, i^{(*,k^*)}$ from the COMP selection step. In turn, this implies that $\mathcal{S}^{(1)}$ and $\mathcal{I}^{(1)}$ are the same. Since the projection steps of the two methods are equivalent, and the active sets are same, both methods yield the same residual $\mathbf{r}^{(1)}$. Now, assume that the equivalence holds for $\mathcal{S}^{(t'-1)}$, $\mathcal{I}^{(t'-1)}$, and $\mathbf{r}^{(t'-1)}$. For iteration t', the same argument as for iteration $t' = 0$ holds. Hence, the two algorithms produce the same $\mathcal{S}^{(t')}$, $\mathcal{I}^{(t')}$, and $\mathbf{r}^{(t')}$. We conclude that both approaches are equivalent. $\square$

Remark: The key component of the proof is the equivalence of the *selection step*. The equivalence between the two algorithms gives us a different understanding of COMP-INTERP and does not provide additional computational gains.

5.4.5 Comparison with the continuous approaches

The continuous-basis approximation methods (CBP [131] and continuous OMP [145]), the greedy continuous methods (ADCG [133]), and the methods based on grid refinement (COMP-INTERP) implicitly assume smoothness of the continuous-time templates. The continuous-basis approximation relies on this assumption for the derivation of the mapping between the local basis and the templates. ADCG requires the gradient of each template at every point. COMP-INTERP uses smooth interpolators to model the non-integer effect.

The methods differ in many ways, as summarized in Table 5.2. The most notable difference is the nature of the estimated event locations. COMP/COMP-INTERP estimate them on a discrete grid, whether it is the original sampling grid or a refined one. On the other hand, CBP and ADCG estimate do not confine event locations to a discrete grid. These methods, however, require operations on a discrete grid as an intermediate step for computing the continuous event locations.

	COMP	COMP-INTERP	ADCG	CBP
Smoothness	No	Yes	Yes	Yes
Functional form	No	No	Yes	No
FIR	Yes	Yes	No	Yes

(a) Assumptions on the templates. FIR refers to finite impulse response.

	COMP	COMP-INTERP	ADCG	CBP
Original grid	Yes	No	No	No
Refined grid	No	Yes	No	No
Continuous	No	No	Yes	Yes

(b) Domain of esimated event times.

Table 5.2: Comparison of convolutional CSC approaches

Compared to the continuous methods, the advantages of COMP-INTERP are its simplicity and speed. It is simpler as it requires neither a mapping between local basis and the original templates, as in CBP, nor functional forms for the templates or their gradients, as in ADCG. It is also faster due to its greedy nature, and because of the efficient implementations of operations that involve the convolutional dictionary.

5.5 Convolutional Dictionary Update with interpolated dictionary

COMP-INTERP is an algorithm to tackle the CSC step of CDL using an interpolated dictionary that can approximate continuous shifts. In this section, we develop an algorithm to solve the CDU step using the interpolated dictionary.

The CDU step involves an optimization problem with respect to $\{\mathbf{h}_c\}_c$. To simplify it, we first re-write $\mathbf{HX}_j$ as

$$\mathbf{HX}_j = \sum_{c=1}^{C}\sum_{i=1}^{N_j^c} x_{j,i}^c \mathbf{S}_{n_{j,i}^c}\mathbf{h}_c, \tag{5.17}$$

where we introduce the matrix representation $\mathbf{S}_{n_{j,i}^c} \in \mathbb{R}^{W\times L}$ of the linear operator that shifts $\mathbf{h}_c$ by $n_{j,i}^c$ samples. $\mathbf{S}_{n_{j,i}^c}$ is a zero-padded identity matrix defined as follows

$$\mathbf{S}_{n_{j,i}^c} = \begin{pmatrix}\mathbf{0}_{n_{j,i}^c\times L} & \mathbf{I}_{L\times L} & \mathbf{0}_{(W-L-n_{j,i}^c)\times L}\end{pmatrix}^{\mathrm{T}}. \tag{5.18}$$

Eq. 5.17 is a result of the commutativity of the convolution operation. It allows us to re-write the optimization problem in Eq. 5.5, with respect to $\{\mathbf{h}_c\}_c$, as follows

$$\min_{\{\mathbf{h}_c\}_c} \sum_{j=1}^{J} \left\| \mathbf{Y}_j - \sum_{c=1}^{C}\sum_{i=1}^{N_j^c} x_{j,i}^c \mathbf{S}_{n_{j,i}^c}\mathbf{h}_c \right\|_2^2. \tag{5.19}$$

Compared to Eq. 5.5, Eq. 5.19 is simpler because $\mathbf{h}_c$ appears as a vector, as opposed to a matrix. Using the interpolated dictionary $\mathbf{H}_{\text{INTERP}}$ to account for the non-integer delays, we can write the objective similarly

$$\begin{aligned}&\min_{\{\mathbf{h}_c\}_c} \sum_{j=1}^{J} \left\| \mathbf{Y}_j - \sum_{c=1}^{C}\sum_{k=0}^{K-1}\sum_{i=1}^{N_j^{(c,k)}} x_{j,i}^{(c,k)} \mathbf{S}_{n_{j,i}^{(c,k)}}\mathbf{h}_{(c,k)} \right\|_2^2 \\ &\text{s.t. } \mathbf{h}_{(c,k)} = \mathbf{F}^k\mathbf{h}_c, \quad \forall k, c,\end{aligned} \tag{5.20}$$

where $\mathbf{F}^k$ is the discrete interpolator matrix defined previously. The constraint enforces the fact that we obtain $\mathbf{h}_{(c,k)}$ from $\mathbf{h}_c$ by interpolation. As is customary in the CDL literature, we assume the templates have unit norm, which we enforce by normalizing the solutions of Eq. 5.19 or Eq. 5.20.

<u>Remark</u>: For notational simplicity, we use x_i and $\mathbf{S}_i$ for $x_{j,i}^{(c,k)}$ and $\mathbf{S}_{n_{j,i}^{(c,k)}}$, respectively. That is, we overload the index i. As a result, Eq. 5.19 becomes

$$\min_{\{\mathbf{h}_c\}_{c=1}^{C}} \sum_{j=1}^{J} \left\| \mathbf{Y}_j - \sum_{c=1}^{C} \sum_{i=1}^{N_j^c} x_i \mathbf{S}_i \mathbf{h}_c \right\|_2^2 \tag{5.21}$$

and Eq. 5.20 becomes

$$\begin{aligned} &\min_{\{\mathbf{h}_c\}_{c=1}^{C}} \sum_{j=1}^{J} \left\| \mathbf{Y}_j - \sum_{c=1}^{C} \sum_{k=0}^{K-1} \sum_{i=1}^{N_j^{(c,k)}} x_i \mathbf{S}_i \mathbf{h}_{(c,k)} \right\|_2^2 \\ &\text{s.t. } \mathbf{h}_{(c,k)} = \mathbf{F}^k \mathbf{h}_c, \quad \forall k, c. \end{aligned} \tag{5.22}$$

5.5.1 CDU with original, non-interpolated, dictionary

To solve Eq. 5.19, we can use any least-squares based algorithm. We focus on KSVD [138], specifically its shift-invariant version [149]. We emphasize two key ideas from KSVD: 1) the templates $\{\mathbf{h}_c\}_c$ are updated sequentially, i.e., one at a time, and 2) when updating a given template, only windows of the data where the template occurs need to be considered.

Suppose we want to update $\mathbf{h}_d$. Let $\Omega_d \subset \{1, \cdots, J\}$ denote the set of indices of windows from which COMP selects at least one occurrence of $\mathbf{h}_d$. For each window $j \in \Omega_d$, we split the sum from Eq. 5.19 into two parts, namely one that involves $\mathbf{h}_d$ and another that involves the remaining templates. The new estimate of $\mathbf{h}_d$, denoted $\widehat{\mathbf{h}}_d$, is given by

$$\begin{aligned} &\widehat{\mathbf{h}}_d = \arg\min_{\mathbf{h}_d} \sum_{j \in \Omega_d} \left\| \mathbf{E}_j - \sum_{i=1}^{N_j^d} x_i \mathbf{S}_i \mathbf{h}_d \right\|_2^2, \\ &\text{where } \mathbf{E}_j = \mathbf{Y}_j - \sum_{c \neq d}^{C} \sum_{i=1}^{N_j^c} x_i \mathbf{S}_i \mathbf{h}_c \end{aligned} \tag{5.23}$$

is the residual from approximating $\mathbf{Y}_j$ with templates other than $\mathbf{h}_d$. The solution $\widehat{\mathbf{h}}_d$ to the least-squares problem is

$$\begin{aligned}\widehat{\mathbf{h}}_d = &\left(\sum_{j\in\Omega_d}\sum_{i=1}^{N_j^d}\sum_{m=1}^{N_j^d} x_i\Big(\mathbf{S}_i\Big)^{\mathrm{T}}\mathbf{S}_m x_m\right)^{-1}\\ &\times\left(\sum_{j\in\Omega_d}\sum_{i=1}^{N_j^d} x_i\Big(\mathbf{S}_i\Big)^{\mathrm{T}}\mathbf{E}_j\right).\end{aligned} \tag{5.24}$$

We can interpret Eq. 5.24 as the weighted average of segments $\mathbf{E}_j$, or $\Big(\mathbf{S}_i\Big)^{\mathrm{T}}\mathbf{E}_j \in \mathbb{R}^L$, in which template $\mathbf{h}_d$ occurs. The average is normalized by a factor that accounts for occurrences of $\mathbf{h}_d$ that overlap: the term $\Big(\mathbf{S}_i\Big)^{\mathrm{T}}\mathbf{S}_m \in \mathbb{R}^{L\times L}$ is a matrix that is non-zero only if the offset $|n_{j,i}^d - n_{j,m}^d|$ between occurrences of $\mathbf{h}_d$ is less than the template length L.

5.5.2 CDU with interpolated dictionary (CDU-INTERP)

To solve Eq. 5.20, we follow an approach similar to that used to solve Eq. 5.19. The constraint from Eq. 5.20 implies that

$$\mathbf{S}_i\mathbf{h}_{(c,k)} = \mathbf{S}_i\mathbf{F}^k\mathbf{h}_c. \tag{5.25}$$

Eq. 5.25 allows us to rewrite Eq. 5.20 as an unconstrained optimization problem

$$\min_{\{\mathbf{h}_c\}_c}\sum_{j=1}^{J}\left\|\mathbf{Y}_j - \sum_{c=1}^{C}\sum_{k=0}^{K-1}\sum_{i=1}^{N_j^{(c,k)}} x_i\mathbf{S}_i\mathbf{F}^k\mathbf{h}_c\right\|_2^2. \tag{5.26}$$

Suppose we want to update $\mathbf{h}_d$. Let $\widetilde{\Omega}_d = \cup_{k=0}^{K-1}\Omega_{(d,k)}$ be the set of indices of windows from which COMP-INTERP selects at least one occurrence of a template from the set $\{\mathbf{h}_{(d,k)}\}_k$. Re-arranging Eq. 5.26 yields the estimate $\widehat{\mathbf{h}}_d$ of $\mathbf{h}_d$

$$\begin{aligned}\widehat{\mathbf{h}}_d &= \arg\min_{\mathbf{h}_d} \sum_{j\in\widetilde{\Omega}_d} \left\| \mathbf{E}_j - \sum_{k=0}^{K-1} \sum_{i=1}^{N_j^{(d,k)}} x_i \mathbf{S}_i \mathbf{F}^k \mathbf{h}_d \right\|_2^2, \\ &\text{where } \mathbf{E}_j = \mathbf{Y}_j - \sum_{c\neq d}^{C} \sum_{k=0}^{K-1} \sum_{i=1}^{N_j^{(c,k)}} x_i \mathbf{S}_i \mathbf{F}^k \mathbf{h}_c, \text{ and}\end{aligned} \tag{5.27}$$

the solution of which is given by

$$\begin{aligned}\widehat{\mathbf{h}}_d = &\left(\sum_{j\in\widetilde{\Omega}_d} \sum_{k=0}^{K-1} \sum_{i=1}^{N_j^{(d,k)}} \sum_{m=1}^{N_j^{(d,k)}} x_i \left(\mathbf{F}^k\right)^{\mathrm{T}} \left(\mathbf{S}_i\right)^{\mathrm{T}} \mathbf{S}_m \mathbf{F}^k x_m \right)^{-1} \\ &\times \left(\sum_{j\in\widetilde{\Omega}_d} \sum_{k=0}^{K-1} \sum_{i=1}^{N_j^{(d,k)}} x_i \left(\mathbf{F}^k\right)^{\mathrm{T}} \left(\mathbf{S}_i\right)^{\mathrm{T}} \mathbf{E}_j \right).\end{aligned} \tag{5.28}$$

The interpretation of the term $\left(\mathbf{F}^k\right)^{\mathrm{T}}\left(\mathbf{S}_i\right)^{\mathrm{T}}\mathbf{E}_j$ in Eq. 5.28 is as follows. First, $(\mathbf{F}^k)^{\mathrm{T}}$ performs a cross-correlation, and thus $\left(\mathbf{F}^k\right)^{\mathrm{T}}\left(\mathbf{S}_i\right)^{\mathrm{T}}\mathbf{E}_j = \mathbf{f}^k \star \left(\left(\mathbf{S}_i\right)^{\mathrm{T}}\mathbf{E}_j\right)$. Next, using Lemma 1 from Section 5.4.4, we have $\left(\mathbf{F}^k\right)^{\mathrm{T}}\left(\mathbf{S}_i\right)^{\mathrm{T}}\mathbf{E}_j = \mathbf{f}^{-k} * \left(\left(\mathbf{S}_i\right)^{\mathrm{T}}\mathbf{E}_j\right)$. This operation 1) extracts from $\mathbf{E}_j$ the segment corresponding to the code and 2) shifts the segment by $-k\Delta_K$, thereby aligning the extracted segments. This alignment results in a more accurate dictionary update, as demonstrated in Section 6.5.

5.5.3 CDL algorithm

We summarize the alternating-minimization procedure for CDL in Algorithm 3. COMP-INTERP refers to Algorithm 2 and CDU-INTERP performs the dictionary update. The $*$ notation from $\mathcal{S}^{j,(*)}$ and $\mathcal{I}^{j,(*)}$ refer to the index sets for window j at COMP-INTERP convergence. We use $\{\mathbf{h}_c^{(*,\Delta_K)}\}_c$ to denote the templates learned with COMP-INTERP/CDU-INTERP with discretization Δ_K. COMP-INTERP is parallelizable across J windows, and therefore amenable to parallel processing. The CDU step is not parallelizable because it needs to aggregate the occurrences of the templates across all J windows.

Algorithm 3: CDL with an interpolated dictionary

```
Input: $\mathbf{Y}$, $\{\mathbf{h}_c^{(0)}\}_c$, $\Delta_K$
Output: $\mathbf{X}^{(*)}$, $\{\mathbf{h}_c^{(*,\Delta_K)}\}_c$
while $s = 0$ to convergence do
    ($CSC$ step)
    for $j = 1$ to $J$ do
        $\mathbf{X}_j^{(s+1)}$, $\mathcal{S}^{j,(*)}$, $\mathcal{I}^{j,(*)}$
        $\leftarrow$ COMP-INTERP$\Big(\mathbf{Y}_j, \{\mathbf{h}_c^{(s)}\}_c, \Delta_K\Big)$

    ($CDU$ step)
    for $c = 1$ to $C$ do
        $\widehat{\mathbf{h}}_c \leftarrow$CDU-INTERP($\theta$), where
        $\theta = \{\mathbf{Y}, \{\mathbf{h}_c^{(s)}\}_c, \{\mathbf{X}_j^{(s+1)}, \mathcal{S}^{j,(*)}, \mathcal{I}^{j,(*)}\}_j, \Delta_K\}$
    $\{\mathbf{h}_c^{(s+1)}\}_c \leftarrow \{\widehat{\mathbf{h}}_c\}_c$
```

5.6 Experiments

We apply the proposed CDL framework to 1D 1) simulated and 2) real electrophysiological data from the brain, and 2D 3) simulated single molecule localization microscopy (SMLM) data. We use two criteria for evaluation: 1) the accuracy and speed of the CSC step, and 2) the accuracy of the CDU step. For the CSC step, we compare CBP and ADCG to the following convolutional greedy methods: COMP, CMP, COMP-slow, and COMP-INTERP. CMP does not have a projection step. The projection step from COMP-slow inverts $\mathbf{H}\Big|_{t'}^{\mathrm{T}}\mathbf{H}\Big|_{t'}$ directly without Cholesky factorization. Table 5.3 summarizes the similarities and differences of the greedy methods. For the interpolator $\mathbf{f}$, we use a cubic interpolator [150].

	Efficient selection	Efficient projection	Δ_K
COMP	Yes	Yes	Δ
CMP	Yes	·	Δ
COMP-slow	Yes	No	Δ
COMP-INTERP	Yes	Yes	Δ/K

Table 5.3: Convolutional greedy methods.

5.6.1 Simulated 1D electrophysiology dataset

We simulated a signal according to the continuous-time generative model of Eq. 5.2. We used two 10-ms-long gamma-tone templates [131] defined for t from -5 to 5 ms

$$\begin{aligned} h_1(t) &\propto \Big(10^3 t\Big) \exp\Big(- \Big(10^3 t\Big)^2 \Big) \cos\Big(\frac{\pi}{2}\Big(10^3 t\Big)\Big) \\ h_2(t) &\propto \Big(10^3 t\Big) \exp\Big(- \Big(10^3 t\Big)^2 \Big). \end{aligned} \tag{5.29}$$

We assumed the same number of occurrences $N_1 = N_2$ of the templates. We chose the times when events occur uniformly at random, i.e., $\tau_i^c \sim \text{Uniform}[0, T]$ for $c = 1, 2$ and $i = 1, \cdots, N_1$. We chose the amplitude of each event uniformly at random, i.e., $x_i^c \sim \text{Uniform}[1, 2]$. As explained in subsequent sections, we used a range of values for the variables T and N_1. We used a sampling rate $f_s = 10^4$ Hz and obtained the discrete-time signal $\mathbf{y}$ by sampling $y(t)$ at every $\Delta = 10^{-4}$ seconds. We added white Gaussian noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ in discrete time, where σ was set according to a desired Signal-to-Noise ratio (SNR). We obtained $\mathbf{h}_1, \mathbf{h}_2 \in \mathbb{R}^{101}$ by acquiring 101 samples from $h_1(t)$ and $h_2(t)$ and normalizing the resulting vectors to have unit length: $\|\mathbf{h}_1\|_2 = \|\mathbf{h}_2\|_2 = 1$.

5.6.2 Results from electrophysiology simulations: CSC step

We set the sparsity level for greedy methods to be the number of events, and fine-tuned the regularization parameter for CBP (with polar basis) and ADCG to match the same sparsity level. We used the true $\mathbf{h}_1, \mathbf{h}_2$ as dictionary elements.

Sparse-coding computation time We computed the duration of the CSC step, using CBP, ADCG, and the above-mentioned greedy methods, as a function of data length T and total number of $N_{\text{tot.}} = N_1 + N_2$. Specifically, we ran two sets of experiments: 1) $T \in [1, 2, 3, 4, 5]$ s with fixed $N_{\text{tot.}} = 30$ and 2) $N_{\text{tot.}} \in [10, 20, 30, 40, 50]$ with fixed $T = 3$ s. For each experiment, we report durations averaged over 50 independent repeats (trials). Due to relatively low $N_{\text{tot.}}$ compared to T, the data do not contain many occurrences of overlapping filters. Consequently, the selection steps of CMP and COMP take a similar amount of time, which allows us to highlight the importance of the efficient projection step.

Table 5.4 shows the duration of the CSC step for various methods, as a function of T and $N_{\text{tot.}}$. We draw the following conclusions

1. COMP is much faster than CBP, with two possible explanations. In terms of implementation, CBP constructs the full convolutional dictionary, whereas COMP does not. Moreover, it is well-known that greedy methods are faster than BP-like methods [135, 136].

2. COMP is also faster than ADCG. Even though ADCG is as fast as COMP for the greedy selection step, the computational bottleneck is the block coordinate descent algorithm for local optimization, *which is critical for estimating timestamps off the discrete sampling grid*.

3. COMP is as fast as CMP. This is true even for large T or $N_{\text{tot.}}$, which involve a computationally-demanding projection step. This, along with a comparison of COMP to COMP-slow highlights the importance of making the projection step efficient. COMP reduces the computation time of COMP-slow by 48% to 85% on average.

4. As $N_{\text{tot.}}$ increases, the computation time for ADCG and COMP/COMP-INTERP increases, as they rely on greedy algorithms to iteratively identify codes. On the other hand, the computation time of CBP stays constant as a result of ℓ_1 regularization, which identifies the active codes simultaneously.

5. COMP is faster than COMP-INTERP, since the selection step of the former requires C cross-correlation operations, while that of the latter requires KC operations. The computation time of the projection step is the same.

Sparse coding accuracy We computed the *average hit error* [145] for COMP, ADCG, CBP, and for COMP-INTERP as a function of the discretization Δ_K. The *average hit error* measures how far, in terms of absolute displacement, the recovered sparse

	1 s	2 s	3 s	4 s	5 s
COMP	**0.066**	**0.114**	**0.175**	**0.223**	**0.271**
CMP	0.049	0.089	0.131	0.180	0.237
COMP-slow	0.176	0.350	0.545	0.759	1.002
COMP-INTERP	**0.445**	**0.781**	**1.161**	**1.565**	**2.038**
ADCG	11.96	26.77	47.25	78.35	169.64
CBP	13.55	51.75	145.27	·	·

(a) Computation time as a function of T.

	10	20	30	40	50
COMP	**0.096**	**0.204**	**0.267**	**0.340**	**0.438**
CMP	0.082	0.168	0.218	0.276	0.345
COMP-slow	0.194	0.666	1.189	2.111	3.043
COMP-INTERP	**0.744**	**1.416**	**1.811**	**2.377**	**3.112**
ADCG	14.89	27.36	47.60	70.20	82.90
CBP	152.44	134.25	135.10	132.60	133.53

(b) Computation time as a function of $N_{\text{tot.}}$.

Table 5.4: Computation time in seconds.

codes are relative to the true sparse codes. We simulated 50 trials with $T = 1$ s and $N_1 = N_2 = 10$ and computed the median average hit error across trials for each method. The average hit error of COMP, ADCG, and CBP is independent of Δ_K, as they do not rely on grid refinement.

Fig. 5-4 shows that the average hit error for COMP-INTERP is lower than that for COMP. The finer the discretization, the greater the precision with which COMP-INTERP identifies the sparse codes, resulting in lower error. The reduction in error is marginal below a certain threshold ($\Delta_K = \frac{1}{10}\Delta$). The finer the discretization, the more correlated the interpolated templates and, in turn, the columns of the dictionary. Therefore, as discretization becomes smaller, the returns, in terms of average hit error, from expanding the dictionary diminish. With no discretization, COMP and COMP-INTERP achieve the same average hit errors as expected. Fig. 5-4 also shows that there is a small gap, between CBP/ADCG and COMP-INTERP. The difference is due to the fact that COMP-INTERP is not as accurate as CBP/ADCG in the presence of significant template overlaps. Due to its greedy nature, OMP has worse performance

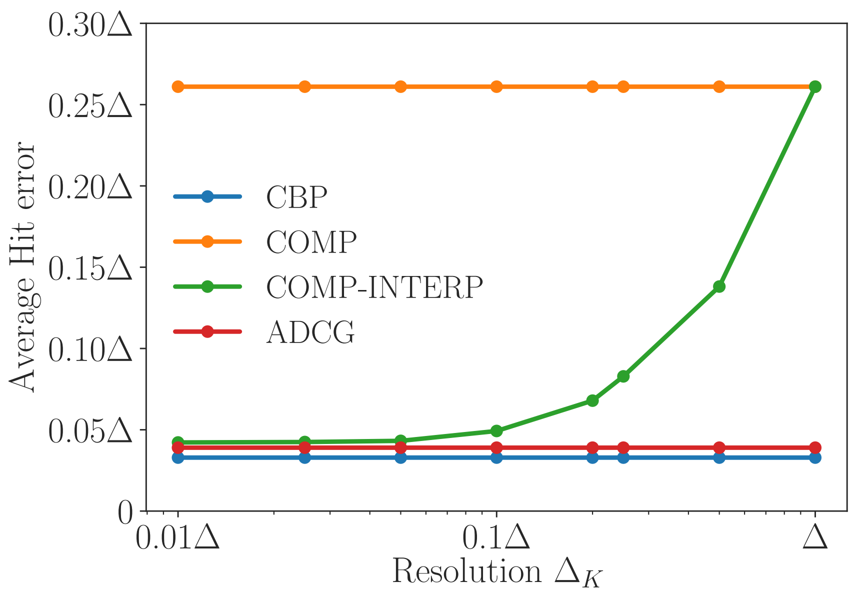

Figure 5-4: Average hit error as a function of Δ_K, on a log scale, for CBP, ADCG, COMP, and COMP-INTERP. Each point represents the *median* average hit error computed across 50 trials of simulated data.

in resolving overlaps than approaches based on basis pursuit [139]. Heuristics [151] can mitigate this.

5.6.3 Results from electrophysiology simulations: CDU step

Dictionary learning accuracy To assess the accuracy of the CDU algorithms, we use the following error distance metric [152] between two templates $\widehat{\mathbf{h}}_c$ and $\widetilde{\mathbf{h}}_c$

$$\mathrm{err}(\widehat{\mathbf{h}}_c, \widetilde{\mathbf{h}}_c) = \sqrt{1 - \langle \widehat{\mathbf{h}}_c, \widetilde{\mathbf{h}}_c \rangle^2}, \text{ for } \|\widehat{\mathbf{h}}_c\|_2 = \|\widetilde{\mathbf{h}}_c\|_2 = 1. \tag{5.30}$$

The lower the metric, the closer $\widehat{\mathbf{h}}_c$ and $\widetilde{\mathbf{h}}_c$. If $\widehat{\mathbf{h}}_c = \mathbf{h}_c^{(s)}$ and $\widetilde{\mathbf{h}}_c = \mathbf{h}_c$, the metric measures how close the learned template at iteration s is to the true template. If $\widehat{\mathbf{h}}_c = \mathbf{h}_c^{(0)}$ and $\widetilde{\mathbf{h}}_c = \mathbf{h}_c^{(*)}$, the metric measures how much the template at convergence has changed from the initial template.

We compared the accuracy of the CDU step with the interpolated and the original dictionary. We simulated 50 trials of data, with $T = 5$ s and $N_1 = N_2 = 150$, for several levels of SNR. We obtained the initial templates $\mathbf{h}_1^{(0)}$ and $\mathbf{h}_2^{(0)}$ by perturbing the original

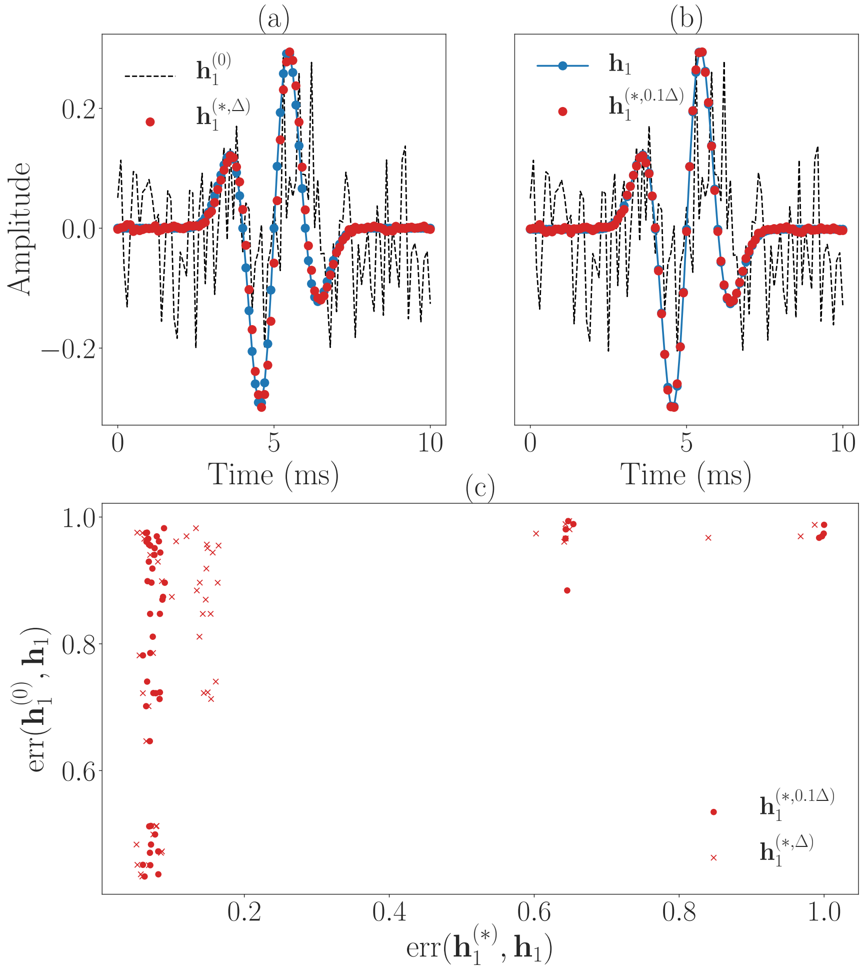

Figure 5-5: Illustration of the initial $\mathbf{h}_1^{(0)}$, learned $\mathbf{h}_1^{(*)}$, and the true template $\mathbf{h}_1$. (a) $\mathbf{h}_1^{(*,\Delta)}$, (b) $\mathbf{h}_1^{(*,0.1\Delta)}$. (c) $\mathrm{err}(\mathbf{h}_1^{(*)}, \mathbf{h}_1)$ plotted against the initial $\mathrm{err}(\mathbf{h}_1^{(0)}, \mathbf{h}_1)$ for SNR=10 dB.

templates $\mathbf{h}_1$ and $\mathbf{h}_2$ with varying levels of Gaussian noise, such that $\mathrm{err}(\mathbf{h}_c^{(0)}, \mathbf{h}_c) \geq 0.5$. We performed 10 iterations of the alternating-minimization algorithm for the following methods: 1) COMP & CDU with Δ and 2) COMP-INTERP & CDU-INTERP with $\Delta_K = 0.1\Delta$, converging to $\{\mathbf{h}_c^{(*,\Delta)}\}_c$ and $\{\mathbf{h}_c^{(*,0.1\Delta)}\}_c$, respectively.

Fig. 5-5 shows $\{\mathbf{h}_c^{(*,\Delta)}\}_c$ and $\{\mathbf{h}_c^{(*,0.1\Delta)}\}_c$ at convergence. A visual inspection of one of the trials (Figs. 5-5(a, b)) shows that $\mathbf{h}_1^{(*,0.1\Delta)}$ matches the true $\mathbf{h}_1$, whereas $\mathbf{h}_1^{(*,\Delta)}$ learns a shape that disagrees with the true $\mathbf{h}_1$ in the middle portion. Fig. 5-5(c) shows that even with high initialization distance $\mathrm{err}(\mathbf{h}_1^{(0)}, \mathbf{h}_1)$ (close to 1), $\mathrm{err}(\mathbf{h}_1^{(*)}, \mathbf{h}_1)$ at convergence is close to 0 in general. On average, we observe that $\mathrm{err}(\mathbf{h}_1^{(*,0.1\Delta)}, \mathbf{h}_1)$

is lower than $\text{err}(\mathbf{h}_1^{(*,\Delta)}, \mathbf{h}_1)$, which highlights the importance of the interpolated dictionary. Although not shown here, we observed that $\text{err}(\mathbf{h}_2^{(*,0.1\Delta)}, \mathbf{h}_2) < \text{err}(\mathbf{h}_2^{(*,\Delta)}, \mathbf{h}_2)$ on average.

5.6.4 Real 1D electrophysiology dataset: spike sorting

We applied our framework to spike sorting. Given a recording of extracellular voltage, the goal of spike sorting is to learn the action potentials (templates) from neurons (sources), and the times when they occur [43]. We used a dataset that consists of an extracellular recording from the rat hippocampus, along with a simultaneous intracellular recording [153] from one neuron. The intracellular recording provides the *ground truth* data, as it unequivocally associates an action potential to a single neuron, and thus enables us to evaluate the accuracy of the CDL frameworks. For this dataset, 620 events occurred from the neuron that was recorded intracellularly. The sampling rate of the extracellular data, which comprise 4 channels, is $f_s = 10^4$ Hz. We used $T = 150$ seconds of data from channel 1 and preprocessed them, following standard procedures [12]. Specifically, we applied a high-pass filter with cut-off frequency 400 Hz, and whitened the data. In addition, we identified peaks that crossed a pre-defined threshold [154] and extracted a segment of length 81 samples centered around each peak. The resulting collection $\mathbf{Y} \in \mathbb{R}^{5000\times 81}$ of $5,000$ segments is the input to our analyses of the real data.

Method setup We assumed $C = 3$, namely that the extracellular recording can detect activity from 3 neurons. We used templates $\{\mathbf{h}_c\}_{c=1}^{3} \in \mathbb{R}^{41}$, each of length 4 ms. We applied, to the extracellular data, 1) COMP & CDU with Δ to obtain $\{\mathbf{h}_c^{(*,\Delta)}\}_{c=1}^{3}$ and 2) COMP-INTERP & CDU-INTERP with $\Delta_K = 0.1\Delta$ to obtain $\{\mathbf{h}_c^{(*,0.1\Delta)}\}_{c=1}^{3}$. We used the following procedure to initialize the templates to $\left\{\mathbf{h}_c^{(0)}\right\}_{c=1}^{3}$. Following segment extraction, we first performed PCA on $\mathbf{Y}$ for dimensionality reduction, and then K-means clustering with three clusters in the lower-dimensional space. We used the centroids of the clusters to obtain $\left\{\mathbf{h}_c^{(0)}\right\}_{c=1}^{3}$. We used an estimate of the variance of the background noise as the termination criterion for the CSC step of COMP and COMP-INTERP. We computed this estimate by extracting data from a segment that remained below a pre-defined threshold for more than 500 ms. We ran 15 iterations of the CDL algorithm.

We compared CBP and ADCG with $\{\mathbf{h}_c^{(0)}\}_{c=1}^3$ to the CSC step of COMP with $\{\mathbf{h}_c^{(*,\Delta)}\}_{c=1}^3$ and COMP-INTERP with $\{\mathbf{h}_c^{(*,0.1\Delta)}\}_{c=1}^3$. We used $\{\mathbf{h}_c^{(0)}\}_{c=1}^3$ for CBP and ADCG, not the learned templates, to emphasize that these frameworks lack the means of updating the initial filters. For a given true spike event from the intracellular data, we associate an event identified using the extracellular data as a *true positive* if the event is within 30 samples (3 ms) of a true event from the intracellular data. Among the learned templates, we associate the template with the highest number of true positive with the neuron from the intracellular data and refer to it as $\mathbf{h}_1$. As we do not have access to the true $\mathbf{h}_1$, we treat $\mathbf{h}_1^{(*,\Delta)}$, $\mathbf{h}_1^{(*,0.1\Delta)}$, or $\mathbf{h}_1^{(0)}$, depending on the CSC approach, as the best estimate of the true spike template. Since ADCG requires values of the templates off the grid, we performed cubic interpolation on each of the filters. We approximated the gradients using finite differences of the interpolated template values.

5.6.5 Results from spike sorting application

Detection error curve We used two statistics to evaluate the spike sorting performance. Following CSC, a threshold is set to identify the times when action potentials occur. A *true miss* is a true spike from the intracellular data within 3 ms of which no threshold-crossing event occurs in the extracellular data. A *false alarm* is a threshold-crossing event from the extracellular data that is not a true spike. Varying the threshold leads to a trade-off between true misses and false alarms. A high-amplitude threshold leads to a low number of false alarms and a large number of true misses, and vice versa for a low-amplitude threshold.

Fig. 5-6 shows the result of sorting spikes associated with $\mathbf{h}_1$ using CBP, ADCG, COMP and COMP-INTERP. The figure shows that the greedy approaches rival CBP and ADCG, and are better in the low true miss regime. The number of true misses for CBP and ADCG do not decrease below 27 and 7, respectively, even with thresholds of low amplitude. This indicates that CBP and ADCG are not able to identify a subset

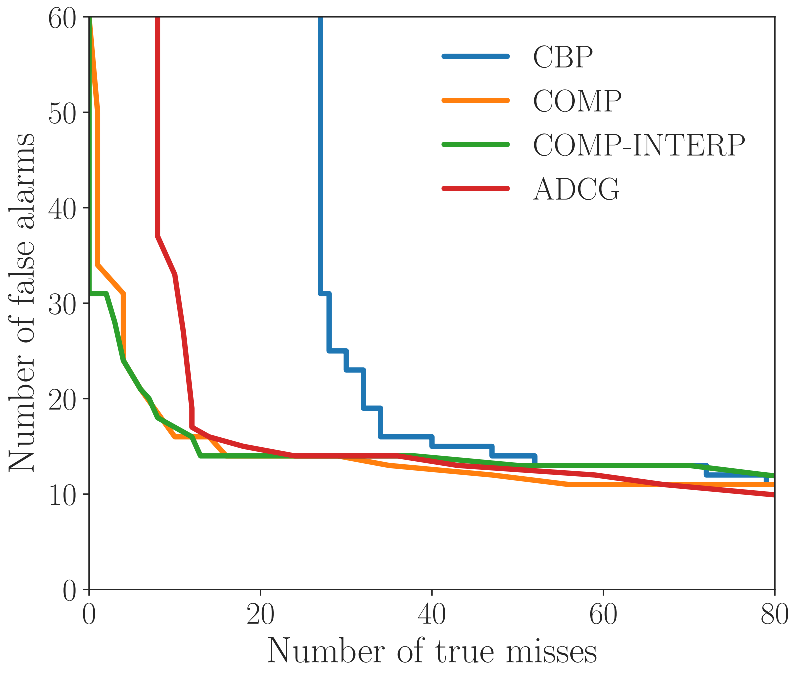

Figure 5-6: Error curves for events associated with $\mathbf{h}_1$ (true spike template). The curves, computed for CBP, ADCG, COMP and COMP-INTERP, show the trade-off between the number of false alarms and the number of true misses for each method.

of true events that the greedy approaches correctly identify.

Difference between ℓ_0 and ℓ_1 The discrepancy in the number of true misses from CBP and COMP/COMP-INTERP motivated us to examine segments for which the number of errors from CBP and the greedy methods differ. Fig. 5-7 shows examples of such segments. Fig. 5-7(a) shows that CBP fails to capture the true spike event (red dot), resulting in a true miss, whereas COMP-INTERP (Fig. 5-7(b)) uses $\mathbf{h}_1$ to correctly identify the event. The failure of CBP and the success of COMP-INTERP point to a key difference between the ℓ_1 and ℓ_0-based CSC step. To minimize its objective function, ℓ_1-based CBP must strike a balance between the reconstruction error, which can be reduced by using additional templates, and the ℓ_1 penalty, which can be reduced by using fewer templates or ones with lower amplitude. In Fig. 5-7(a), the choice of regularization parameter λ is such that CBP chooses to use one template with large amplitude, thereby missing the true event. Although a smaller λ can lower the number of true misses, this would result in spurious events that would increase the number of false alarms. This points to a limitation of ℓ_1-based methods, the need to tune λ carefully. COMP-INTERP (and COMP), on the other hand, can select as many events as needed to make the reconstruction error below the error threshold. In Fig. 5-7(b), COMP-INTERP first selects $\mathbf{h}_3$, and then $\mathbf{h}_1$, which corresponds to the true event.

Example of a non-integer shift Fig. 5-7(c) is an example of a segment where COMP results in a false alarm, but COMP-INTERP does not (Fig. 5-7(d)). COMP is forced to use the true spike template, $\mathbf{h}_1$, whereas COMP-INTERP uses $\mathbf{h}_3$ to select the secondary peak. This highlights the benefits of using the interpolated dictionary. That being said, we observe from Fig. 5-6 that these two have similar performance, with COMP-INTERP slightly outperforming in the low true miss regime.

Learned templates Fig. 5-8 shows the templates that were learned by CDL using the extracellular data, and COMP-INTERP in the CSC step. We observed that $\mathbf{h}_c^{(*,\Delta)}$ and $\mathbf{h}_c^{(*,0.1\Delta)}$ are nearly identical, possibly due to the high sampling rate of 10^4 Hz. The fact that the shapes of the learned templates are not significantly different from those of the initial templates suggests that we initialized the templates well. To determine how different the learned templates are from the initial ones, we computed $\text{err}(\mathbf{h}_c^{(0)}, \mathbf{h}_c^{(*,\Delta)})$ for $c = 1, 2, 3$, and then took the maximum of the three values. We found that the maximum equaled 0.32, which indicates that, although not obvious visually, the CDL algorithm did learn new templates.

5.6.6 Simulated 2D dataset: single molecule localization microscopy (SMLM)

We also applied our framework to SMLM [44]. The task is to accurately localize the locations of activated single molecules given a series of 2D low-resolution images taken by an optical microscope. Specifically, given images of size 64×64 pixels, with each pixel corresponding to an area of size 100 nm × 100 nm, the goal is to determine the location of molecules with up to 10 nm accuracy, a task referred to as *super-resolution microscopy*. The images are assumed to be generated through a two-dimensional convolutional generative model, where a point spread function (PSF) is convolved with

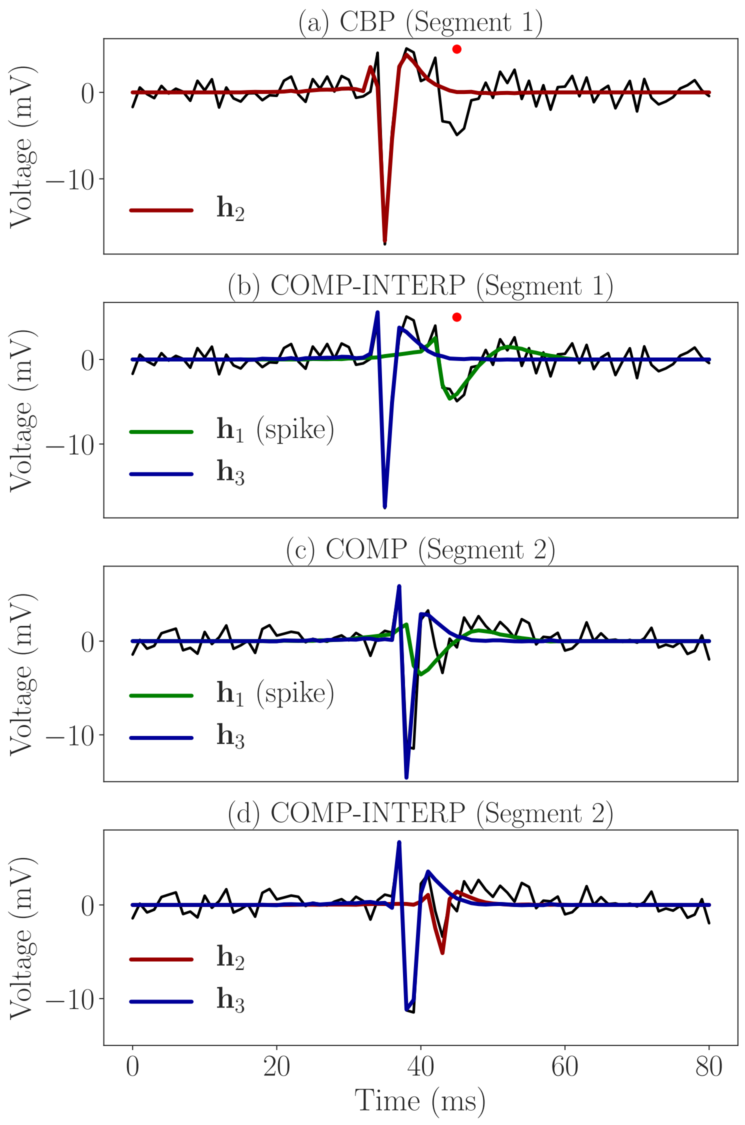

Figure 5-7: Example of applying CBP, COMP, and COMP-INTERP to segments of real data (black trace). The red dot shows where the true event occurs. The green, red, and blue traces are reconstructions of the segments using only $\mathbf{h}_1$ (true spike template), $\mathbf{h}_2$ and $\mathbf{h}_3$, respectively. (a) A segment where CBP fails to correctly identify the occurrence of an event, and (b) COMP-INTERP does. (c) A different segment where COMP incorrectly uses $\mathbf{h}_1$, but (d) COMP-INTERP does not.

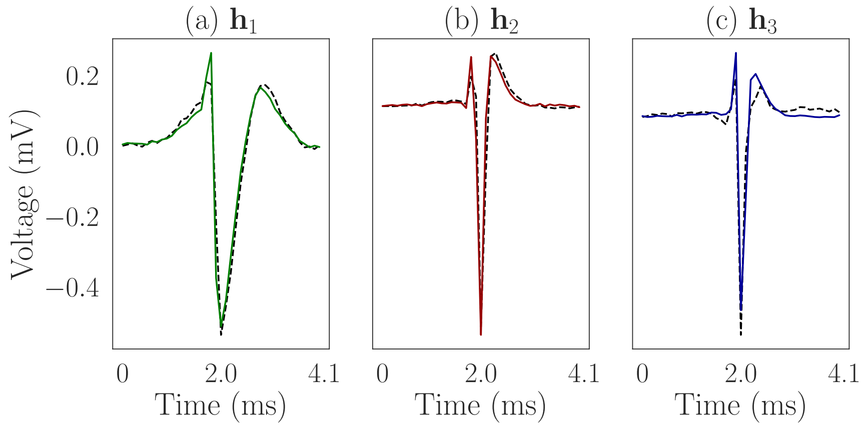

Figure 5-8: Initial $\mathbf{h}_c^{(0)}$ (black dashed trace), and learned $\mathbf{h}_c^{(*,\Delta)}$ (colored trace). The values of the metric $\text{err}(\mathbf{h}_c^{(0)}, \mathbf{h}_c^{(*,\Delta)})$ for $c = 1, 2, 3$ were 0.21, 0.29, and 0.32, respectively.

sparse codes (the locations of molecules) and perturbed by photon noise. Fig. 5-9(a) shows an example of SMLM data.

We use a publicly-available simulated dataset, where a symmetric 2D Gaussian template with FWHM = 723 nm, equivalent to $\sigma = 110$ nm, was used for simulation. Fig. 5-9(b) shows a sampled template with the ground truth $\sigma = 110$ nm. We used $J = 100$ image frames for the CDL task, equivalent to $\mathbf{Y} \in \mathbb{R}^{64\times 64\times 100}$. The number of single molecules and their locations are different across the frames. With $J = 100$, we have approximately 700 occurrences of the single molecules.

Method setup We assumed $C = 1$ and used $\mathbf{h}_1 \in \mathbb{R}^{9\times 9}$, equivalent to a patch of size 900 nm ×900 nm. We define additional notations as follows. Let $\mathbf{h}_1[u, v]$ denote element (u, v) of $\mathbf{h}_1$. We initialized $\mathbf{h}_1^{(0)}$ to a symmetric Gaussian template with $\sigma = 200$ nm (Fig. 5-9(c)), which is wider than the true template ($\sigma = 110$ nm). For the CSC step, we terminated COMP/COMP-INTERP when the residual norm $\|\mathbf{r}\|_2$, which decreases with correct identification of the codes at each iteration, began to increase. As in the earlier applications, we used two versions of the CDL, 1) COMP & CDU with a single pixel resolution and 2) COMP-INTERP & CDU-INTERP with 0.1 sub-pixel resolution. A sub-pixel resolution of 0.1 indicates that the method has access to a total of 100 non-integer shifts (a combination of 10 non-integer delays in the horizontal and vertical directions). We ran 5 iterations of the CDL algorithm to obtain $\mathbf{h}_1^{(*,\Delta)}$ and $\mathbf{h}_1^{(*,0.1\Delta)}$, respectively.

5.6.7 Results from SMLM application

Learned templates Figs. 5-9(d-f) show the learned filters. Compared to the initial Gaussian template (Fig. 5-9(c)), $\mathbf{h}_1^{(*,0.1\Delta)}$ (Fig. 5-9(d)) is much narrower and resembles the true template in Fig. 5-9(b). Although not shown here, $\mathbf{h}_1^{(*,\Delta)}$ is visually similar to $\mathbf{h}_1^{(*,0.1\Delta)}$. Since the true filter is a *symmetric* template, with rows equidistant from the center row having the same shape, we expect to see this property for the learned filters as well. To analyze this, we define a distance metric $\text{dist}(\mathbf{h}, i)$ as follows

$$\begin{aligned}&\text{dist}(\mathbf{h}, i) = \left(\sum_{v=1}^{V} \left(\mathbf{h}[m+i, v] - \mathbf{h}[m-i, v]\right)^2\right)^{0.5} \\ &\text{for } i = 0, \cdots, m-1, \text{ and } \|\mathbf{h}\|_2 = 1,\end{aligned} \tag{5.31}$$

where m is the index for the center row ($m = 5$ and $V = 9$ for this application) and i represents the offset from the center row. The closer the metric is to 0, the more *symmetric* the template. Fig. 5-9(f) shows this metric for $\mathbf{h}_1^{(0)}$, $\mathbf{h}_1^{(*,\Delta)}$, and $\mathbf{h}_1^{(*,0.1\Delta)}$. We observe that $\text{dist}(\mathbf{h}_1^{(*,0.1\Delta)}, i)$ is lower than $\text{dist}(\mathbf{h}_1^{(*,\Delta)}, i)$ for all offsets, with the largest deviations coming from $i = 1, 2$. This suggests that CDU-INTERP preserves the symmetry of the initial template much better than CDU. The difference comes from the *alignment* operation that implicitly happens in CDU-INTERP, as previously discussed. CDU-INTERP shifts/aligns the extracted patches. On the other hand, simple CDU takes a weighted average of the extracted patches, losing information on the symmetry of the filter.

We emphasize that the existing sparse approximation approaches for SMLM [133], [146], [155] assume access to the ground-truth template. Our framework learns the optimal template from the data and obviates the need for this assumption.

Performance curve We used two metrics to evaluate localization accuracy. First, we used the F-index, given as the harmonic mean of the precision and recall, and a common measure of localization accuracy [44]. The F-index is a function of the radius of tolerance. For a given radius r, if the distance between a true and a recovered code is less than r, the recovered code is considered a *true positive.* The larger the radius, the larger the number of true positives, and the higher the F-index. Second,

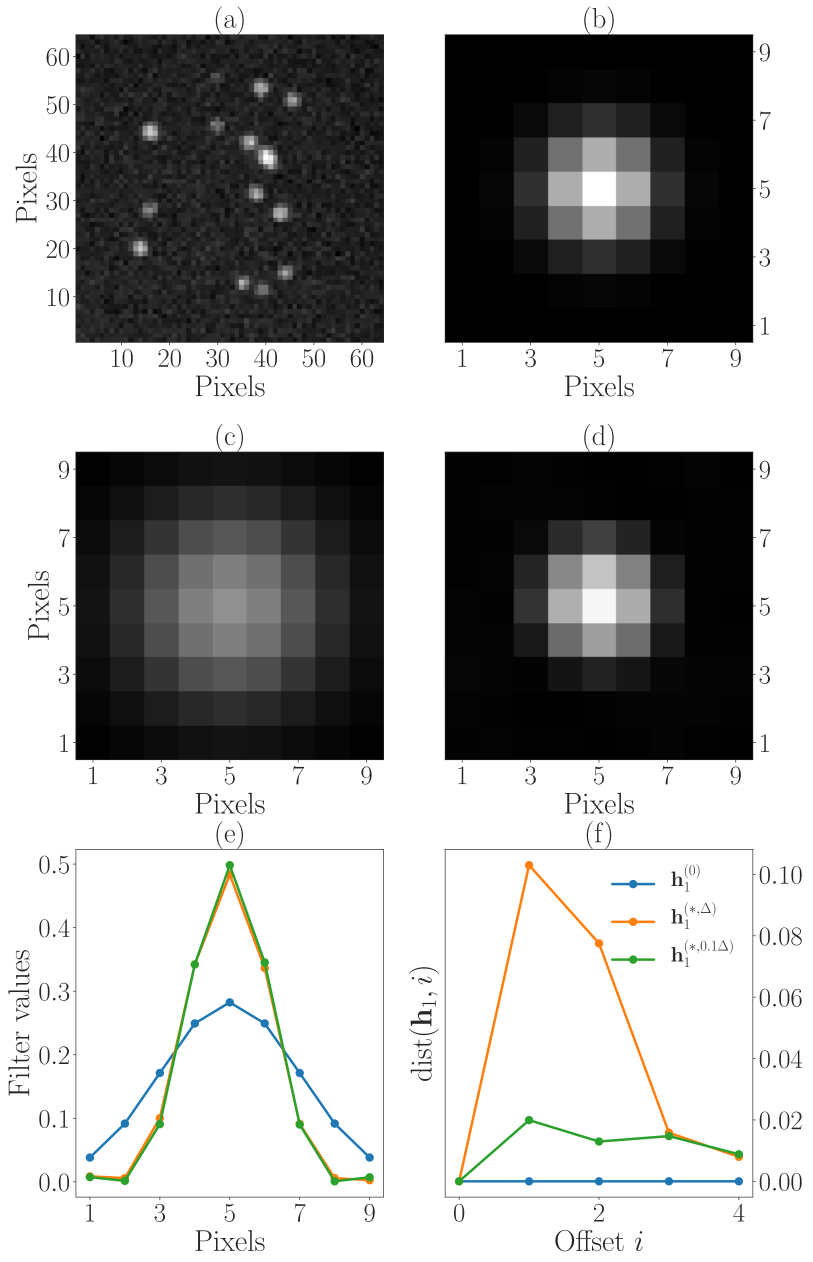

Figure 5-9: SMLM application. (a) A single 64×64 frame of SMLM data with 8 molecules, with 100 nm $\times$100 nm /pixel. (b) A Gaussian template with ground truth $\sigma = 110$ nm. (c) $\mathbf{h}_1^{(0)}$ with large σ. (d) $\mathbf{h}_1^{(*,\Delta)}$ for COMP & CDU with single pixel resolution. (e) Center row of $\mathbf{h}_1^{(0)}$ (blue), $\mathbf{h}_1^{(*,\Delta)}$ (orange), and $\mathbf{h}_1^{(*,0.1\Delta)}$ (green). (f) Metric $\text{dist}(\mathbf{h}_1, i)$ defined in Eq. 5.31.

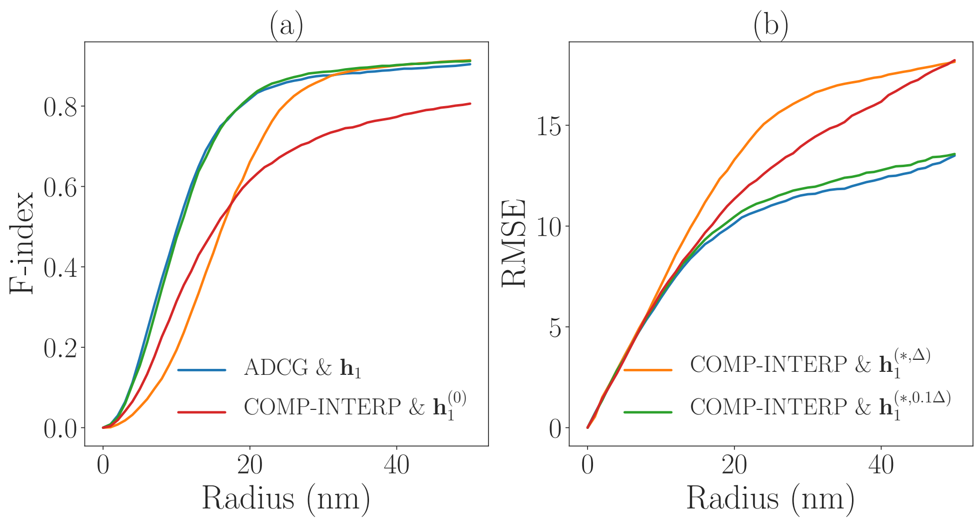

Figure 5-10: SMLM performance for ADCG (blue), COMP-INTERP with $\mathbf{h}_1^{(0)}$ (red), COMP-INTERP with $\mathbf{h}_1^{(*,\Delta)}$ (orange), and COMP-INTERP with $\mathbf{h}_1^{(*,0.1\Delta)}$ (green). (a) F-index (b) RMSE.

we used RMSE, given as a root mean-squared error between the true locations and the true-positive locations. RMSE also depends on the radius. The lower the RMSE, the closer the true-positive codes are to the true codes.

Fig. 5-10 shows the F-index (Fig. 5-10(a)) and the RMSE (Fig. 5-10(b)) for ADCG with the ground-truth filter, COMP-INTERP with $\mathbf{h}_1^{(0)}$, COMP-INTERP with 0.1 sub-pixel resolution and $\mathbf{h}_1^{(*,\Delta)}$, and 4) COMP-INTERP with 0.1 sub-pixel resolution and $\mathbf{h}_1^{(*,0.1\Delta)}$. We observe that both ADCG and COMP-INTERP with $\mathbf{h}_1^{(*,0.1\Delta)}$ achieve the best performance in terms of F-index and RMSE. Although COMP-INTERP with $\mathbf{h}_1^{(*,\Delta)}$ eventually achieves a similar F-index with increasing radius, the wide gap in F-index, ranging for radii between 0 to 30 nm suggests that the recovered codes are located further away from the true codes, compared to those recovered by the best two methods. This is further supported by the much higher RMSE value for COMP-INTERP with $\mathbf{h}_1^{(*,\Delta)}$. Finally, COMP-INTERP with $\mathbf{h}_1^{(0)}$ suggests that mis-specifying the dictionary leads to poor performance in terms of both metrics. In conclusion, 1) the difference between $\mathbf{h}_1^{(*,0.1\Delta)}$ and $\mathbf{h}_1^{(*,\Delta)}$ and 2) the performance gap between COMP-INTERP with $\mathbf{h}_1^{(*,0.1\Delta)}$ and the other greedy methods demonstrate the importance of CDL with interpolation.

5.7 Discussion

We introduced novel Convolutional Sparse Coding (CSC) and Convolutional Dictionary Update (CDU) algorithms using discrete samples of continuous-domain signals that consist of shifted copies from multiple sources, each with its template.

For CSC, unlike existing methods to estimate events that occur on the continuous domain, and off the grid, our algorithm operates on a refined discrete sampling grid, with tunable precision. To overcome the lack of access to values of the templates or the signal outside the original grid, we construct an expanded, overcomplete dictionary that comprises discrete shifts of the original templates, along with non-integer shifts aligned with the refined grid, and obtained by smooth interpolation. The expanded dictionary increases the computational demands of CSC. To mitigate this, we focus on greedy pursuit methods and proposed an efficient implementation of convolutional OMP (COMP), which forms the basis of an efficient implementation of COMP with the interpolated dictionary (COMP-INTERP). The efficient COMP exploits the locality of the templates. It is faster than continuous basis pursuit [131] and greedy continuous frameworks [132], [133, 134], which are grid-free CSC algorithms, and achieves competitive coding accuracy. We demonstrated this empirically in an application to 1D real and simulated electrophysiology data, as well as 2D super-resolution microscopy.

For CDU, we proposed a novel algorithm to update the templates while accounting for non-integer delays. Empirically, we showed that accounting for non-integer delays in the CDU step leads to more accurate dictionary learning and better code identification than otherwise. In an application to super-resolution microscopy [44], we demonstrated that our algorithm can automatically learn the underlying point-spread function, obviating the current need to assume it is known.

We conclude that our approach is a simple, yet efficient paradigm for convolutional dictionary learning, that faithfully accounts for the continuous nature of the domain of signals.

Chapter 6

Smoooth convolutional dictionary learning with GP constraint

Primer In the previous chapter, we performed smooth convolutional dictionary learning in a favorable setting of high signal-to-noise ratio (SNR). The dictionary (the collection of the templates) is usually learned in a data-driven manner, without constraints. In this chapter, we have the same goal of learning smooth convolutional dictionary and their corresponding codes, but in the low SNR setting. To circumvent the problem of overfitting in the low SNR setting, we introduce explicit smoothness constraint via Gaussian Process (GP) prior on the dictionary elements. This chapter is adapted from the following work

- Andrew H. Song, Bahareh Tolooshams, and Demba Ba, *Gaussian Process Convolutional Dictionary Learning*, IEEE Signal Processing Letters, 2021

6.1 Introduction

In practice, when data are scarce or have a low signal-to-noise ratio (SNR), learned dictionaries overfit the data in the absence of constraints. Consequently, the interpretability of the dictionary and its predictive performance on unobserved data suffer. The problem is aggravated for data from non-Gaussian distributions such as binomial data, due to the non-linear mapping from dictionary to observations [156]. There is also evidence that the templates for naturally-occurring data could be considered *smooth* [157, 43].

The recent literature suggests that there are several approaches to learning smooth shift-invariant templates. One approach models the templates with parametric functions, such as the bi-exponential [51] function or a mixture of Gaussians [158]. Another line of work imposes total variation or Tikhonov-like penalties [159, 160, 161] on the templates. More recently, smooth templates were obtained by passing learned dictionary through pre-designed lowpass filters [157, 13, 162].

We propose an alternative flexible, nonparametric approach, by assuming that the templates are generated from a Gaussian Process (GP) [46]. We make the following contributions[1]

CDL via GP regularization We introduce GPCDL, a framework for CDL with GP regularization, which can be applied to observations from the natural exponential family [163]. We show that the learned dictionary is accurate in conditions where the unregularized alternatives overfit. The learning procedure is a simple extension of iteratively reweighted least squares and allows us to easily incorporate the GP prior.

GP prior as Wiener filter We show that, under some assumptions, the GP prior acts as a lowpass Wiener filter [164], which allows GPCDL to learn smooth dictionaries. From this unique perspective, we elucidate the trade-off between the amount of training data and the parameters of the GP prior.

The paper is organized as follows: Section 6.2 and 6.3 introduce the background and the GPCDL framework. Section 6.4 develops the interpretation of GPCDL as Wiener filtering. In Section 6.5 and 6.6, the results and conclusion are presented.

[1]The code can be found at https://github.com/andrewsong90/gpcdl

6.2 Background

6.2.1 Notation

We denote the zero and identity matrices as $\mathbf{0}$ and $\mathbf{I}$, with appropriate dimensions. $\mathbf{A}_{(k,k')}$ refers to the entry of matrix $\mathbf{A}$ at location (k, k'). The diag$(\cdot)$ refers to a diagonal matrix, with entries equal to the vector argument. When applied to a vector, a function operates in an element-wise manner.

6.2.2 Natural exponential family

Let $\mathbf{y}^j \in \mathbb{R}^N$ be observations from the natural exponential family with mean $\boldsymbol{\mu}^j = \mathbb{E}[\mathbf{y}^j]$, for $j = 1, \ldots, J$. With $\mathbf{1}_N$ as the N-length vector of ones, the log-likelihood is given as

$$\log \ell(\mathbf{y}^j) = \frac{f(\boldsymbol{\mu}^j)^{\mathrm{T}}\mathbf{y}^j - \mathbf{1}_N^{\mathrm{T}} b(f(\boldsymbol{\mu}^j))}{\phi} + c(\mathbf{y}^j, \phi), \tag{6.1}$$

where ϕ is a dispersion parameter and the functions $b(\cdot)$, $c(\cdot)$, as well as the invertible link $f(\cdot)$, are distribution-dependent.

We consider $f(\boldsymbol{\mu}^j)$ to be the sum of scaled and time-shifted copies of C finite-length templates $\{\mathbf{h}_c\}_{c=1}^C \in \mathbb{R}^K$, each localized, i.e., $K \ll N$. We express $f(\boldsymbol{\mu}^j)$ as a convolution, i.e., $f(\boldsymbol{\mu}^j) = \sum_{c=1}^C \mathbf{h}_c * \mathbf{x}_c^j + \mathbf{a}^j$, where the *code vector* $\mathbf{x}_c^j \in \mathbb{R}^{N-K+1}$ is a train of scaled impulses and $\mathbf{a}^j \in \mathbb{R}^N$ is a baseline. The entry of $\mathbf{x}_c^j$ at index $n_{c,i}^j$ corresponds to the location of the i^{th} event with amplitude $x_{c,i}^j$. Alternatively, we can write $f(\boldsymbol{\mu}^j) - \mathbf{a}^j = \sum_c \mathbf{X}_c^j \mathbf{h}_c = \sum_c \sum_{i=1}^{N_c^j} x_{c,i}^j \mathbf{S}_{c,i}^j \mathbf{h}_c$, where $\mathbf{S}_{c,i}^j = [\mathbf{0}_{K\times(n_{c,i}^j-1)}\ \mathbf{I}_{K\times K}\ \mathbf{0}_{K\times(N-K-n_{c,i}^j+1)}]^{\mathrm{T}} \in \mathbb{R}^{N\times K}$ is the linear operator that shifts $\mathbf{h}_c$ by $n_{c,i}^j$ samples and N_c^j is the number of occurrences of $\mathbf{h}_c$ in $\mathbf{y}^j$ [13].

6.2.3 Gaussian Process

Gaussian Processes (GPs) offer a nonparameteric and flexible Bayesian approach for signal modeling [46], which we use as a smooth prior on $\mathbf{h}_c$. We first define functions $h_c : [0, T) \to \mathbb{R}$, $\forall c$, generated from a GP prior with zero-mean and stationary kernel $\kappa_c(t,t')$, i.e., $h_c(t) \sim \text{GP}(0, \kappa_c(t,t')), \forall c$. We assume that the filter $\mathbf{h}_c$ is sampled from $h_c(\cdot)$, and for simplicity, with constant sampling interval Δ such that $T = K\Delta$. This yields $\mathbf{h}_c \sim \mathcal{N}(\mathbf{0}, \Sigma_c)$, where $\Sigma_c \in \mathbb{R}^{K\times K}$ is the covariance matrix and $\Sigma_{c,(k,k')} = \kappa_c(k\Delta, k'\Delta)$.

We focus on kernels in the *Matern* family [165], parameterized by ν, variance σ_c^2, and lengthscale $\l_c$. The parameter ν controls the smoothess of the kernel and is defined *a priori* by the user. The popular choice is $\nu = p + 1/2$, $p \in \mathbb{N}^+$, since this leads to simplification of the kernel expression. The parameters σ_c^2 and l_c can be chosen by maximum-likelihood estimation or cross-validation [46].

The power spectral density (PSD) of the kernel, denoted $\gamma_c(\omega)$, a function of the normalized frequency $\omega \in [-\pi, \pi]$, is obtained by taking the Fourier transform of the kernel [166]. We focus on $\nu = 1.5$ throughout this work, noting that the same holds for any other GP kernels. For $\nu = 1.5$, we have

$$\begin{aligned}\Sigma_{c,(k,k')} &= \sigma_c^2\Big(1 + \frac{\sqrt{3}(k-k')\Delta}{l_c}\Big)\exp\Big(-\sqrt{3}\frac{(k-k')\Delta}{l_c}\Big)\\ \gamma_c(\omega) &= (4/\sqrt{3})\sigma_c^2 l_c/(1 + l_c^2\omega^2/3)^2.\end{aligned}$$

An example of $\gamma_c(\omega)$ for $\nu = 1.5$ is depicted in Fig. 6-1(a) for varying l_c. As ω increases, $\gamma_c(\omega)$ decays monotonically.

6.3 CDL with GP regularization

6.3.1 Objective

Combining the log-likelihood $\log p(\mathbf{y}|\{\mathbf{h}_c\})$, where we use $\mathbf{y}$ to denote $\{\mathbf{y}^j\}$, and the log-prior, we cast the GPCDL problem as minimizing the negative *log-posterior* $\mathcal{L}(\mathbf{y})$,

$$\min_{\substack{\{\mathbf{h}_c\}_{c=1}^C \\ \{\mathbf{x}_c^j\}_{c=1,j=1}^{C,J}}} \overbrace{\underbrace{\sum_j \frac{-f(\boldsymbol{\mu}^j)^{\mathrm{T}}\mathbf{y}^j + \mathbf{1}_N^{\mathrm{T}} b(f(\boldsymbol{\mu}^j))}{\phi}}_{-\log p(\mathbf{y}|\{\mathbf{h}_c\})} + \sum_c \frac{\mathbf{h}_c^{\mathrm{T}}\Sigma_c^{-1}\mathbf{h}_c}{2}}^{\mathcal{L}(\mathbf{y})} \tag{6.2}$$
$$\text{s.t.} \quad \|\mathbf{x}_c^j\|_0 < \beta \text{ and } \|\mathbf{h}_c\|_2 \le 1, \forall j, c.$$

We use the ℓ_0 pseudo-norm for the sparsity constraint (number of nonzeros), with sparsity level β. The GP prior is incorporated as a quadratic regularizer on $\mathbf{h}_c$. This formulation can be naturally extended to the multivariate setting.

Parametric approaches express $\{\mathbf{h}_c\}$ as combinations of parametric functions [51, 158]. Despite requiring few parameters, these approaches require a careful choice of functions and parameters (e.g., the number of functions) to minimize model misspecification error. GPCDL is a nonparametric approach and avoids the misspecification issue at the expense of more parameters, i.e., the templates. By imposing structure on $\{\mathbf{h}_c\}$ with the GP prior, GPCDL promotes smooth templates, while maintaining the flexibility of the nonparametric paradigm.

We use alternating minimization to solve Eq. (6.2), where $\mathcal{L}(\mathbf{y})$ is minimized with respect to $\{\mathbf{h}_c\}$ and $\{\mathbf{x}_c^j\}$, by alternating between a convolutional sparse coding (CSC) step, optimizing for $\{\mathbf{x}_c^j\}$, and a convolutional dictionary update (CDU) step, optimizing for $\{\mathbf{h}_c\}$ [167]. For CSC, we use Convolutional Orthogonal Matching Pursuit (COMP) [168, 59], a greedy algorithm that iteratively identifies the template and the code that minimize $-\log p(\mathbf{y}|\{\mathbf{h}_c\})$. We define β as the *minimal* active number of elements which, when reconstructed in the form of $\boldsymbol{\mu}$, results in $-\log p(\mathbf{y}|\{\mathbf{h}_c\})$ lower than a threshold computed from the baseline period of each dataset. More details can be found in [59].

6.3.2 Convolutional Dictionary Update

Given the estimates for $\mathbf{X}_c^j$, we use Newton's method to minimize $\mathcal{L}(\mathbf{y})$ with respect to $\mathbf{h}_c$, referred to, in the context of the exponential family, as iteratively reweighted least squares (IRLS) [36]. At iteration t, we compute its gradient and Hessian

$$\begin{aligned}\nabla_{\mathbf{h}_c}\mathcal{L}(\mathbf{y}) &= -\phi^{-1}\sum_j (\mathbf{X}_c^j)^{\mathrm{T}}(\mathbf{y}^j - \boldsymbol{\mu}^{j,(t)}) + \Sigma_c^{-1}\mathbf{h}_c^{(t)}, \qquad (6.3)\\ \nabla^2_{\mathbf{h}_c}\mathcal{L}(\mathbf{y}) &= \phi^{-1}\sum_j (\mathbf{X}_c^j)^{\mathrm{T}}\,\mathrm{diag}((f'(\boldsymbol{\mu}^{j,(t)}))^{-1})\mathbf{X}_c^j + \Sigma_c^{-1},\end{aligned}$$

where f' denotes the derivative of f. We briefly discuss how the gradient and the Hessian were obtained - For notational simplicity, we drop dependence on j and t. The gradients of the first and the last term of $\mathcal{L}(\mathbf{y})$, which are $-\nabla_{\mathbf{h}_c} f(\boldsymbol{\mu})^{\mathrm{T}}\mathbf{y}$ and $\nabla_{\mathbf{h}_c}\mathbf{h}_c^{\mathrm{T}}\Sigma_c^{-1}\mathbf{h}_c$ respectively, are given as follows

$$\begin{aligned} -\nabla_{\mathbf{h}_c} f(\boldsymbol{\mu})^{\mathrm{T}}\mathbf{y} &= -(\mathbf{X}_c)^{\mathrm{T}}\mathbf{y} \\ \nabla_{\mathbf{h}_c}\mathbf{h}_c^{\mathrm{T}}\Sigma_c^{-1}\mathbf{h}_c &= 2\Sigma_c^{-1}\mathbf{h}_c. \end{aligned}$$

For the gradient of the second term $\nabla_{\mathbf{h}_c}\mathbf{1}_N^{\mathrm{T}} b(f(\boldsymbol{\mu}))$, denoting $\boldsymbol{\eta} = f(\boldsymbol{\mu})$ for simplicity, we have the following

$$\begin{aligned} \nabla_{\mathbf{h}_c}\mathbf{1}_N^{\mathrm{T}} b(\boldsymbol{\eta}) &= \frac{\partial b(\boldsymbol{\eta})}{\partial \mathbf{h}_c}\mathbf{1}_N \\ &= \frac{\partial \boldsymbol{\eta}}{\partial \mathbf{h}_c}\frac{\partial b(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}}\mathbf{1}_N \\ &= \mathbf{X}_c^{\mathrm{T}}\operatorname{diag}(\boldsymbol{\mu})\mathbf{1}_N = \mathbf{X}_c^{\mathrm{T}}\boldsymbol{\mu}. \end{aligned}$$

We use the well-known relationship for the natural exponential family [16], which states that $db(\boldsymbol{\eta}_i)/d\boldsymbol{\eta}_i = \mathbb{E}[\mathbf{y}_i] = \boldsymbol{\mu}_i$ with the subscript i referring to ith element of the corresponding vector. We get $\nabla_{\mathbf{h}_c}\mathcal{L}(\mathbf{y})$ in Eq. (3) by collecting these terms. For the Hessian, we compute $\partial(\mathbf{X}_c^{\mathrm{T}}\boldsymbol{\mu})/\partial\mathbf{h}_c$ as follows

$$\begin{aligned} \frac{\partial \mathbf{X}_c^{\mathrm{T}}\boldsymbol{\mu}}{\partial \mathbf{h}_c} &= \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{h}_c}\mathbf{X}_c = \frac{\partial f^{-1}(\boldsymbol{\eta})}{\partial \mathbf{h}_c}\mathbf{X}_c \\ &= \frac{\partial \boldsymbol{\eta}}{\partial \mathbf{h}_c}\frac{\partial f^{-1}(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}}\mathbf{X}_c \\ &= \mathbf{X}_c^{\mathrm{T}}\operatorname{diag}((f'(\boldsymbol{\mu}_i))^{-1})\mathbf{X}_c. \end{aligned}$$

Letting $\mathbf{W}_c^{j,(t)} = \operatorname{diag}((f'(\boldsymbol{\mu}^{j,(t)}))^{-1})$, we have

$$\begin{aligned} \mathbf{h}_c^{(t+1)} &= \mathbf{h}_c^{(t)} - (\nabla^2_{\mathbf{h}_c}\mathcal{L}(\mathbf{y}))^{-1}\nabla_{\mathbf{h}_c}\mathcal{L}(\mathbf{y}) \\ &= (\phi^{-1}\sum_{j'}(\mathbf{X}_c^{j'})^{\mathrm{T}}\mathbf{W}_c^{j',(t)}\mathbf{X}_c^{j'} + \Sigma_c^{-1})^{-1}\sum_j(\mathbf{X}_c^j)^{\mathrm{T}}\mathbf{z}_c^{j,(t+1)}, \end{aligned} \tag{6.4}$$

where $\mathbf{z}_c^{j,(t+1)} = \phi^{-1}(\mathbf{W}_c^{j,(t)}\mathbf{X}_c^j\mathbf{h}_c^{(t)} + (\mathbf{y}^j - \boldsymbol{\mu}^{j,(t)})) \in \mathbb{R}^N$. After each update, we normalize

$\mathbf{h}_c^{(t+1)}$ to have unit norm. We update $\mathbf{h}_c$ in a cyclic manner and proceed to the next CSC step. The role of $(\mathbf{X}_c^j)^{\mathrm{T}}$ in $(\mathbf{X}_c^j)^{\mathrm{T}}\mathbf{z}_c^{j,(t+1)}$ is to extract the segments of $\mathbf{z}_c^{j,(t+1)}$ where $\mathbf{h}_c^{(t)}$ occurs, and take their weighted average [13]. Since $K \ll N$, the computational complexity of matrix inversion for Σ_c and $\nabla^2_{\mathbf{h}_c}\mathcal{L}(\mathbf{y})$ is negligible.

In summary, the CDU step seamlessly incorporates the GP constraint into the classical IRLS algorithm [163]. Since the optimization is not dependent on the form of Σ_c, we can choose different Σ_c to enforce different degrees of smoothness. This is simpler compared to approaches utilizing total-variation like penalties [159, 160], which require custom, dedicated primal-dual optimization methods for different penalties [169].

6.4 Analysis of converged dictionary

We now analyze how GPCDL promotes the smoothness of $\mathbf{h}_c$. We focus mainly on the case where the observations are Gaussian for intuition. We assume that the templates are non-overlapping, that is $(\mathbf{S}_{c,i}^j)^{\mathrm{T}}\mathbf{S}_{c,i'}^j = \mathbf{0}$ for $i \neq i'$.

Gaussian case IRLS converges in a single iteration (we omit the index t), with f as the identity and $\mathbf{W}_c^j = \mathbf{I}_{N\times N}$. This yields $\mathbf{z}_c^j = \phi^{-1}(\mathbf{y}^j - \sum_{c'\neq c}\mathbf{X}_{c'}^j\mathbf{h}_{c'} - \mathbf{a}^j)$. The dispersion is the observation noise variance, i.e., $\phi = \sigma_\varepsilon^2$.

$$\begin{aligned}\mathbf{h}_c &= \left(\sigma_\varepsilon^{-2}\sum_j(\mathbf{X}_c^j)^{\mathrm{T}}\mathbf{X}_c^j + \Sigma_c^{-1}\right)^{-1}\sum_j(\mathbf{X}_c^j)^{\mathrm{T}}\mathbf{z}_c^j \\ &= \left(\sigma_\varepsilon^{-2}\sum_{j,i}(x_{c,i}^j)^2\mathbf{I} + \Sigma_c^{-1}\right)^{-1}\sum_j(\mathbf{X}_c^j)^{\mathrm{T}}\mathbf{z}_c^j,\end{aligned} \tag{6.5}$$

where the second equality follows from $(\mathbf{S}_{c,i}^j)^{\mathrm{T}}\mathbf{S}_{c,i'}^j = \mathbf{0}$ for $i = i'$. The factor $\alpha^2 = \sigma_\varepsilon^{-2}\sum_{j,i}(x_{c,i}^j)^2$, which we term *code-SNR*, represents the SNR of the sparse codes, since $\sum_{j,i}(x_{c,i}^j)^2$ and σ_ε^2 are the energy of the codes and the noise, respectively.

Let us now examine $\mathbf{F}\mathbf{h}_c$, the spectra of $\mathbf{h}_c$, where $\mathbf{F} \in \mathbb{C}^{K\times K}$ is a discrete Fourier transform matrix, with $\mathbf{F}_{k,k'} = \exp(-2\pi i(k-1)(k'-1)/K)$, and $\omega_k = 2\pi k/K$. Using the eigen-decomposition for a stationary kernel [112], we get

$$\Sigma_c \simeq \mathbf{F}^{-1}\operatorname{diag}([\gamma_c(\omega_1),\ldots,\gamma_c(\omega_K)])\mathbf{F}. \tag{6.6}$$

Denoting $\mathbf{E}_c = \sum_j (\mathbf{X}_c^j)^{\mathrm{T}}\mathbf{z}_c^j$ for notational simplicity, and using $\mathbf{F}\mathbf{F}^{-1} = \mathbf{I}$, we have

$$\begin{aligned}\mathbf{F}\mathbf{h}_c &\simeq \mathbf{F}(\alpha^2\mathbf{I} + \mathbf{F}^{-1}\operatorname{diag}([\gamma_c^{-1}(\omega_1), \ldots, \gamma_c^{-1}(\omega_K)])\mathbf{F})^{-1}\mathbf{E}_c \\ &= \operatorname{diag}\Big(\mathbf{g}\Big)\mathbf{F}\widetilde{\mathbf{h}}_c, \end{aligned} \tag{6.7}$$

where $\mathbf{g}_k = \gamma_c(\omega_k)/(\gamma_c(\omega_k) + \alpha^{-2})$ and $\widetilde{\mathbf{h}}_c = \mathbf{E}_c/\alpha^2$. We can interpret Eq. (6.7) as *Wiener filter* [164] with gain $\mathbf{g}_k$ at $\omega = \omega_k$ on $\widetilde{\mathbf{h}}_c$, the learned template *without* the regularization.

The gain $\mathbf{g}_k$ depends on two factors: 1) the code-SNR α^2 and 2) the PSD of the GP prior $\gamma_c(\omega_k)$. For fixed α^2, the larger (and smaller) $\gamma_c(\omega_k)$, the closer $\mathbf{g}_k$ to 1 (and 0). Therefore, $\mathbf{g}_k$ acts as a lowpass filter and suppresses high-frequency content, allowing accurate learning of smooth $\mathbf{h}_c$. Fig. 6-1 demonstrates how different l_c lead to different gains $\mathbf{g}$. If α^2 is increased by collecting more data (increasing J), $\mathbf{g}_k$ increases across the entire ω axis and the filtering effect diminishes. This agrees with the Bayesian intuition that with more data, the likelihood dominates the prior. Note that with increasing J, $\widetilde{\mathbf{h}}_c$ itself becomes more accurate [170].

This suggests that GPCDL shares the same philosophy as [157, 13], since the learned dictionary is lowpass-filtered. However, the filters are designed differently. For GPCDL, the Wiener filter is *data-adaptive*, as the gain is determined *a posteriori* from the balance between the likelihood (data) and the prior. In contrast, the filter is designed *a priori* in [157, 13, 162], without reference to the data or optimization criteria.

We note that a similar form has been studied in the spectral filtering theory for Tikhonov regularization [171]. Tikhonov regularization can be recovered from Eq. (6.2) with $\Sigma_c = \sigma_c^2\mathbf{I}$. The diagonal covariance yields $\gamma_c(\omega_k) = \gamma_c$, $\forall k$, and consequently constant gain $\mathbf{g}_k = \mathbf{g}$, $\forall k$, resulting in $\widetilde{\mathbf{h}}$ with a smaller norm, shown in Fig. 6-1 (green). For GPCDL, however, Σ_c is symmetric and non-diagonal. This allows GPCDL to have frequency-dependent Wiener filter gain.

General case For non-Gaussian distributions, two factors complicate the interpretation: 1) IRLS requires multiple iterations to converge and 2) $\mathbf{W}_c^{j,(t)}$ is dependent on

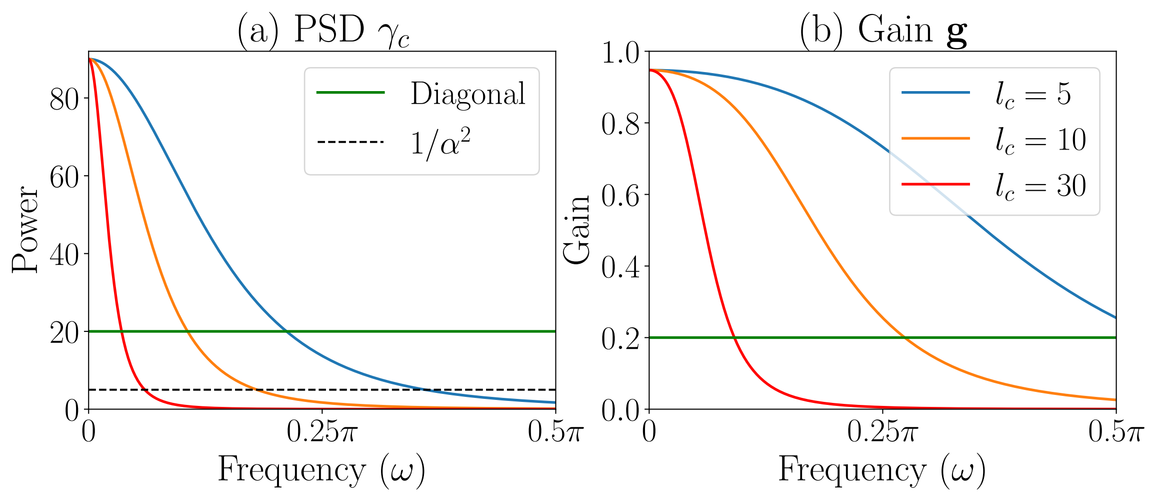

Figure 6-1: (a) PSD $\gamma_c(\omega)$ for Matern kernel with $\nu = 1.5$ and varying l_c for fixed α. The green line corresponds to Tikhonov regularization with diagonal σ_c^2. (b) The filter gain $\mathbf{g}$.

ω_k and iteration t. However, we conjecture that smoothing still takes place. Specifically, $\mathbf{R}^{(t)} = (\mathbf{X}_c^j)^{\mathrm{T}}\mathbf{W}_c^{j,(t)}\mathbf{X}_c^j$ is still a diagonal matrix with $\mathbf{R}_{(k,k)}^{(t)} = \phi^{-1}\sum_{j,i}(x_{c,i}^j)^2 \cdot (f'(\boldsymbol{\mu}_{n_{c,i}^j+k-1}^{j,(t)}))^{-1}$. This consequently yields $\mathbf{g}_k = \gamma_c(\omega_k)/(\gamma_c(\omega_k) + (\mathbf{R}_{(k,k)}^{(t)})^{-1})$, computed using $\mathbf{R}^{(t)} = \mathbf{F}^{-1}\mathbf{R}^{(t)}\mathbf{F}$. Therefore, the relation between $\gamma_c(\omega_k)$ and $(\mathbf{R}_{(k,k)}^{(t)})^{-1}$ holds as in the Gaussian case. Consequently, $\mathbf{g}_k$ filters the spectra of weighted-averaged segments from $\mathbf{z}_c^{j,(t)}$, extracted by the operator $(\mathbf{X}_c^j)^{\mathrm{T}}$. Empirically, we observe that low-pass filtering still occurs.

6.5 Experiments

We apply our framework to two datasets: 1) simulated data (Gaussian) and 2) neural spiking data from rats (Bernoulli). We use the Matern kernel with $\nu = 1.5$, fix $\sigma_c^2 = 1$, and vary l_c to control the regularization. We use the mixture of Gaussians (MOG) model $\mathbf{h}_c^{\mathrm{MOG}}[k] = \sum_{d=1}^{D} a_{c,d}\exp(-(k-\mu_{c,d})^2/\sigma_{c,d}^2)$ as baseline, with parameters $\{a_d, \mu_d, \sigma_d^2\}_{d=1}^{D}$ determined by maximum-likelihood estimation. MOG represents a smooth parametric approach. We run 15 iterations of our algorithm, with $\widehat{\mathbf{h}}_c$ and $\widehat{\mathbf{x}}_c$ denoting the solutions at convergence.

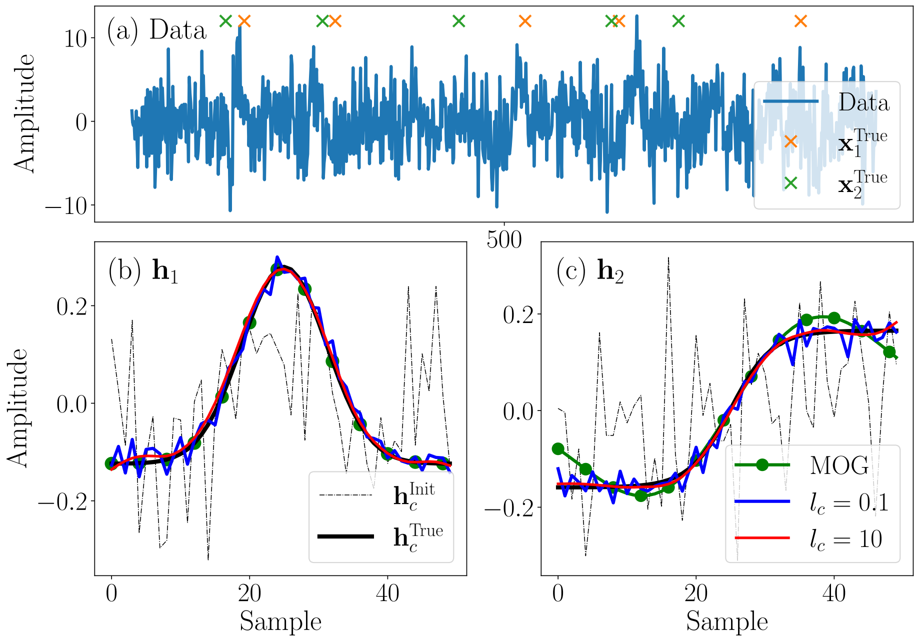

Figure 6-2: Simulation results for $J = 100$ and $\sigma_\varepsilon^2 = 10$. (a) An example data trace and true codes. (b-c) Dictionary elements.

6.5.1 Simulated data

Dataset We simulated Gaussian data with $\{\mathbf{h}_c^{\text{True}}\}_{c=1}^2 \in \mathbb{R}^{50}$ (Fig. 6-2 (black) Gaussian and sigmoid), each appearing 4 times with magnitude uniformly sampled from $[10, 20]$, throughout the length $N = 1{,}000$ signal. The signal is perturbed with Gaussian noise with variance $\sigma_\varepsilon^2 = 5$. For evaluation, we use the dictionary error, $\text{err}(\widehat{\mathbf{h}}_c) = (1 - \langle \widehat{\mathbf{h}}_c, \mathbf{h}_c^{\text{True}} \rangle^2)^{\frac{1}{2}}$ [170]. We perturbed $\mathbf{h}_c^{\text{True}}$ with Gaussian noise and obtain $\mathbf{h}_c^{\text{Init}}$ (dotted black) with $\text{err}(\mathbf{h}_c^{\text{Init}}) > 0.7$. We averaged the power $\omega \in [0.5\pi, \pi]$ to obtain the dispersion $\widehat{\phi} = \widehat{\sigma}_\varepsilon^2$.

Results Table 6.1 shows the error, averaged over 10 independent runs, for varying SNR and lengthscale l_c. The larger the l_c, the stronger the GP regularization, resulting in considerably lower errors, as visually supported in Fig. 6-2. The learned $\widehat{\mathbf{h}}_c$ for $l_c = 0.1$ (blue) corresponding to minimal regularization, contains high-frequency noise. With GP regularization ($l_c = 10$, red), the noise is filtered out, and thus $\widehat{\mathbf{h}}_c$ is more accurate with the same code-SNR. As expected, the overall errors are lower with higher code-SNR, where $(J, \sigma_\varepsilon^2) = (10, 10)$ and $(100, 5)$ correspond to the lowest and the highest code-SNR. Even with high code-SNR, we observe the benefits of GP

Table 6.1: Dictionary error $\text{err}(\widehat{\mathbf{h}}_c)$ for simulated data with $J = \{10, 100\}$ and $\sigma_\varepsilon^2 = \{5, 10\}$.

	l_c	0.1		25		100	
Error	σ_ε^2 \ J	10	100	10	100	10	100
$\text{err}(\widehat{\mathbf{h}}_1)$	5	0.29	0.18	0.18	0.12	**0.13**	**0.06**
$\text{err}(\widehat{\mathbf{h}}_1)$	10	0.45	0.30	0.36	0.23	**0.20**	**0.11**
$\text{err}(\widehat{\mathbf{h}}_2)$	5	0.32	0.18	0.21	0.11	**0.10**	**0.06**
$\text{err}(\widehat{\mathbf{h}}_2)$	10	0.46	0.31	0.28	0.24	**0.17**	**0.14**

regularization.

Figs. 6-2 (b-c) also depict $\widehat{\mathbf{h}}_1^{\text{MOG}}$ and $\widehat{\mathbf{h}}_2^{\text{MOG}}$, optimized with $D = 1$ and 2, respectively. This shows potential issues of model misspecification in the parametric approach, as observed in Fig. 6-2 (c), where $\widehat{\mathbf{h}}_2^{\text{MOG}}$ cannot adequately model the sigmoid. On the other hand, the nonparametric GPCDL does not face this issue.

6.5.2 Neural activity data from barrel cortex

Dataset We used neural spiking data collected from the barrel cortex of mice [172]. The experiments consist of multiple trials, with each trial $N = 3{,}000$ ms and $\mathbf{y}^j \in \{0, 1\}^N$. During each trial, a stimulus (Fig. 6-3 (b)) is used to deflect the whisker of a mouse every 125 ms. We set $K = 125$ accordingly. Because of the presence of a single stimulus, we assumed $C = 1$ as in [59]. For $\mathbf{h}_1^{\text{Init}}$, we used the first-order difference of the stimulus (dotted black). We used the logit function as the canonical link and set $\phi = 1$. We assumed a constant baseline $\mathbf{a}^j = \mathbf{a}, \forall j$ and estimate it from all J segments. We also assumed $\mathbf{x}_1^j = \mathbf{x}_1, \forall j$.

We used $J = 30$ trials for training and $J_{\text{test}} = 10$ trials for testing for each neuron. We performed 3-fold cross-validation on the training data to find l_1^{CV} that yields the highest predictive log-likelihood (pll). We used the entire training data to estimate $\widehat{\mathbf{h}}_1$ and $\widehat{\mathbf{x}}_1$. We used pll and R^2 [50] as performance metrics.

Results Table 6.2 shows the metrics for two neurons. Figs. 6-3 (c-d) shows $\widehat{\mathbf{h}}_1$ corresponding to varying l_1 for Neuron 1 with $J = 30$ (red). Both the highest pll and R^2 for the cross-validation is achieved for $l_1^{\text{CV}} = 25$. For the test data, l_1^{CV} also

Table 6.2: Metrics (the higher the better) for two neurons with $J = 30$. MOG represents the mixture of Gaussians.

		Train			Test			
ID	l_c	0.01	25	200	0.01	25	200	MOG
1	pll	0.57	**0.60**	0.57	0.61	**0.65**	0.5	0.62
1	R^2	0.28	**0.30**	0.25	0.27	**0.30**	0.25	0.29
2	pll	0.59	**0.63**	0.62	0.64	**0.70**	0.67	0.69
2	R^2	0.22	**0.24**	0.23	0.18	**0.23**	0.21	0.23

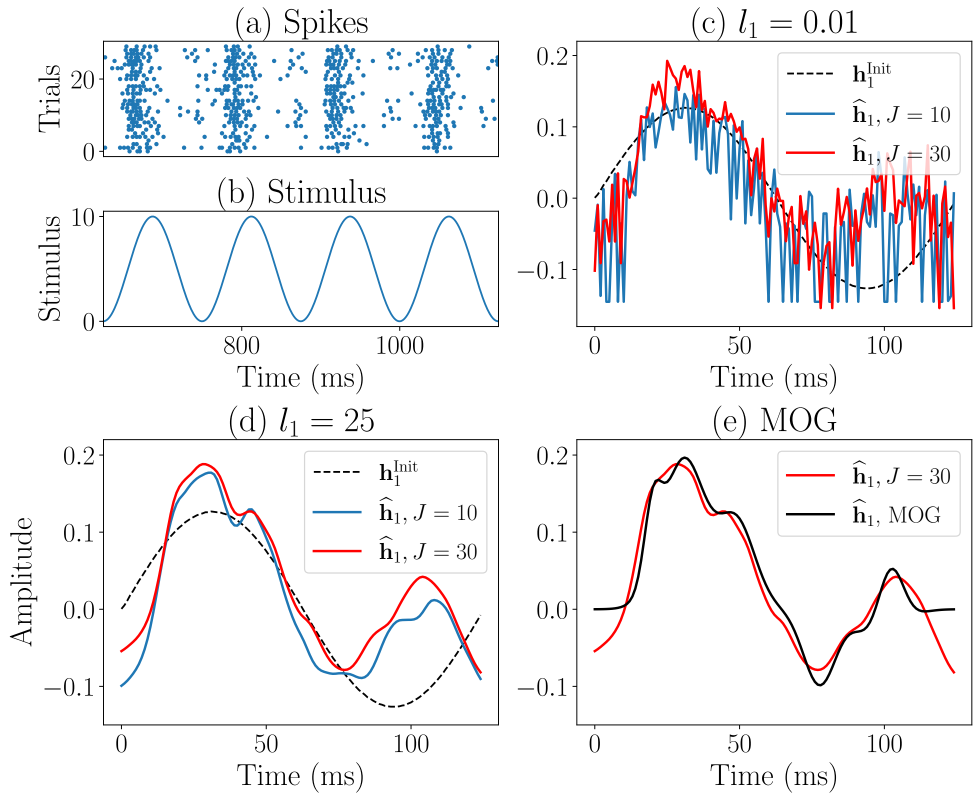

Figure 6-3: Real data for Neuron 1. (a-b) Raster plot of the spikes and periodic stimulus. (c-d) $\widehat{\mathbf{h}}_1$ with $J = 30$ (red) and $J = 10$ (blue) for various l_1. (e) The parametric baseline, $\widehat{\mathbf{h}}_1^{\text{MOG}}$.

performs the best. We observe the two peaks in $\widehat{\mathbf{h}}_1$ (red), around 30 and 100 ms, validated by the repeated pattern of the strong bursts of spikes followed by the weak burst. For $l_1 = 0.01$, although the two peaks can be identified, $\widehat{\mathbf{h}}_1$ lacks smoothness, as a result of overfitting to the *integer-valued* observations without the smoothness constraint. For $l_1 = 200$ with strong regularization, $\widehat{\mathbf{h}}_1$ is overly smoothed and produces lower metrics.

Comparison between $J=10$ (blue) and $J=30$ (red) shows the benefits of the regularization for limited data. Without regularization (Fig. 6-3 (c)), $\widehat{\mathbf{h}}_1$ for $J=10$ is noisier than that for $J=30$ due to the scarcity of data, in addition to the nonlinear link. For l_c^{CV}, $\widehat{\mathbf{h}}_1$ for both cases are similar, showing that the regularized dictionary is robust for limited data.

Finally, we compared $\widehat{\mathbf{h}}_1$ with $\widehat{\mathbf{h}}_1^{\text{MOG}}$. We chose $D=6$ that minimizes the Akaike Information Criterion [116]. Fig. 6-3 (e) shows that $\widehat{\mathbf{h}}_1^{\text{MOG}}$ is indeed very similar to $\widehat{\mathbf{h}}_1$ with l_1^{CV}. However, Table 6.2 shows that the nonparametric and regularized approaches outperform the parametric alternative, indicating the flexibility of the nonparametric approach.

6.6 Conclusion

We proposed a framework for learning convolutional dictionaries using data from the natural exponential family by regularizing the classical objective with a Gaussian process prior. We show that the smoothness constraint leads to a dictionary with better performance. GPCDL is a powerful framework that combines 1) the smoothness previously achieved by parametric functions, which is vulnerable to model misspecification issues, or penalty functions, which are nontrivial to optimize, and 2) the flexibility of the nonparametric dictionary.

Chapter 7

Model-based deep learning

Primer This chapter is adapted from the following work (* Equal contribution)

- Bahareh Tolooshams*, Andrew H. Song*, Simona Temereanca, and Demba Ba, *Convo- lutional dictionary learning based auto-encoders for natural exponential-family distributions*, International Conference on Machine Learning (ICML), 2020
- Alexander Lin, **Andrew H. Song**, and Demba Ba, *Mixture Model Auto-Encoders: Deep Clustering through Dictionary Learning*, Submitted to IEEE ICASSP, 2021

7.1 Introduction

Learning shift-invariant patterns from a dataset has given rise to work in different communities, most notably in signal processing (SP) and deep learning. In the former, this problem is referred to as convolutional dictionary learning (CDL) [173]. CDL imposes a linear *generative model* where the data are generated by a sparse linear combination of shifts of localized patterns. In the latter, convolutional neural networks (NNs) [174] have excelled in identifying shift-invariant patterns.

Recently, the iterative nature of the optimization algorithms for performing CDL has inspired the utilization of NNs as an efficient and scalable alternative, starting with the seminal work of [53], and followed by [175, 176, 177]. Specifically, the iterative steps are expressed as a recurrent NN, and thus solving the optimization simply becomes passing the input through an unrolled NN [178, 179]. At one end of the spectrum, this perspective, through weight-tying, leads to architectures with significantly fewer parameters than a generic NN. At the other end, by untying the weights, it motivates new architectures that depart, and could not be arrived at, from the generative perspective [53, 176].

The majority of the literature at the intersection of generative models and NNs assumes that the data are real-valued and therefore are not appropriate for binary or count-valued data, such as neural spiking data and photon-based images [180]. Nevertheless, several works on Poisson image denoising, arguably the most popular application involving non real-valued data, can be found separately in both communities. In the SP community, the negative Poisson data likelihood is either explicitly minimized [181, 156] or used as a penalty term added to the objective of an image denoising problem with Gaussian noise [182]. Being rooted in the dictionary learning formalism, these methods operate in an *unsupervised* manner. Although they yield good denoising performance, their main drawbacks are scalability and computational efficiency.

In the deep learning community, NNs tailored to image denoising [183, 184, 185], which are reminiscent of residual learning, have shown great performance on Poisson image denoising. However, since these 1) are not designed from the generative model perspective and/or 2) are *supervised* learning frameworks, it is unclear how they can be adapted to the classical CDL, where the task is *unsupervised* and the interpretability of the parameters is important. NNs with a generative flavor, namely variational auto-encoders (VAEs), have been extended to utilize non real-valued data [186, 187]. However, these architectures cannot be adapted to solve the CDL task.

To address this gap, we make the following contributions:

Auto-encoder inspired by CDL for non real-valued data We introduce a flexible class of auto-encoder (AE) architectures for data from the natural exponential-family that combines the perspectives of generative models and NNs. We term this framework, depicted in Fig. 7-1, the deep convolutional exponential-family auto-encoder (DCEA).

Unsupervised learning of convolutional patterns We show through simulation that DCEA performs CDL and learns convolutional patterns from binomial observations. We also apply DCEA to real neural spiking data and show that it fits the data better than baselines.

Supervised learning framework DCEA, when trained in a supervised manner, achieves similar performance to state-of-the-art algorithms for Poisson image denoising with orders of magnitude fewer parameters compared to other baselines, owing to its design based on a generative model.

Gradient dynamics of shallow exponential auto-encoder Given some assumptions on the binomial generative model with dense dictionary and "good" initializations, we prove in Theorem **??** that shallow exponential auto-encoder (SEA), when trained by gradient descent, recovers the dictionary.

7.2 Problem Formulation

Natural exponential-family distribution For a given observation vector $\mathbf{y} \in \mathbb{R}^N$, with mean $\boldsymbol{\mu} \in \mathbb{R}^N$, we define the log-likelihood of the *natural exponential family* [163] as

$$\log p(\mathbf{y}|\boldsymbol{\mu}) = f\Big(\boldsymbol{\mu}\Big)^{\mathrm{T}}\mathbf{y} + g(\mathbf{y}) - B\Big(\boldsymbol{\mu}\Big), \tag{7.1}$$

where we have assumed that, conditioned on $\boldsymbol{\mu}$, the elements of $\mathbf{y}$ are independent. The natural exponential family includes a broad family of probability distributions such as the Gaussian, binomial, and Poisson. The functions $g(\cdot)$, $B(\cdot)$, as well as the invertible *link function* $f(\cdot)$, all depend on the choice of distribution.

Convolutional generative model We assume that $f(\boldsymbol{\mu})$ is the sum of scaled and time-shifted copies of C finite-length filters (dictionary) $\{\mathbf{h}_c\}_{c=1}^{C} \in \mathbb{R}^K$, each localized, i.e., $K \ll N$. We can express $f(\boldsymbol{\mu})$ in a convolutional form: $f(\boldsymbol{\mu}) = \sum_{c=1}^{C} \mathbf{h}_c * \mathbf{x}^c$, where $*$ is the convolution operation, and $\mathbf{x}^c \in \mathbb{R}^{N-K+1}$ is a train of scaled impulses which we refer to as *code vector*. Using linear-algebraic notation, $f(\boldsymbol{\mu}) = \sum_{c=1}^{C} \mathbf{h}_c * \mathbf{x}^c = \mathbf{H}\mathbf{x}$, where $\mathbf{H} \in \mathbb{R}^{N \times C(N-K+1)}$ is a matrix that is the concatenation of C

Toeplitz (i.e., banded circulant) matrices $\mathbf{H}^c \in \mathbb{R}^{N\times(N-K+1)}$, $c = 1,\ldots,C$, and $\mathbf{x} = [(\mathbf{x}^1)^\mathrm{T},\ldots,(\mathbf{x}^C)^\mathrm{T}]^\mathrm{T} \in \mathbb{R}^{C(N-K+1)}$.

We refer to the input/output domain of $f(\cdot)$ as the data and dictionary domains, respectively. We interpret $\mathbf{y}$ as a time-series and the non-zero elements of $\mathbf{x}$ as the times when each of the C filters are active. When $\mathbf{y}$ is two-dimensional (2D), i.e., an image, $\mathbf{x}$ encodes the spatial locations where the filters contribute to its mean $\boldsymbol{\mu}$.

Exponential convolutional dictionary learning (ECDL) Given J observations $\{\mathbf{y}^j\}_{j=1}^J$, we estimate $\{\mathbf{h}_c\}_{c=1}^C$ and $\{\mathbf{x}^j\}_{j=1}^J$ that minimize the negative log-likelihood $\sum_{j=1}^J l(\mathbf{x}^j) = -\sum_{j=1}^J \log p(\mathbf{y}^j|\{\mathbf{h}_c\}_{c=1}^C, \mathbf{x}^j)$ under the convolutional generative model, subject to sparsity constraints on $\{\mathbf{x}^j\}_{j=1}^J$. We enforce sparsity using the ℓ_1 norm, which leads to the non-convex optimization problem

$$\min_{\substack{\{\mathbf{h}_c\}_{c=1}^C\\ \{\mathbf{x}^j\}_{j=1}^J}} \sum_{j=1}^J \overbrace{-(\mathbf{H}\mathbf{x}^j)^\mathrm{T}\mathbf{y}^j + B\Big(f^{-1}\Big(\mathbf{H}\mathbf{x}^j\Big)\Big)}^{l(\mathbf{x}^j)} + \lambda\|\mathbf{x}^j\|_1, \tag{7.2}$$

where the regularizer λ controls the degree of sparsity. A popular approach to deal with the non-convexity is to minimize the objective over one set of variables, while the others are fixed, in an alternating manner, until convergence [170]. When $\{\mathbf{x}^j\}_{j=1}^J$ is being optimized with fixed $\mathbf{H}$, we refer to the problem as convolutional sparse coding (CSC). When $\mathbf{H}$ is being optimized with $\{\mathbf{x}^j\}_{j=1}^J$ fixed, we refer to the problem as convolutional dictionary update (CDU).

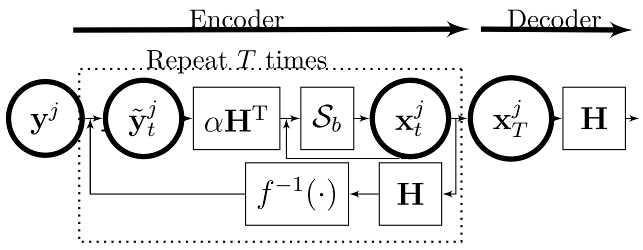

Figure 7-1: DCEA architecture for ECDL. The encoder/decoder structure mimics the CSC/CDU sequence in CDL. The encoder performs T iterations of CSC. Each iteration uses *working observations* (residuals) $\tilde{\mathbf{y}}_t^j$ obtained by iteratively modifying $\mathbf{y}^j$ using the filters $\mathbf{H}$ and a nonlinearity $f^{-1}(\cdot)$ that depends on the distribution of $\mathbf{y}^j$. The dictionary $\mathbf{H}$ is updated through the backward pass.

7.3 Deep convolutional exponential auto-encoder

We propose a class of auto-encoder architectures to solve the ECDL problem, which we term deep convolutional exponential-family auto-encoder (DCEA). Specifically, we make a one-to-one connection between the CSC/CDU steps and the encoder/decoder of DCEA depicted in Fig. 7-1. We focus only on CSC for a single $\mathbf{y}^j$, as the CSC step can be parallelized across examples.

7.3.1 The architecture

Encoder The forward pass of the encoder maps the input $\mathbf{y}^j$ into the sparse code $\mathbf{x}^j$. Given the filters $\{\mathbf{h}_c\}_{c=1}^C$, the encoder solves the ℓ_1-regularized optimization problem from Eq. (7.2)

$$\min_{\mathbf{x}^j} l(\mathbf{x}^j) + \lambda \|\mathbf{x}^j\|_1 \tag{7.3}$$

in an iterative manner by unfolding T iterations of the proximal gradient algorithm [188]. For Gaussian observations, Eq. (7.3) becomes an ℓ_1-regularized least squares problem, for which several works have unfolded the proximal iteration into a recurrent network [53, 189, 176, 190].

We use $\mathcal{S}_b \in \{\text{ReLU}_b, \text{Shrinkage}_b\}$ to denote a proximal operator with bias $b \geq 0$. We consider three operators,

$$\begin{aligned} \text{ReLU}_b(\mathbf{z}) &= (\mathbf{z} - b) \cdot \mathbb{1}_{\{\mathbf{z} \geq b\}} \\ \text{Shrinkage}_b(\mathbf{z}) &= \text{ReLU}_b(\mathbf{z}) - \text{ReLU}_b(-\mathbf{z}), \end{aligned} \tag{7.4}$$

where $\mathbb{1}$ is an indicator function. If we constrain the entries of $\mathbf{x}^j$ to be non-negative, we use $\mathcal{S}_b = \text{ReLU}_b$. Otherwise, we use $\mathcal{S}_b = \text{Shrinkage}_b$. A single iteration of the proximal gradient step is given by

$$\begin{aligned} \mathbf{x}_t^j &= \mathcal{S}_b\left(\mathbf{x}_{t-1}^j - \alpha \nabla_{\mathbf{x}_{t-1}^j} \log p\Big(\mathbf{y}^j | \{\mathbf{h}_c\}_{c=1}^C, \mathbf{x}_{t-1}^j\Big)\right) \\ &= \mathcal{S}_b\Big(\mathbf{x}_{t-1}^j + \alpha \mathbf{H}^{\mathrm{T}} (\underbrace{\mathbf{y} - f^{-1}\Big(\mathbf{H}\mathbf{x}_{t-1}^j\Big)}_{\widetilde{\mathbf{y}}_t^j})\Big), \end{aligned} \tag{7.5}$$

where $\mathbf{x}_t^j$ denotes the sparse code after t iterations of unfolding, and α is the step size of the gradient update. The term $\widetilde{\mathbf{y}}_t^j$ is referred to as *working observation*. The choice of α, which we explore next, depends on the generative distribution. We also note that there is a one-to-one mapping between the regularization parameter λ and the bias b of $\mathcal{S}_b$. We treat λ, and therefore b, as hyperparameters that we tune to the desired sparsity level. The matrix $\mathbf{H}^{\mathrm{T}}$ effectively computes the correlation between $\widetilde{\mathbf{y}}_t^j$ and $\{\mathbf{h}_c\}_{c=1}^C$. Assuming that we unfold T times, the output of the encoder is $\mathbf{x}_T^j$.

The architecture consists of two nonlinear activation functions: $\mathcal{S}_b(\cdot)$ to enforce sparsity, and $f^{-1}(\cdot)$, the inverse of the link function. For Gaussian observations $f^{-1}(\cdot)$ is linear with slope 1. For other distributions in the natural exponential family, the encoder uses $f^{-1}(\cdot)$, a mapping from the dictionary domain to the data domain, to transform the input $\mathbf{y}^j$ at each iteration into a working observation $\widetilde{\mathbf{y}}_t^j$.

Decoder & training We apply the decoder $\mathbf{H}$ to $\mathbf{x}_T^j$ to obtain the linear predictor $\mathbf{H}\mathbf{x}_T^j$. This decoder completes the forward pass of DCEA. We use the negative log-likelihood $\mathcal{L}_{\mathbf{H}}^{\text{unsup.}} = \sum_{j=1}^J l(\mathbf{x}^j)$ as the loss function applied to the decoder output for updating the dictionary. We train the weights of DCEA, fully specified by the filters $\{\mathbf{h}_c\}_{c=1}^C$, by gradient descent through backpropagation. Note that the ℓ_1 penalty is not a function of $\mathbf{H}$ and is not in the loss function.

Table 7.1: Generative models for DCEA.

	Gaussian	Binomial	Poisson
$\mathbf{y}^j$	$\mathbb{R}$	$[0..M_j]$	$[0..\infty)$
$f^{-1}(\cdot)$	$I(\cdot)$	$\text{sigmoid}(\cdot)$	$\exp(\cdot)$
$B(\mathbf{z})$	$\mathbf{z}^{\mathrm{T}}\mathbf{z}$	$-\mathbf{1}^{\mathrm{T}}\log(\mathbf{1}-\mathbf{z})$	$\mathbf{1}^{\mathrm{T}}\mathbf{z}$
$\widetilde{\mathbf{y}}_t^j$	$\mathbf{y}^j - \mathbf{H}\mathbf{x}_{t-1}^j$	$\mathbf{y}^j - M_j \cdot \text{sigmoid}(\mathbf{H}\mathbf{x}_{t-1}^j)$	$\mathbf{y}^j - \exp(\mathbf{H}\mathbf{x}_{t-1}^j)$
$\mathbf{x}_t^j$	$\mathcal{S}_b\left(\mathbf{x}_{t-1}^j + \alpha\mathbf{H}^{\mathrm{T}}\widetilde{\mathbf{y}}_t^j\right)$	$\mathcal{S}_b\left(\mathbf{x}_{t-1}^j + \alpha\mathbf{H}^{\mathrm{T}}(\frac{1}{M_j}\widetilde{\mathbf{y}}_t^j)\right)$	$\mathcal{S}_b\left(\mathbf{x}_{t-1}^j + \alpha\mathbf{H}^{\mathrm{T}}\left(\text{Elu}(\widetilde{\mathbf{y}}_t^j)\right)\right)$

7.3.2 Binomial and Poisson generative models

We focus on two representative distributions for the natural exponential family: binomial and Poisson. For the binomial distribution, $\mathbf{y}^j$ assumes integer values from

0 to M_j. For the Poisson distribution, $\mathbf{y}^j$ can, in principle, be any non-negative integer values, although this is rare due to the exponential decay of the likelihood for higher-valued observations. Table 7.1 summarizes the relevant parameters for these distributions.

The fact that binomial and Poisson observations are integer-valued and have limited range, whereas the underlying $\boldsymbol{\mu}_j = f^{-1}(\mathbf{H}\mathbf{x}^j)$ is real-valued, makes the ECDL challenging. This is compounded by the nonlinearity of $f^{-1}(\cdot)$, which distorts the error in the *data* domain, when mapped to the *dictionary* domain. In comparison, in Gaussian CDL 1) the observations are real-valued and 2) $f^{-1}(\cdot)$ is linear.

This implies that, for successful ECDL, $\mathbf{y}^j$ needs to assume a diverse set of integer values. For the binomial distribution, this suggests that M_j should be large. For Poisson, as well as binomial, the maximum of $\boldsymbol{\mu}_j$ should also be large. This explains why the performance is generally lower in Poisson image denoising for a lower peak, where the peak is defined as the maximum value of $\boldsymbol{\mu}_j$ [156].

Practical design considerations for architecture As the encoder of DCEA performs iterative proximal gradient steps, we need to ensure that $\mathbf{x}_T^j$ converges. Convergence analysis of ISTA [140] shows that if $l(\mathbf{x}^j)$ is convex and has L–Lipschitz continuous gradient, which loosely means the Hessian is upper-bounded everywhere by $L > 0$, choosing $\alpha \in (0, 1/L]$ guarantees convergence. For the Gaussian distribution, L is the square of the largest singular value of $\mathbf{H}$ [52], denoted $\sigma^2_{\max}(\mathbf{H})$, and therefore $\alpha \in (0, 1/\sigma^2_{\max}(\mathbf{H})]$. For the Binomial distribution, $L = \frac{1}{4}\sigma^2_{\max}(\mathbf{H})$, and therefore $\alpha \in (0, 4/\sigma^2_{\max}(\mathbf{H})]$.

The gradient of the Poisson likelihood is not Lipschitz continuous. Therefore, we cannot set α a priori. In practice, the step size at every iteration is determined through a back-tracking line search [191], a process that is not trivial to replicate in the forward pass of a NN. We observed that the lack of a principled approach for picking α results, at times, in the residual assuming large negative values, leading to instabilities. Therefore, we imposed a finite lower-bound on the residual through an exponential linear unit (Elu) [192] which, empirically, we found to work the best compared to other nonlinear activation units. The convergence properties of this approach require further theoretical analysis that is outside of the scope of this paper.

7.4 Connection to unsupervised/supervised paradigm

We now analyze the connection between DCEA and ECDL. We first examine how the convolutional generative model places constraints on DCEA. Finally, we explain how DCEA can be modified for a supervised task.

7.4.1 Constrained structure of DCEA

We discuss key points that allow DCEA to perform ECDL.

- **Linear 1-layer decoder** In ECDL, the only sensible decoder is a one layer decoder comprising $\mathbf{H}$ and a linear activation. In contrast, the decoder of a typical AE consists of multiple layers along with nonlinear activations.
- **Tied weights across encoder and decoder** Although our encoder is *deep*, the same weights ($\mathbf{H}$ and $\mathbf{H}^{\mathrm{T}}$) are repeated across the layers.
- **Alternating minimization** The forward pass through the encoder performs the CSC step, and the backward pass, via backpropagation, performs the CDU step.

7.4.2 DCEA as a supervised framework

For the *supervised* paradigm, given the desired output (i.e., clean image, $\mathbf{y}^j_{\text{clean}}$, in the case of image denoising), we relax the DCEA architecture and untie the weights [176, 193, 190] as follows

$$\mathbf{x}^j_t = \mathcal{S}_{\mathbf{b}}\Big(\mathbf{x}^j_{t-1} + \alpha(\mathbf{W}^e)^{\mathrm{T}}(\mathbf{y} - f^{-1}\Big(\mathbf{W}^d\mathbf{x}^j_{t-1}\Big))\Big), \tag{7.6}$$

where we still use $\mathbf{H}$ as the decoder. We use $\{\mathbf{w}^e_c\}^C_{c=1}$ and $\{\mathbf{w}^d_c\}^C_{c=1}$ to denote the filters associated with $\mathbf{W}^e$ and $\mathbf{W}^d$, respectively. We train the bias vector $\mathbf{b}$, unlike in the unsupervised setting where we tune it by grid search [190, 194]. Compared to DCEA for ECDL, the number of parameters to learn has increased three-fold.

Although the introduction of additional parameters implies the framework is no longer exactly optimizing the parameters of the convolutional generative model, DCEA still maintains the core principles of the convolutional generative model. First, DCEA performs CSC, as $\mathbf{W}^e, \mathbf{W}^d,$ and $\mathbf{H}$ are convolutional matrices and $\mathcal{S}_\mathbf{b}$ ensures sparsity of $\mathbf{x}_T^j$. Second, the encoder uses $f^{-1}(\cdot)$, as specified by natural exponential family distributions. Therefore, we allow only a moderate departure from the generative model to balance the problem formulation and the problem-solving mechanism. Indeed, as we show in the Poisson image denoising of Section 7.5, the denoising performance for DCEA with untied weights is superior to that of DCEA with tied weights.

Indeed, the constraints can be relaxed further. For instance, 1) the proximal operator $\mathcal{S}_\mathbf{b}$ can be replaced by a deep NN [195], 2) the inverse link function $f^{-1}(\cdot)$ can be replaced by a NN [196], 3) $\mathbf{W}^d$, $\mathbf{W}^e$, and $\mathbf{H}$ can be untied across different iterations [178], and 4) the linear 1-layer decoder can be replaced with a deep nonlinear decoder. These would increase the number of trainable parameters, allowing for more expressivity and improved performance. Nevertheless, as our goal is to maintain the connection to *sparsity* and the *natural exponential family*, while keeping the number of parameters small, we do not explore these possibilities in this work.

7.5 Experiments

We apply our framework in three different settings.

- **Poisson image denoising** (*supervised*) We evaluate the performance of supervised DCEA in Poisson image denoising and compare it to state-of-the-art algorithms.
- **ECDL for simulation** (*unsupervised*) We use simulations to examine how the unsupervised DCEA performs ECDL for *binomial* data. With access to ground-truth data, we evaluate the accuracy of the learned dictionary. Additionally, we conduct ablation studies in which we relax the constraints on the DCEA architecture and

assess how accuracy changes.

- **ECDL for neural spiking data** (*unsupervised*) Using neural spiking data collected from mice [172], we perform unsupervised ECDL using DCEA. As is common in the analysis of neural data [17], we assume a *binomial* generative model.

Table 7.2: PSNR performance (in dB) of Poisson image denoising for five different models on test images for peak 1, 2, and 4: 1) SPDA, 2) BM3D+VST, 3) Class-agnostic, 4) DCEA constrained (DCEA-C), and 5) DCEA unconstrained (DCEA-UC).

		Camera	House	Peppers	Set12	BSD68	# of Params
Peak 1	SPDA	20.23	22.73	19.99	20.39	·	160,000
	BM3D+VST	20.37	22.35	19.89	·	21.01	N/A
	Class-agnostic	**21.59**	22.87	**21.43**	**21.51**	21.78	655,544
	DCEA-C (ours)	20.68	21.70	20.22	20.72	21.27	20,618
	DCEA-UC (ours)	21.47	**23.00**	20.91	21.37	**21.84**	61,516
Peak 2	SPDA	21.54	**25.09**	21.23	21.70	·	160,000
	BM3D+VST	22.13	24.18	21.97	·	22.21	N/A
	Class-agnostic	**23.25**	24.77	**23.19**	**22.97**	22.90	655,544
	DCEA-C (ours)	22.01	23.22	21.70	22.02	22.31	20,618
	DCEA-UC (ours)	22.94	24.52	22.94	22.79	**22.92**	61,516
Peak 4	SPDA	21.90	26.09	22.09	22.56	·	160,000
	BM3D+VST	23.94	26.04	24.07	·	23.54	N/A
	Class-agnostic	**24.87**	**26.59**	**24.83**	**24.40**	23.98	655,544
	DCEA-C (ours)	23.60	25.11	23.68	23.51	23.54	20,618
	DCEA-UC (ours)	24.66	26.47	24.71	24.37	**24.10**	61,516

7.5.1 Denoising Poisson images

We evaluated the performance of DCEA on Poisson image denoising for various peaks. We used the peak signal-to-noise-ratio (PSNR) as a metric. DCEA is trained in a *supervised* manner on the PASCAL VOC image set [197] containing $J = 5{,}700$ training images. $\mathcal{S}_{\mathbf{b}}$ is set to $\text{ReLU}_{\mathbf{b}}$. We used two test datasets: 1) Set12 (12 images) and 2) BSD68 (68 images) [198].

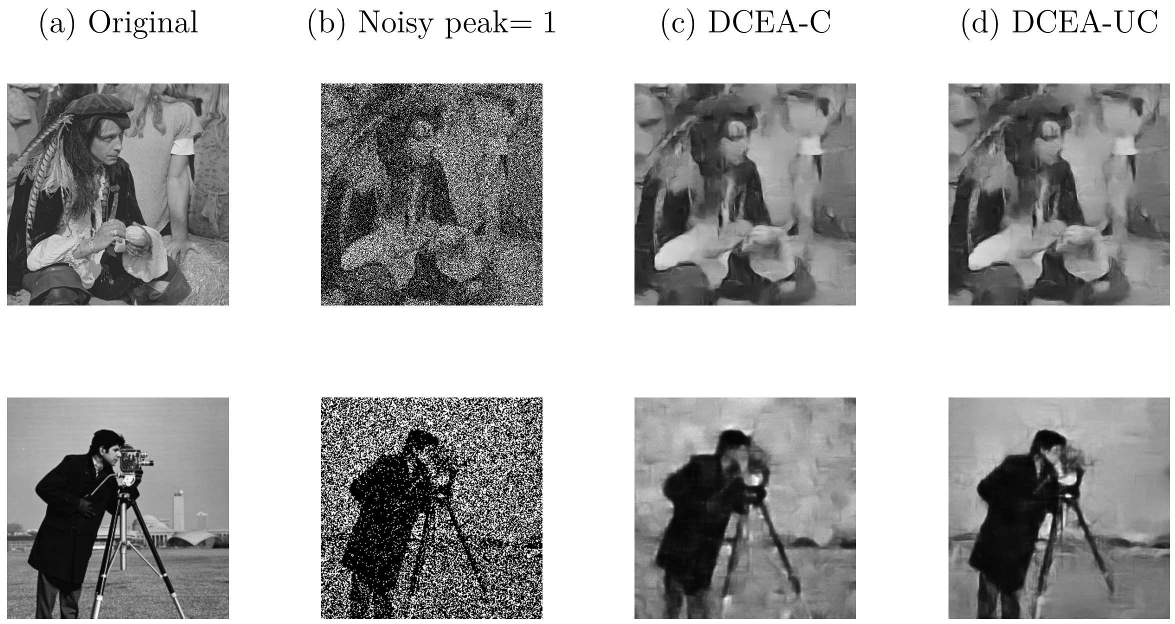

Figure 7-2: Denoising performance on test images with peak= 1. (a) Original, (b) noisy, (c) DCEA-C, and (d) DCEA-UC.

Methods We trained two versions of DCEA to assess whether relaxing the generative model, thus increasing the number of parameters, helps improve the performance: 1) DCEA constrained (DCEA-C), which uses $\mathbf{H}$ as the convolutional filters and 2) DCEA unconstrained (DCEA-UC), which uses $\mathbf{H}$, $\mathbf{W}^e$, and $\mathbf{W}^d$, as suggested in Eq. (7.6). We used $C = 169$ filters of size 11×11, where we used convolutions with strides of 7 and followed a similar approach to [193] to account for all shifts of the image when reconstructing. In terms of the number of parameters, DCEA-C has 20,618 ($= 169 \times 11 \times 11 + 169$) and DCEA-UC has 61,516 ($= 3 \times 169 \times 11 \times 11 + 169$), where the last terms refer to the bias $\mathbf{b}$. We set $\alpha = 1$.

We unfolded the encoder for $T = 15$ iterations. We initialized the filters using draws from a standard Gaussian distribution scaled by $\sqrt{1/L}$, where we approximate L using the iterative power method. We used the ADAM optimizer with an initial learning rate of 10^{-3}, which we decrease by a factor of 0.8 every 25 epochs, and trained the network for 400 epochs. At every iteration, we crop a random 128×128 patch, $\mathbf{y}^j_{\text{clean}}$, from a training image and normalize it to $\boldsymbol{\mu}_{j,\text{clean}} = \mathbf{y}^j_{\text{clean}}/Q^j$, where $Q^j = \max(\mathbf{y}^j_{\text{clean}})/\text{peak}$, such that the maximum value of $\boldsymbol{\mu}_{j,\text{clean}}$ equals the desired peak. Then, we generate a count-valued Poisson image with rate $\boldsymbol{\mu}_{j,\text{clean}}$, i.e., $\mathbf{y}^j \sim \text{Poisson}(\boldsymbol{\mu}_{j,\text{clean}})$. We minimized the mean squared error between the clean image, $\mathbf{y}^j_{\text{clean}}$, and its reconstruction, $Q^j\widehat{\boldsymbol{\mu}}_j = Q^j \exp(\mathbf{H}\mathbf{x}^j_T)$.

We compared DCEA against the following baselines. For a fair comparison, we do not use the binning strategy [181] of these methods, as a pre-processing step.

- **Sparse Poisson dictionary algorithm (SPDA)** This is a patch-based dictionary learning framework [156], using the Poisson generative model with the ℓ_0 pseudo-norm to learn the dictionary in an *unsupervised* manner, for a given noisy image. SPDA uses 400 filters of length 400, which results in 160,000 parameters.
- **BM3D + VST** BM3D is an image denoising algorithm based on a sparse representation in a transform domain, originally designed for Gaussian noise. This algorithm applies a variance-stabilizing transform (VST) to the Poisson images to make them closer to Gaussian-perturbed images [199].
- **Class-agnostic denoising network (CA)** This is a denoising residual NN for both Gaussian and Poisson images [184], trained in a *supervised* manner.

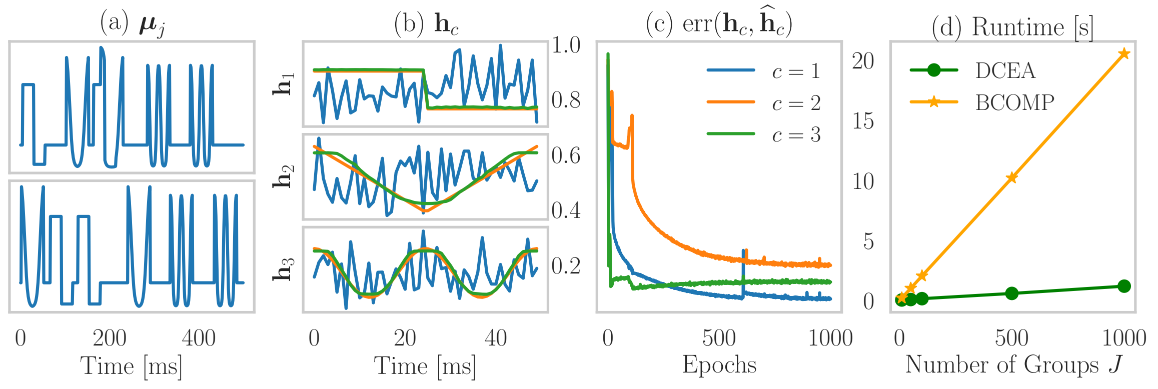

Figure 7-3: Simulated results with DCEA. (a) Example rate functions, $\boldsymbol{\mu}_j$, for two different groups. (b) Initial (blue), true (orange), and learned (green) filters for binomial data. (c) err($\mathbf{h}_c, \widehat{\mathbf{h}}_c$) over 1,000 epochs. (d) Total runtime for inference for DCEA and BCOMP.

Results Table 7.2 shows that DCEA outperforms SPDA and BM3D + VST, and shows competitive performance against CA, with an order of magnitude fewer parameters. Fig. 7-2 shows the denoising performance of DCEA-C and DCEA-UC on two test images from Set12 (see **Appendix** for more examples). We summarize a few additional points from this experiment.

- SPDA vs. DCEA-UC DCEA-UC is significantly more computationally efficient compared to SPDA. SPDA takes several minutes, or hours in some cases, to denoise a single Poisson noisy image whereas, upon training, DCEA performs denoising in less than a second.

- CA vs. DCEA-UC DCEA-UC achieves competitive performance against CA, despite an order of magnitude difference in the number of parameters (650K for CA vs. 61K for DCEA). We conjecture that given the same number of parameters, DCEA-UC would outperform CA. For example, we found that replacing the linear decoder with a nonlinear two-layer decoder ($\approx$ 120K number of parameters) in DCEA-UC resulted in an increase in PSNR of $\sim$0.2 dB.

- DCEA-C vs. DCEA-UC We also observe that DCEA-UC achieves better performance than DCEA-C. As discussed in Section 7.4.2, the relaxation of the generative model, which allows for a three-fold increase in the number of parameters, helps improve the performance.

7.5.2 Application to simulated neural spiking data

Accuracy of ECDL for DCEA

We simulated time-series of neural spiking activity from $J = 1{,}000$ neurons according to the binomial generative model. We used $C = 3$ templates of length $K = 50$ and, for each example j, generated $f(\boldsymbol{\mu}_j) = \mathbf{H}\mathbf{x}^j \in \mathbb{R}^{500}$, where each filter $\{\mathbf{h}_c\}_{c=1}^3$ appears five times uniformly random in time. Fig. 7-3(a) shows an example of two different means, $\boldsymbol{\mu}_{j_1}, \boldsymbol{\mu}_{j_2}$ for $j_1 \neq j_2$. Given $\boldsymbol{\mu}_j$, we simulated two sets of binary time-series, each with $M_j = 25$, $\mathbf{y}^j \in \{0, 1, \ldots, 25\}^{500}$, one of which is used for training and the other for validation.

Methods For DCEA, we initialized the filters using draws from a standard Gaussian, tuned the regularization parameter λ (equivalently b for $\mathcal{S}_b$) manually, and trained using the unsupervised loss. We place non-negativity constraints on $\mathbf{x}^j$ and thus use $\mathcal{S}_b = \text{ReLU}_b$. For baseline, we developed and implemented a method which we refer to as binomial convolutional orthogonal matching pursuit (BCOMP). At present, there does not exist an optimization-based framework for ECDL. Existing dictionary learning methods for non-Gaussian data are patch-based [200, 156]. BCOMP combines efficient convolutional greedy pursuit [137] and binomial greedy pursuit [168]. BCOMP solves Eq. (7.2), but uses $\|\mathbf{x}^j\|_0$ instead of $\|\mathbf{x}^j\|_1$.

Results Fig. 7-3(b) demonstrates that DCEA (green) is able to learn $\{\mathbf{h}_c\}_{c=1}^3$ accurately. Letting $\{\widehat{\mathbf{h}}_c\}_{c=1}^3$ denote the estimates, we quantify the error between a filter and its estimate using the standard measure [170], $\text{err}(\mathbf{h}_c, \widehat{\mathbf{h}}_c) = \sqrt{1 - \langle \mathbf{h}_c, \widehat{\mathbf{h}}_c \rangle^2}$, for $\|\mathbf{h}_c\| = \|\widehat{\mathbf{h}}_c\| = 1$. Fig. 7-3(c) shows the error between the true and learned filters by DCEA, as a function of epochs (we consider all possible permutations and show the one with the lowest error). The fact that the learned and the true filters match demonstrates that DCEA is indeed performing ECDL. Finally, Fig. 7-3(d) shows the runtime for both DCEA (on GPU) and BCOMP (on CPU) on CSC task, as a function of number of groups J, where DCEA is much faster. This shows that DCEA, due to 1) its simple implementation as an unrolled NN and 2) the ease with which the framework can be deployed to GPU, is an efficient/scalable alternative to optimization-based BCOMP.

Generative model relaxation for ECDL

Here, we examine whether DCEA with untied weights, which implies a departure from the original convolutional generative model, can still perform ECDL accurately. To this end, we repeat the experiment from Section 7.5.2 with DCEA-UC, whose parameters are $\mathbf{H}$, $\mathbf{W}^e$, and $\mathbf{W}^d$. Fig. 7-4 shows the learned filters, $\widehat{\mathbf{w}}_c^e$, $\widehat{\mathbf{w}}_c^d$, and $\widehat{\mathbf{h}}_c$ for $c = 1$ and 2, along with the true filters. For visual clarity, we only show the learned filters for which the distance to the true filters are the closest, among $\widehat{\mathbf{w}}_c^e$, $\widehat{\mathbf{w}}_c^d$, and $\widehat{\mathbf{h}}_c$. We observe that none of them match the true filters. In fact, the error between the learned and the true filters are bigger than the initial error.

This is in sharp contrast to the results of DCEA-C (Fig. 7-3(b)), where $\mathbf{H} =$

$\mathbf{W}^e = \mathbf{W}^d$. This shows that, to accurately perform ECDL, the NN architecture needs to be strictly constrained such that it optimizes the objective formulated from the convolutional generative model.

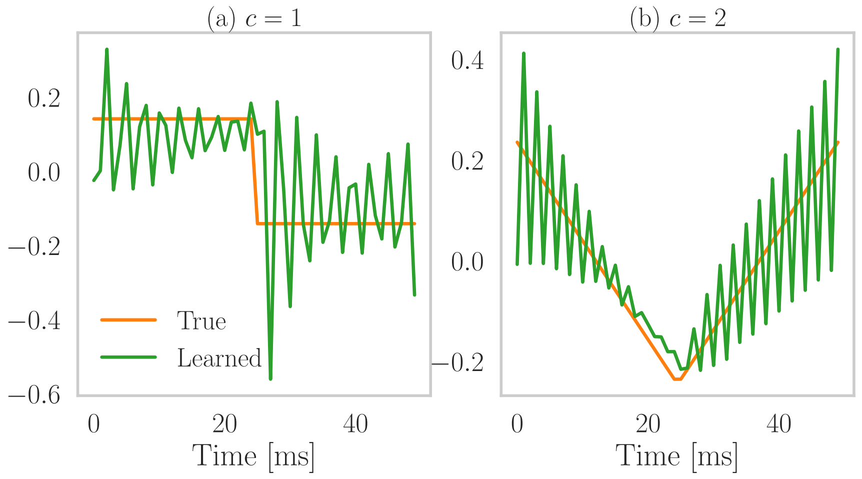

Figure 7-4: The learned (green) and true (orange) filters for DCEA-UC, when the weights are untied.

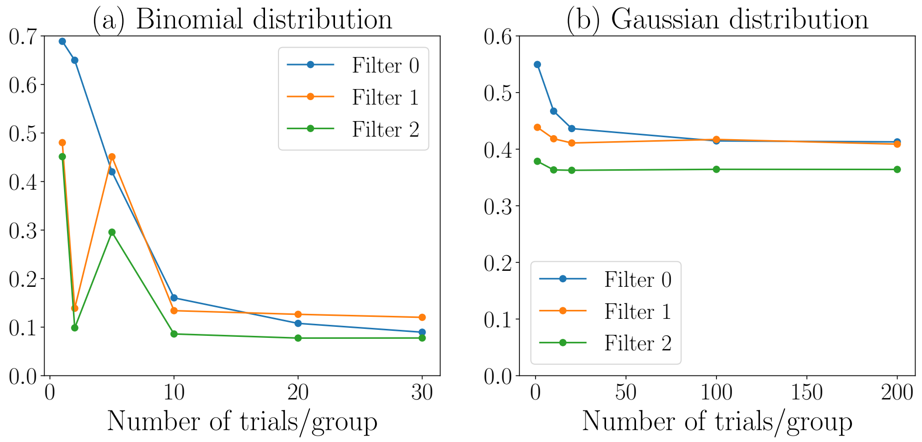

Figure 7-5: Dictionary error, $\text{err}(\mathbf{h}_c, \widehat{\mathbf{h}}_c)$, as a function of number of trials M_j per group for the (a) Binomial and (b) Gaussian models. Each point represents the median of 20 independent trials.

Effect of model mis-specification on ECDL

Here, we examine how model mis-specification in DCEA, equivalent to mis-specifying 1) the loss function (negative log-likelihood) and 2) the nonlinearity $f^{-1}(\cdot)$, affects the accuracy of ECDL. We trained two models: 1) DCEA with sigmoid link and binomial likelihood (DCEA-b), the correct model for this experiment, and 2) DCEA with linear link and Gaussian likelihood (DCEA-g). Fig. 7-5 shows how the error $\mathrm{err}(\mathbf{h}_c, \widehat{\mathbf{h}}_c)$, at convergence, changes as a function of the number of observations M_j.

We found that DCEA-b successfully recovers dictionaries for large M_j (>15). Not surprisingly, as M_j, i.e. SNR, decreases, the error increases. DCEA-g with 200 observations achieves an error close to 0.4, which is significantly worse than the 0.09 error of DCEA-b with $M_j = 30$. These results highlight the importance, for successful dictionary learning, of specifying an appropriate model. The framework we propose, DCEA, provides a flexible inference engine that can accommodate a variety of data-generating models in a seamless manner.

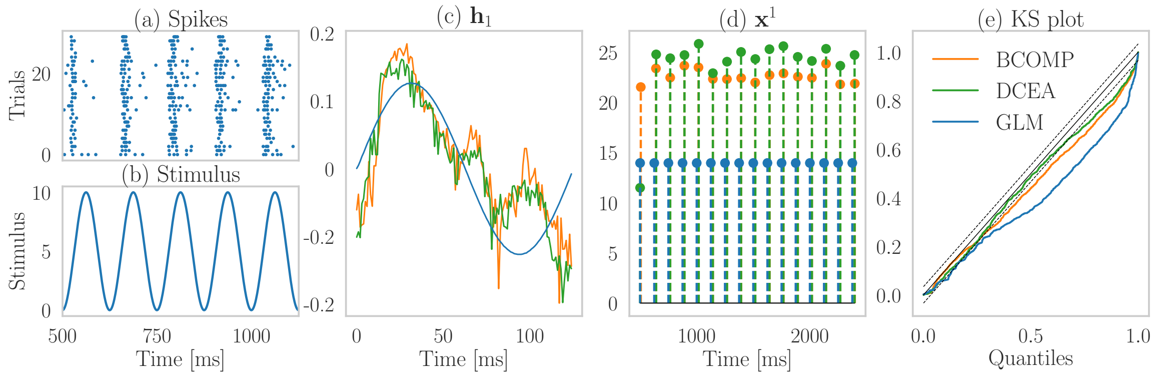

Figure 7-6: A segment of data from a neuron and result of applying DCEA and BCOMP. (a) A dot indicates a spike from the neuron. (b) Stimulus used to move the whisker. (c) Whisker velocity covariate (blue) used in GLM analysis, along with whisker velocities estimated with BCOMP (orange) and DCEA (green) using all 10 neurons in the dataset. The units are $\frac{\mathrm{mm}}{10}$ per ms. (d) The estimated sparse codes (onset of whisker deflection). (e) Analysis of Goodness-of-fit using KS plots. The dotted lines represent 95% confidence intervals.

7.5.3 Neural spiking data from somatosensory thalamus

We now apply DCEA to neural spiking data from somatosensory thalamus of rats recorded in response to periodic whisker deflections [172]. The objective is to learn the features of whisker motion that modulate neural spiking strongly. In the experiment, a piezoelectric simulator controls whisker position using an *ideal* position waveform.

As the interpretability of the learned filters is important, we constrain the weights of encoder and decoder to be $\mathbf{H}$. DECA lets us learn, in an *unsupervised* fashion, the features that best explains the data.

The dataset consists of neural spiking activity from $J = 10$ neurons in response to periodic whisker deflections. Each example j consists of $M_j = 50$ trials lasting 3,000 ms, i.e., $\mathbf{y}^{j,m} \in \mathbb{R}^{3000}$. Fig. 7-6(a) depicts a segment of data from a neuron. Each trial begins/ends with a baseline period of 500 ms. During the middle 2,000 ms, a periodic deflection with period 125 ms is applied to a whisker by the piezoelectric stimulator. There are 16 total deflections, five of which are shown in Fig. 7-6(b). The stimulus represents ideal whisker position. The blue curve in Fig. 7-6(c) depicts the whisker velocity obtained as the first derivative of the stimulus.

Methods We compare DCEA to ℓ_0-based ECDL using BCOMP (introduced in the previous section), and a generalized linear model (GLM) [163] with whisker-velocity covariate [201]. For all three methods, we let $C = 1$ and $\mathbf{h}_1 \in \mathbb{R}^{125}$, initialized using the whisker velocity (Fig. 7-6(c), blue). We set $\lambda = 0.119$ for DCEA and set the sparsity level of BCOMP to 16. As in the simulation, we used $\mathcal{S}_b = \text{ReLU}_b$ to ensure non-negativity of the codes. We used 30 trials from each neuron to learn $\mathbf{h}_1$ and the remaining 20 trials as a test set to assess goodness-of-fit.

Results The orange and green curves from Fig. 7-6(c) depict the estimates of whisker velocity computed from the neural spiking data using BCOMP and DCEA, respectively. The figure indicates that the spiking activity of this population of 10 neurons encodes well the whisker velocity, and is most strongly modulated by the maximum velocity of whisker movement.

Fig. 7-6(d) depicts the 16 sparse codes that accurately capture the onset of stimulus in each of the 16 deflection periods. The heterogeneity of amplitudes estimated by DCEA and BCOMP is indicative of the variability of the neural response to whisker deflections repeated 16 times, possibly capturing other characteristics of cellular and circuit response dynamics (e.g., adaptation). This is in sharp contrast to the GLM–

detailed in the **Appendix**–which uses the ideal whisker velocity (Fig. 7-6(c), blue) as a covariate, and assumes that neural response to whisker deflections is constant across deflections.

In Fig. 7-6(e), we use the Kolmogorov-Smirnov (KS) test to compare how well DCEA, BCOMP, and the GLM fit the data for a representative neuron in the dataset [202]. KS plots are a visualization of the KS test for assessing the Goodness-of-fit of models to point-process data, such as neural spiking data (see **Appendix** for details). The figure shows that DCEA and BCOMP are a much better fit to the data than the GLM.

We emphasize that 1) the similarity of the learned $\mathbf{h}_1$ and 2) the similar goodness-of-fit of DCEA and BCOMP to the data shows that DCEA performs ECDL. In addition, this analysis shows the power of the ECDL as an *unsupervised* and *data-driven* approach for data analysis, and a superior alternative to GLMs, where the features are hand-crafted.

7.6 Conclusion

We introduced a class of neural networks based on a generative model for convolutional dictionary learning (CDL) using data from the natural exponential-family, such as count-valued and binary data. The proposed class of networks, which we termed deep convolutional exponential auto-encoder (DCEA), is competitive compared to state-of-the-art supervised Poisson image denoising algorithms, with an order of magnitude fewer trainable parameters.

We analyzed gradient dynamics of shallow exponential-family auto-encoder (i.e., unfold the encoder once) for binomial distribution and proved that when trained with approximate gradient descent, the network recovers the dictionary corresponding to the binomial generative model.

We also showed using binomial data simulated according to the convolutional exponential-family generative model that DCEA performs dictionary learning, in an unsupervised fashion, when the parameters of the encoder/decoder are constrained.

The application of DCEA to neural spike data suggests that DCEA is superior to GLM analysis, which relies on hand-crafted covariates.

Chapter 8

Conclusion

In this thesis, we studied a systematic pipeline for conducting neural signal processing research from the generative perspective. By carefully identifying what the 1) latent variables/processes and 2) the corresponding mathematical characterizations of the domain priors should be, we demonstrated that more interpretable and better performing results could be achieved. Specifically, we focused on the smoothness of the neural dynamical processes (through Gaussian process), the smoothness of the neural patterns (also through Gaussian process), the shift-invariance of the neural patterns (through convolutional generative model), and the sparsity of the neural activations (through ℓ_0 and ℓ_1 norms).

Although some of the principles laid out in this thesis could be familiar to the readers who are well-versed in the Bayesian philosophy, we strongly believe that restating these principles rooted in the neural signal processing context provides a unique and interesting perspective for neuroscience. We hope that this thesis, which not only contains scientific/engineering results but also insights/lessons that have been accrued over several years of hard work and numerous failures, serves as a guideline to prospective students who want to research in the field of neural signal processing.

Appendix A

Appendix for Time-frequency analysis with Gaussian Process

A.1 Continuous model interpretation of PLSO

We can establish the equivalent continuous model of the PLSO in Eq. 4.2, using stochastic different equation

$$\frac{d\tilde{z}_j(t)}{dt} = \underbrace{\left(\left(-\frac{1}{l_j}\right) \oplus \begin{pmatrix} 0 & -\omega_j \\ \omega_j & 0 \end{pmatrix}\right)}_{\mathbf{F}} \tilde{z}_j(t) + \varepsilon(t), \tag{A.1}$$

where $\tilde{z}_j(t) : \mathbb{R} \to \mathbb{R}^2$, $\oplus$ denotes the Kronecker sum and $\varepsilon(t) \sim \mathcal{N}(0, \sigma_j^2 \mathbf{I}_{2\times 2})$. Discretizing the solution of Eq. A.1 at Δ, such that $\widetilde{\mathbf{z}}_{j,k} = \tilde{z}_j(k\Delta)$, yields Eq. 4.2. Consequently, we obtain the following for $\Delta > 0$

$$\begin{aligned} &\exp(\mathbf{F}\Delta) = \exp(-\Delta/l_j)\mathbf{R}(\omega_j), \\ &\sigma_j^2 \int_0^\Delta \exp(\mathbf{F}(\Delta-\tau)) \exp(\mathbf{F}(\Delta-\tau))^{\mathrm{T}} d\tau = \sigma_j^2 \left(1 - \exp(-2\Delta/l_j)\right) \mathbf{I}_{2\times 2}. \end{aligned}$$

This interpretation extends to the nonstationary PLSO. The corresponding continuous model for $\widetilde{\mathbf{z}}_{j,mN+n}$ in Eq. 4.3 is the same as Eq. A.1, with different variance $\mathbb{E}[\varepsilon_j(t)\varepsilon_j^{\mathrm{T}}(t)] = \sum_{m=1}^{M} \sigma_{j,m}^2 \cdot \mathbf{1}\left(\left(\frac{m-1}{M}\right) T \leq t < \left(\frac{m}{M}\right) T\right) \mathbf{I}_{2\times 2}$.

A.2 PSD for complex AR(1) process

We compute the steady-state covariance denoted as $\mathbf{P}_\infty^j$. Since we assume $\mathbf{P}_1^j = \sigma_j^2 \mathbf{I}_{2\times 2}$, it is easy to show that $\mathbf{P}_k^j$ is a diagonal matrix from $\mathbf{R}(\omega_j)\mathbf{R}^{\mathrm{T}}(\omega_j) = \mathbf{I}_{2\times 2}$. Denoting $\mathbf{P}_\infty^j = \alpha \mathbf{I}_{2\times 2}$, we use the discrete Lyapunov equation

$$
\begin{aligned}
&\mathbf{P}_\infty^j = \exp(-2\Delta/l_j)\mathbf{R}(\omega_j)\mathbf{P}_\infty^j \mathbf{R}^{\mathrm{T}}(\omega_j) + \sigma_j^2 \left(1 - \exp\left(-2\Delta/l_j\right)\right) \mathbf{I}_{2\times 2} \\
&\Rightarrow \alpha = \exp(-2\Delta/l_j)\alpha + \sigma_j^2 \left(1 - \exp\left(-2\Delta/l_j\right)\right) \\
&\Rightarrow \mathbf{P}_\infty^j = \sigma_j^2 \mathbf{I}_{2\times 2},
\end{aligned}
$$

which implies that under the assumption $\mathbf{P}_1^j = \sigma_j^2 \mathbf{I}_{2\times 2}$, we are guaranteed $\mathbf{P}_k^j = \sigma_j^2 \mathbf{I}_{2\times 2}$, $\forall k$. To compute the PSD of the stationary process $\mathbf{z}_j$, we now need to compute the autocovariance. Since only $\mathbf{z}_{j,k}^{\Re}$ contributes to $\mathbf{y}_k$, we compute the autocovariance of $\mathbb{E}[\mathbf{z}_{j,k}^{\Re}\mathbf{z}_{j,k+n}^{\Re}]$ as

$$
\begin{aligned}
\mathbb{E}[\mathbf{z}_{j,k}^{\Re}\mathbf{z}_{j,k+n}^{\Re}] &= \mathbb{E}[\mathbf{z}_{j,k}^{\Re} \cdot \Re(\rho_j^n \exp(i\omega_j n)\mathbf{z}_{j,k})] \\
&= \rho_j^n \mathbb{E}[\mathbf{z}_{j,k}^{\Re}\mathbf{z}_{j,k}^{\Re} \cos \omega_j n] = \rho_j^n \cos \omega_j n \cdot \mathbb{E}[\{\mathbf{z}_{j,k}^{\Re}\}^2] \\
&= \rho_j^n \sigma_j^2 \cos w_j n,
\end{aligned}
$$

where $\Re(\cdot)$ denotes the operator that extracts the real part of the complex argument and we used the fact that $\mathbb{E}[\mathbf{z}_{j,k}^{\Re}\mathbf{z}_{j,k}^{\Im}] = 0$. The spectra for the j^{th} component, $S_j(\omega)$ can be written as

$$
\begin{aligned}
S_j(\omega) &= \sum_{n=-\infty}^{\infty} \mathbb{E}\left[\mathbf{z}_{j,k}^{\Re}\mathbf{z}_{j,k+n}^{\Re}\right] \exp\left(-i\omega n\right) \\
&= \sum_{n=-\infty}^{\infty} \rho_j^n \sigma_j^2 \cos w_j n \exp\left(-i\omega n\right) \\
&= \sigma_j^2 \sum_{n=-\infty}^{\infty} \rho_j^n \left\{\exp(i\omega_j n) + \exp(-i\omega_j n)\right\} \exp\left(-i\omega n\right) \\
&= \sigma_j^2 \sum_{n=-\infty}^{\infty} \rho_j^n \exp(-i(\omega \pm \omega_j)n).
\end{aligned}
$$

Unpacking the infinite sum for one of the terms,

$$\begin{aligned}
\sum_{n=-\infty}^{\infty} \rho_j^n \exp(-i(\omega-\omega_j)n) &= 1 + \sum_{n=1}^{\infty} \rho_j^n \exp(-i(\omega-\omega_j)n) + \rho_j^n \exp(i(\omega-\omega_j)n) \\
&= 1 + \frac{\rho_j \exp(-i(\omega-\omega_j))}{1-\rho_j \exp(-i(\omega-\omega_j))} + \frac{\rho_j \exp(i(\omega-\omega_j))}{1-\rho_j \exp(i(\omega-\omega_j))} \\
&= 1 + \frac{2\rho_j \cos(\omega-\omega_j) - 2\rho_j^2}{(1-\rho_j \exp(-i(\omega-\omega_j)))\,(1-\rho_j \exp(i(\omega-\omega_j)))} \\
&= \frac{1-\rho_j^2}{1+\rho_j^2 - 2\rho_j \cos(\omega-\omega_j)}.
\end{aligned}$$

Using the relation $\rho_j = \exp(-\Delta/l_j)$ and unpacking the infinite sum for the other term, we have

$$\begin{aligned}
S_j(w) = &\frac{\sigma_j^2(1-\exp(-2\Delta/l_j))}{1+\exp(-2\Delta/l_j) - 2\exp(-\Delta/l_j)\cos(\omega-\omega_j)} \\
&+ \frac{\sigma_j^2(1-\exp(-2\Delta/l_j))}{1+\exp(-2\Delta/l_j) - 2\exp(-\Delta/l_j)\cos(\omega+\omega_j)}.
\end{aligned}$$

Since Fourier transform is a linear operator, we can conclude that $\gamma(\omega) = \sigma_\nu^2 + \sum_{j=1}^{J} S_j(\omega)$.

Appendix B

Appendix for model-based deep learning

B.1 Generalized linear model (GLM) for whisker experiment

In this section, for ease of notation, we consider the simple case of $M_j = 1$ (Bernoulli). However, the detail can be generalized to the binomial generative model.

We describe the GLM [17] used for analyzing the neural spiking data from the whisker experiment [201], and which we compared to BCOMP and DCEA in Fig. 4. Fig. 4(b) depicts a segment of the periodic stimulus used in the experiment to deflect the whisker. The units are in $\frac{\text{mm}}{10}$. The full stimulus lasts 3000 ms and is equal to zero (whisker at rest) during the two baseline periods from 0 to 500 ms and 2500 to 3000 ms. In the GLM analysis, we used whisker velocity as a stimulus covariate, which corresponds to the first difference of the position stimulus $\mathbf{s} \in \mathbb{R}^{3000}$. The blue curve in Fig. 4(c) represents one period of the whisker-velocity covariate. We associated a single stimulus coefficient $\boldsymbol{\beta}_{\text{stim}} \in \mathbb{R}$ to this covariate. In addition to the stimulus covariate, we used history covariates in the GLM. We denote by $\boldsymbol{\beta}_H^j \in \mathbb{R}^{L_j}$ the coefficients associated with these covariates, where $j = 1, \cdots, J$ is the neuron index. We also define a^j to be the base firing rate for neuron j. The GLM is given by

$$\begin{aligned} &\mathbf{y}^j[n] \sim \text{Bernoulli}(\mathbf{p}^j[n]) \\ &\text{s.t. } \mathbf{p}^j[n] = \Big(1 + \exp\Big(- a^j - \boldsymbol{\beta}_{\text{stim}} \cdot \underbrace{(\mathbf{s}[n] - \mathbf{s}[n-1])}_{\text{whisker velocity}} - \sum_{l=1}^{L_j} \boldsymbol{\beta}_H^j[l] \cdot \mathbf{y}^j[n-l]\Big)\Big)^{-1} \end{aligned} \tag{B.1}$$

The parameters $\{a^j\}_{j=1}^J$, $\boldsymbol{\beta}_{\text{stim}}$, and $\{\boldsymbol{\beta}_H^j\}_{j=1}^J$ are estimated by minimizing the negative likelihood of the neural spiking data $\{\mathbf{y}^j\}_{j=1}^{10}$ with $M_j = 30$ *from all neurons* using IRLS. We picked the order L_j (in ms) of the history effect for neuron j by fitting the GLM to each of the 10 neurons *separately* and finding the value of $\approx 5 \leq L_j \leq 100$ that minimizes the Akaike Information Criterion [17].

Interpretation of the GLM as a convolutional model Because whisker position is periodic with period 125 ms, so is whisker velocity. Letting $\mathbf{h}_1$ denote whisker velocity in the interval of length 125 ms starting at 500 ms (blue curve in Fig. 4(c)), we can interpret the GLM in terms of the convolutional model of Eq. 8. In this interpretation, $\mathbf{H}$ is the convolution matrix associated with the *fixed* filter $\mathbf{h}_1$ (blue curve in Fig. 4(c)), and $\mathbf{x}^j$ is a sparse vector with 16 equally spaced nonzero entries all equal to $\boldsymbol{\beta}_{\text{stim}}$. The first nonzero entry of $\mathbf{x}^j$ occurs at index 500. The number of indices between nonzero entries is 125. The blue dots in Fig. 4(d) reflect this interpretation.

Incorporating history dependence in the generative model GLMs of neural spiking data [17] include a constant term that models the baseline probability of spiking a^j, as well as a term that models the effect of spiking history. This motivates us to use the model

$$\log \frac{p(\mathbf{y}^{j,m} \mid \{\mathbf{h}_c\}_{c=1}^C, \mathbf{x}^j, \mathbf{x}_H^j)}{1 - p(\mathbf{y}^{j,m} \mid \{\mathbf{h}_c\}_{c=1}^C, \mathbf{x}^j, \mathbf{x}_H^j)} = a^j + \mathbf{H}\mathbf{x}^j + \mathbf{Y}_j \mathbf{x}_H^j, \tag{B.2}$$

where $\mathbf{y}^{j,m} \in \{0,1\}^N$ refers to m^{th} trial of the binomial data $\mathbf{y}^j$. The n^{th} row of $\mathbf{Y}_j \in \mathbb{R}^{N \times L_j}$ contains the spiking history of neuron j at trial m from $n - L_j$ to n, and $\mathbf{x}_H^j \in \mathbb{R}^{L_j}$ are coefficients that capture the effect of spiking history on the propensity of neuron j to spike. We use the same L_j estimated from GLM. We estimate a^j from the average firing probability during the baseline period. The addition of the history term simply results in an additional set of variables to alternate over in the alternating-minimization interpretation of ECDL. We estimate it by adding a loop around BCOMP or backpropagation through DCEA. Every iteration of this loop first assumes $\mathbf{x}_H^j$ are fixed. Then, it updates the filters and $\mathbf{x}^j$. Finally, it solves a convex optimization problem to update $\mathbf{x}_H^j$ given the filters and $\mathbf{x}^j$. In the interest of space, we do not describe this algorithm formally.

B.2 Kolmogorov-smirnov plots and the time-rescaling theorem

Loosely, the time-rescaling theorem states that rescaling the inter-spike intervals (ISIs) of the neuron using the (unknown) underlying conditional intensity function (CIF) will transform them into i.i.d. samples from an exponential random variable with rate 1. This implies that, if we apply the CDF of an exponential random variable with rate 1 to the rescaled ISIs, these should look like i.i.d. draws from a uniform random variable in the interval $[0, 1]$. KS plots are a visual depiction of this result. They are obtained by computing the rescaled ISIs using an estimate of the underlying CIF and applying the CDF of an exponential random variable with rate 1 to them. These are then sorted and plotted against ideal uniformly-spaced empirical quantiles from a uniform random variable in the interval $[0, 1]$. The CIF that fits the data the best is the one that yields a curve that is the closest to the 45-degree diagonal. Fig. 4(e) depicts the KS plots obtained using the CIFs estimated using DCEA, BCOMP and the GLM.

Bibliography

[1] EN Brown, R Kass, and Mitra P. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuroscience*, 7:456–461, 2004.

[2] Marius Pachitariu, Nicholas A. Steinmetz, Shabnam N. Kadir, Matteo Carandini, and Kenneth D. Harris. Fast and accurate spike sorting of high-channel count probes with KiloSort. In *Advances in Neural Information Processing Systems 30*, 2016.

[3] Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, Colleen J Gillon, Danijar Hafner, Adam Kepecs, Nikolaus Kriegeskorte, Peter Latham, Grace W Lindsay, Kenneth D Miller, Richard Naud, Christopher C Pack, Panayiota Poirazi, Pieter Roelfsema, João Sacramento, Andrew Saxe, Benjamin Scellier, Anna C Schapiro, Walter Senn, Greg Wayne, Daniel Yamins, Friedemann Zenke, Joel Zylberberg, Denis Therien, and Konrad P Kording. A deep learning framework for neuroscience. *Nature Neuroscience*, 22(11):1761–1770, 2019.

[4] Christoph M. Michel and Denis Brunet. Eeg source imaging: A practical review of the analysis steps. *Frontiers in Neurology*, 10:325, 2019.

[5] Malcolm Proudfoot, Mark W Woolrich, Anna C Nobre, and Martin R Turner. Magnetoencephalography. *Practical Neurology*, 14(5):336–343, 2014.

[6] Raichle ME Fox MD. Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging. *Nature Reviews Neuroscience*, 8:700–711, 2007.

[7] Grienberger C. and Konnerth A. Imaging calcium in neurons. *Neuron*, 73:862–885, 2012.

[8] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. Past, present and future of spike sorting techniques. *Brain Research Bulletin*, 119:106–117, 2015.

[9] Caterina Cinel, Davide Valeriani, and Riccardo Poli. Neurotechnologies for human cognitive augmentation: Current state of the art and future prospects. *Frontiers in Human Neuroscience*, 13:13, 2019.

[10] David A. Moses, Sean L. Metzger, Jessie R. Liu, Gopala K. Anumanchipalli, Joseph G. Makin, Pengfei F. Sun, Josh Chartier, Maximilian E. Dougherty, Patricia M. Liu, Gary M. Abrams, Adelyn Tu-Chan, Karunesh Ganguly, and Edward F. Chang. Neuroprosthesis for decoding speech in a paralyzed person with anarthria. *New England Journal of Medicine*, 385(3):217–227, 2021.

[11] Robert E Kass, Uri Eden, and Emery N Brown. *Analysis of neural data.* Springer series in statistics. Springer, 2014.

[12] Chaitanya Ekanadham, Daniel Tranchina, and Eero P. Simoncelli. A unified framework and method for automatic neural spike identification. *Journal of Neuroscience Methods*, 222:47–55, 2014.

[13] A. H. Song, F. J. Flores, and D. Ba. Convolutional dictionary learning with grid refinement. *IEEE Transactions on Signal Processing*, 68:2558–2573, 2020.

[14] Martin A Lindquist, Ji Meng Loh, Lauren Y Atlas, and Tor D Wager. Modeling the hemodynamic response function in fMRI: efficiency, bias and mis-modeling. *NeuroImage*, 45(1 Suppl):S187–S198, mar 2009.

[15] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics. Springer New York Inc., 2009.

[16] Anne C. Smith and Emery N. Brown. Estimating a State-Space Model from Point Process Observations. *Neural Computation*, 15(5):965–991, 05 2003.

[17] Wilson Truccolo, Uri T Eden, Matthew Fellows, John Donoghue, and Emery N. Brown. A Point Process Framework for Relating Neural Spiking Activity to Spiking History, Neural Ensemble, and Extrinsic Covariate Effects. *Journal of Neurophysiology*, 93(2):1074–1089, 2005.

[18] D. Ba, B. Babadi, P. L. Purdon, and E. N. Brown. Robust spectrotemporal decomposition by iteratively reweighted least squares. *Proceedings of the National Academy of Sciences*, 111(50):E5336–E5345, 2014.

[19] S. Kim, M. K. Behr, D. Ba, and E. N. Brown. State-space multitaper time-frequency analysis. *Proceedings of the National Academy of Sciences*, 115(1):E5–E14, 2018.

[20] A. H. Song, D. Ba, and E. N. Brown. PLSO: A generative framework for decomposing nonstationary time-series into piecewise stationary oscillatory components. *Uncertainty in Artificial Intelligence (UAI)*, 2021.

[21] Anne C. Smith, Loren M. Frank, Sylvia Wirth, Marianna Yanike, Dan Hu, Yasuo Kubota, Ann M. Graybiel, Wendy A. Suzuki, and Emery N. Brown. Dynamic analysis of learning in behavioral experiments. *Journal of Neuroscience*, 24(2):447–461, 2004.

[22] Noa Malem-Shinitski, Yingzhuo Zhang, Daniel T. Gray, Sara N. Burke, Anne C. Smith, Carol A. Barnes, and Demba Ba. A separable two-dimensional random field model of binary response data from multi-day behavioral experiments. *Journal of Neuroscience Methods*, 307:175–187, 2018.

[23] Michael J. Prerau, Ritchie E. Brown, Matt T. Bianchi, Jeffrey M. Ellenbogen, and Patrick L. Purdon. Sleep neurophysiological dynamics through the lens of multitaper spectral analysis. *Physiology*, 32(1):60–92, 2017.

[24] Andrew H. Song, Leon Chlon, Hugo Soulat, John Tauber, Sandya Subramanian, Demba Ba, and Michael J. Prerau. Multitaper infinite hidden markov model for eeg. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5803–5807, 2019.

[25] Emery N. Brown, Ralph Lydic, and Nicholas D. Schiff. General anesthesia, sleep, and coma. *New England Journal of Medicine*, 363(27):2638–2650, 2010.

[26] P. L. Purdon, E. T. Pierce, E. A. Mukamel, M. J. Prerau, J. L. Walsh, K. F. K. Wong, A. F. Salazar-Gomez, P. G. Harrell, A. L. Sampson, A. Cimenser, S. Ching, N. J. Kopell, C. Tavares-Stoeckel, K. Habeeb, R. Merhar, and E. N. Brown. Electroencephalogram signatures of loss and recovery of consciousness from propofol. *Proceedings of the National Academy of Sciences*, 110(12):E1142–E1151, 2013.

[27] Andrew H. Song, Aaron Kucyi, Vitaly Napadow, Emery N. Brown, Marco L. Loggia, and Oluwaseun Akeju. Pharmacological modulation of noradrenergic arousal circuitry disrupts functional connectivity of the locus ceruleus in humans. *Journal of Neuroscience*, 37(29):6938–6945, 2017.

[28] Alexander Lin, Yingzhuo Zhang, Jeremy Heng, Stephen A. Allsop, Kay M. Tye, Pierre E. Jacob, and Demba Ba. Clustering time series with nonlinear dynamics: A bayesian non-parametric and particle-based approach. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2476–2484. PMLR, 16–18 Apr 2019.

[29] Lecun Y., Bengio Y., and Hinton G. Deep learning. *Nature*, 521:436–444, 2015.

[30] Christopher M Bishop. *Pattern recognition and machine learning.* New York : Springer, 2006.

[31] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[32] Yuanjun Gao, Evan W Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[33] Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, Jaimie M Henderson, Krishna V Shenoy, L F Abbott, and David Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, 2018.

[34] Qi She and Anqi Wu. Neural dynamics discovery via gaussian process recurrent neural networks. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 454–464. PMLR, 22–25 Jul 2020.

[35] Yuanjun Gao, Lars Busing, Krishna V Shenoy, and John P Cunningham. High-dimensional neural spike train analysis with generalized count linear dynamical systems. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[36] Ludwig Fahrmeir and Gerhard Tutz. *Multivariate statistical modelling based on generalized linear models*. Springer Science & Business Media, 2013.

[37] Eftychios A. Pnevmatikakis, Daniel Soudry, Yuanjun Gao, Timothy A. Machado, Josh Merel, David Pfau, Thomas Reardon, Yu Mu, Clay Lacefield, Weijian Yang, Misha Ahrens, Randy Bruno, Thomas M. Jessell, Darcy S. Peterka, Rafael Yuste, and Liam Paninski. Simultaneous denoising, deconvolution, and demixing of calcium imaging data. *Neuron*, 89(2):285–299, 2016.

[38] Hastie T., Tibshirani R., and Wainwright M. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman and Hall/CRC, 2015.

[39] Alexandre Gramfort, Matthieu Kowalski, and Matti Hämäläinen. Mixed-norm estimates for the M/EEG inverse problem using accelerated gradient methods. *Physics in medicine and biology*, 57(7):1937–1961, apr 2012.

[40] Christoph M. Michel and Denis Brunet. Eeg source imaging: A practical review of the analysis steps. *Frontiers in Neurology*, 10:325, 2019.

[41] Michael Lewicki and Terrence J Sejnowski. Coding time-varying signals using sparse, shift-invariant representations. In *Advances in Neural Information Processing Systems*, volume 11, 1999.

[42] Grace Wahba. Automatic smoothing of the log periodogram. *Journal of the American Statistical Association*, 75(369):122–132, 1980.

[43] Michael Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78, 1998.

[44] Daniel Sage, Hagai Kirshner, Thomas Pengo, Nico Stuurman, Junhong Min, Suliana Manley, and Michael Unser. Quantitative evaluation of software packages for single-molecule localization microscopy. *Nature Methods*, 12:717–724, 2015.

[45] Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of Computational Neuroscience*, 29(1):107–126, 2010.

[46] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[47] Gabriela Czanner, Uri T. Eden, Sylvia Wirth, Marianna Yanike, Wendy A. Suzuki, and Emery N. Brown. Analysis of between-trial and within-trial neural spiking dynamics. *Journal of Neurophysiology*, 99(5):2672–2693, 2008.

[48] Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 914–922. PMLR, 20–22 Apr 2017.

[49] Yingzhuo Zhang, Noa Malem-Shinitski, Stephen A. Allsop, Kay M. Tye, and Demba Ba. Estimating a Separably Markov Random Field from Binary Observations. *Neural Computation*, 30(4):1046–1079, 04 2018.

[50] Yuan Zhao and Il Memming Park. Variational latent gaussian process for recovering single-trial dynamics from population spike trains. *Neural Computation*, 29(5):1293–1316, 2017.

[51] Joshua T. Vogelstein, Adam M. Packer, Timothy A. Machado, Tanya Sippy, Baktash Babadi, Rafael Yuste, and Liam Paninski. Fast nonnegative deconvolution for spike train inference from population calcium imaging. *Journal of Neurophysiology*, 104(6):3691–3704, 2010.

[52] 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.

[53] Karol Gregor and Yann Lecun. Learning fast approximations of sparse coding. In *International Conference on Machine Learning*, pages 399–406, 2010.

[54] Gabriel Schamberg, Demba Ba, and Todd P. Coleman. A modularized efficient framework for non-markov time series estimation. *IEEE Transactions on Signal Processing*, 66(12):3140–3154, 2018.

[55] R. G. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 2101–2109. AAAI Press, 2017.

[56] A. H. Song, S. Chakravarty, and E. N. Brown. A smoother state space multitaper spectrogram. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 33–36, July 2018.

[57] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

[58] Andrew Song, Bahareh Tolooshams, and Demba Ba. Gaussian process convolutional dictionary learning. *IEEE Signal Processing Letters*, pages 1–1, 2021.

[59] Bahareh Tolooshams, Andrew Hyungsuk Song, Simona Temereanca, and Demba Ba. Convolutional dictionary learning based auto-encoders for natural exponential-family distributions. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9493–9503, 2020.

[60] Alexander Lin, Andrew H. Song, and Demba Ba. Mixture model auto-encoders: Deep clustering through dictionary learning, 2021.

[61] RH Shumway and DS Stoffer. *Times Series Analaysis and Its Applications with R Examples*. Springer, 2010.

[62] Behtash Babadi and Emery N Brown. A review of multitaper spectral analysis. *IEEE Transactions on Biomedical Engineering*, 61(5):1555–1564, 2014.

[63] David J Thomson. Spectrum estimation and harmonic analysis. *Proceedings of the IEEE*, 70(9):1055–1096, 1982.

[64] G Wahba. Improper priors, spline smoothing and the problem of guarding against model errors in regression. *J. of the Royal Stat. Soc. Series B*, 40(3), 1978.

[65] Y Bar-Shalom, XR Li, and T Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software.* Wiley-Interscience, 2001.

[66] AP Dempster, NM Laird, and DB Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of Royal Stat. Soc. Series B Statistical Methodology*, 39(1):1–38, 1977.

[67] Patrick L Purdon, Aaron Sampson, Kara J Pavone, and Emery N Brown. Clinical electroencephalography for anesthesiologists: Part I: background and basic signatures. *Anesthesiology: The Journal of the American Society of Anesthesiologists*, 123(4):937–960, 2015.

[68] S. Chakravarty, K. Nikolaeva, D. Kishnan, F. J. Flores, P. L. Purdon, and E. N. Brown. Pharmacodynamic modeling of propofol-induced general anesthesia in young adults. In *IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT)*, pages 44–47, Nov 2017.

[69] S. S. Haykin and A. O. Steinhardt. *Adaptive Radar Detection and Estimation.* Wiley-Interscience, 1992.

[70] T. F. Quatieri. *Discrete-Time Speech Signal Processing: Principles and Practice.* Prentice-Hall, 2008.

[71] J. S. Lim. *Two-Dimensional Signal and Image Processing.* Prentice-Hall, 1990.

[72] W. J. Emery and R. E. Thomson. *Data Analysis Methods in Physical Oceanography.* Elsevier, 2001.

[73] P. Mitra and H. Bokil. *Observed Brain Dynamics.* Oxford Univ Press, 2007.

[74] R. Dahlhaus. Fitting time series models to nonstationary processes. *The Annals of Statistics*, 25(1):1 – 37, 1997.

[75] T. H. Koornwinder. *Wavelets : An Elementary Treatment of Theory and Applications.* World Scientific, 1995.

[76] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-time Signal Processing.* Prentice-Hall, Inc., 2009.

[77] D. B. Percival and A. T. Walden. *Spectral Analysis for Physical Applications.* Cambridge Univ Press, 1993.

[78] S. Kim, M. K. Behr, D. Ba, and E. N. Brown. State-space multitaper time-frequency analysis. *Proc. Natl. Acad. Sci. USA*, 115(1):E5–E14, 2018.

[79] Demba Ba, Behtash Babadi, Patrick L. Purdon, and Emery N. Brown. Robust spectrotemporal decomposition by iteratively reweighted least squares. *Proceedings of the National Academy of Sciences*, 111(50):E5336–E5345, 2014.

[80] Proloy Das and Behtash Babadi. Dynamic bayesian multitaper spectral analysis. *IEEE Transactions on Signal Processing*, 66(6):1394–1409, 2018.

[81] Patrick L. Purdon, Eric T. Pierce, Eran A. Mukamel, Michael J. Prerau, John L. Walsh, Kin Foon K. Wong, Andres F. Salazar-Gomez, Priscilla G. Harrell, Aaron L. Sampson, Aylin Cimenser, ShiNung Ching, Nancy J. Kopell, Casie Tavares-Stoeckel, Kathleen Habeeb, Rebecca Merhar, and Emery N. Brown. Electroencephalogram signatures of loss and recovery of consciousness from propofol. *Proceedings of the National Academy of Sciences*, 110(12):E1142–E1151, 2013.

[82] 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)*, 39(1):1–22, 1977.

[83] S. Adak. Time-dependent spectral analysis of nonstationary time series. *Journal of the American Statistical Association*, 93(444):1488–1501, 1998.

[84] D. R. Brillinger. *Time Series: Data Analysis and Theory.* SIAM, 1981.

[85] B. Babadi and E. N. Brown. A review of multitaper spectral analysis. *IEEE Trans. Biomed. Eng.*, 61(5):1555–1564, 2014.

[86] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications.* Springer, 2017.

[87] John G. Proakis and Dmitris K. Manolakis. *Digital Signal Processing.* Prentice-Hall, Inc., 2006.

[88] F. Itakura and S. Saito. A statistical method for estimation of speech spectral density and formant frequencies. 1970.

[89] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time Signal Processing (3rd Ed.).* Prentice-Hall, Inc., 2009.

[90] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. Yen, C. C. Tung, and H. H. Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1971):903–995, 1998.

[91] I. Daubechies, J. Lu, and H. Wu. Synchrosqueezed wavelet transforms: An empirical mode decomposition-like tool. *Applied and Computational Harmonic Analysis*, 30(2):243 – 261, 2011.

[92] K. W. Wilson, B. Raj, P. Smaragdis, and A. Divakaran. Speech denoising using nonnegative matrix factorization with priors. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4029–4032, 2008.

[93] D. Griffin and J. Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[94] A. T. Cemgil and S. J. Godsill. Probabilistic phase vocoder and its application to interpolation of missing values in audio signals. In *2005 13th European Signal Processing Conference*, pages 1–4, 2005.

[95] T. Matsuda and F. Komaki. Time series decomposition into oscillation components and phase estimation. *Neural Computation*, 29(2):332–367, 2017.

[96] A. M. Beck, E. P. Stephen, and P. L. Purdon. State space oscillator models for neural data analysis. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4740–4743, 2018.

[97] A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1067–III–1075, 2013.

[98] W. J. Wilkinson, M. Riis Andersen, J. D. Reiss, D. Stowell, and A. Solin. Unifying probabilistic models for time-frequency analysis. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3352–3356, May 2019.

[99] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 1960.

[100] H. Li and Z. Lin. Accelerated proximal gradient methods for nonconvex programming. In *Advances in Neural Information Processing Systems 28*, pages 379–387. Curran Associates, Inc., 2015.

[101] M. B. Priestley. Evolutionary spectra and non-stationary processes. *Journal of the Royal Statistical Society. Series B (Methodological)*, 27(2):204–237, 1965.

[102] R. Dahlhaus. Fitting time series models to nonstationary processes. *Ann. Statist.*, 25(1):1–37, 02 1997.

[103] Y. Qi, T. P. Minka, and R. W. Picard. Bayesian spectrum estimation of unevenly sampled nonstationary data. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–1473–II–1476, 2002.

[104] S. J. Koopman and K. M. Lee. Seasonality with trend and cycle interactions in unobserved components models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 58(4):427–448, 2009.

[105] E. N. Brown, V. Solo, Y. Choe, and Z. Zhang. Measuring period of human biological clock: infill asymptotic analysis of harmonic regression parameter estimates. *Methods in enzymology*, 383:382–405, 2004.

[106] A. Solin and S. Särkkä. Explicit Link Between Periodic Covariance Functions and State Space Models. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 904–912. PMLR, 2014.

[107] O. Rosen, D. S. Stoffer, and S. Wood. Local spectral analysis via a bayesian mixture of smoothing splines. *Journal of the American Statistical Association*, 104(485):249–262, 2009.

[108] P. Das and B. Babadi. Dynamic bayesian multitaper spectral analysis. *IEEE Transactions on Signal Processing*, 66(6):1394–1409, 2018.

[109] H. Soulat, E. P. Stephen, A. M. Beck, and P. L. Purdon. State space methods for phase amplitude coupling analysis. *bioRxiv*, 2019.

[110] G. Casella. An introduction to empirical bayes data analysis. *The American Statistician*, 39(2):83–87, 1985.

[111] P. Whittle. Estimation and information in stationary time series. *Ark. Mat.*, 2(5):423–434, 08 1953.

[112] R. E. Turner and M. Sahani. Time-frequency analysis as probabilistic inference. *IEEE Transactions on Signal Processing*, 62(23):6171–6183, 2014.

[113] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151:3–34, 2015.

[114] J. Barzilai and J. M. Borwein. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 01 1988.

[115] C. K. Carter and R. Kohn. On gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

[116] Hirotugu Akaike. Likelihood of a model and information criteria. *Journal of Econometrics*, 16(1):3–14, 1981.

[117] B. Gold, N. Morgan, and D. Ellis. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music.* Wiley-Interscience, USA, 2nd edition, 2011.

[118] R. B. Gramacy and H. K. H. Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.

[119] A. Solin, J. Hensman, and R. E Turner. Infinite-horizon gaussian processes. In *Advances in Neural Information Processing Systems 31*, pages 3486–3495. 2018.

[120] G. Kitagawa and W. Gersch. A smoothness priors time-varying ar coefficient modeling of nonstationary covariance time series. *IEEE Transactions on Automatic Control*, 30(1):48–56, 1985.

[121] M. West, R. Prado, and A. D. Krystal. Evaluation and Comparison of EEG Traces: Latent Structure in Nonstationary Time Series. *Journal of the American Statistical Association*, 94(448):1083–1095, 1999.

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

[123] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[124] K. Mizuseki, A. Sirota, E. Pastalkova, and G. Buzsaki. Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop. *Neuron*, 64:267–280, 2009.

[125] R. Rubinstein, A. M. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.

[126] Maneesh Sahani, John S. Pezaris, and Richard A. Andersen. On the separation of signals from neighboring cells in tetrode recordings. In *Advances in Neural Information Processing Systems 10*. 1998.

[127] S. R. Cole, R. van der Meij, E. J. Peterson, C. de Hemptinne, P. A. Starr, and B. Voytek. Nonsinusoidal Beta Oscillations Reflect Cortical Pathophysiology in Parkinson's Disease. *J. Neurosci.*, 37(18):4830–4840, 2017.

[128] Eric Betzig, George H. Patterson, Rachid Sougrat, O. Wolf Lindwasser, Scott Olenych, Juan S. Bonifacino, Michael W. Davidson, Jennifer Lippincott-Schwartz, and Harald F. Hess. Imaging intracellular fluorescent proteins at nanometer resolution. *Science*, 313(5793):1642–1645, 2006.

[129] C. Garcia-Cardona and B. Wohlberg. Convolutional dictionary learning: A comparative review and new algorithms. *IEEE Transactions on Computational Imaging*, 4(3):366–381, 2018.

[130] K. C. McGill and L. J. Dorfman. High-resolution alignment of sampled waveforms. *IEEE Transactions on Biomedical Engineering*, BME-31(6):462–468, 1984.

[131] C. Ekanadham, D. Tranchina, and E. P. Simoncelli. Recovery of sparse translation-invariant signals with continuous basis pursuit. *IEEE Transactions on Signal Processing*, 59(10):4735–4744, 2011.

[132] Kristian Bredies and Hanna Katriina Pikkarainen. Inverse problems in spaces of measures. *ESAIM: Control, Optimisation and Calculus of Variations*, 19(1):190–218, 2013.

[133] Nicholas. Boyd, Geoffrey. Schiebinger, and Benjamin. Recht. The alternating descent conditional gradient method for sparse inverse problems. *SIAM Journal on Optimization*, 27(2):616–639, 2017.

[134] Quentin Denoyelle, Vincent Duval, Gabriel Peyré, and Emmanuel Soubies. The sliding frank–wolfe algorithm and its application to super-resolution microscopy. *Inverse Problems*, 36(1), 2019.

[135] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, Dec 2007.

[136] Stefan Kunis and Holger Rauhut. Random sampling of sparse trigonometric polynomials, II. orthogonal matching pursuit versus basis pursuit. *Foundations of Computational Mathematics*, 8(6):737–763, 2008.

[137] Boris Mailhé, Rémi Gribonval, Pierre Vandergheynst, and Frédéric Bimbot. Fast orthogonal sparse approximation algorithms over local dictionaries. *Signal Processing*, 91:2822–2835, 2011.

[138] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[139] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic Decomposition by Basis Pursuit. *SIAM REVIEW c Society for Industrial and Applied Mathematics*, 43(1):129–159, 2001.

[140] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[141] S. G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[142] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constructive Approximation*, 13(1):57–98, 1997.

[143] Sacha Krstulovic and Rémi Gribonval. MPTK: Matching Pursuit made tractable. In *Proc. Int. Conf. Acoust. Speech Signal Process.*, 2006.

[144] Arthur Szlam, Koray Kavukcuoglu, and Yann LeCun. Convolutional matching pursuit and dictionary training. *CoRR*, abs/1010.0422, 2010.

[145] Karin C Knudson, Jacob Yates, Alexander Huk, and Jonathan W Pillow. Inferring sparse representations of continuous signals with continuous orthogonal matching pursuit. *Advances in Neural Information Processing Systems 27*, 27:1215–1223, 2014.

[146] G. Tang, B. N. Bhaskar, and B. Recht. Sparse recovery over continuous dictionaries-just discretize. In *2013 Asilomar Conference on Signals, Systems and Computers*, pages 1043–1047, 2013.

[147] Vincent Duval and Gabriel Peyré. Sparse regularization on thin grids I: the lasso. *Inverse Problems*, 33(5), 2017.

[148] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. *Cs Technion*, 40(8):1–15, 2008.

[149] Micha Aharon. *Overcomplete Dictionaries for Sparse Representation Signals*. PhD thesis, Technion - Israel Institute of Technology, 2006.

[150] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1981.

[151] A. Fannjiang and W. Liao. Coherence pattern–guided compressive sensing with unresolved grids. *SIAM Journal on Imaging Sciences*, 5(1):179–202, 2012.

[152] Alekh Agarwal, Animashree Anandkumar, Prateek Jain, and Praneeth Netrapalli. Learning Sparsely Used Overcomplete Dictionaries via Alternating Minimization. *SIAM Journal on Optimization*, 26(4):2775–2799, 2016.

[153] D a Henze, Z Borhegyi, J Csicsvari, A Mamiya, K D Harris, and G Buzsáki. Intracellular features predicted by extracellular recordings in the hippocampus in vivo. *Journal of neurophysiology*, 84(1):390–400, 2000.

[154] R. Quian Quiroga, Z. Nadasdy, and Y. Ben-Shaul. Unsupervised Spike Detection and Sorting with Wavelets and Superparamagnetic Clustering. *Neural Computation*, 16(8):1661–1687, 2004.

[155] Lei Zhu, Wei Zhang, Daniel Elnatan, and Bo Huang. Faster STORM using compressed sensing. *Nature Methods*, 9:721–723, 2012.

[156] Raja Giryes and Michael Elad. Sparsity-based poisson denoising with dictionary learning. *IEEE Transactions on Image Processing*, 23(12):5057–5069, 2014.

[157] Brendt Wohlberg and Przemek Wozniak. PSF estimation in crowded astronomical imagery as a convolutional dictionary learning problem. *IEEE Signal Processing Letters*, 28:374–378, 2021.

[158] Nitin Sadras, Bijan Pesaran, and Maryam M Shanechi. A point-process matched filter for event detection and decoding from population spike trains. *Journal of Neural Engineering*, 16(6), 2019.

[159] L. Huo, X. Feng, C. Pan, S. Xiang, and C. Huo. Learning smooth dictionary for image denoising. In *Ninth International Conference on Natural Computation (ICNC)*, pages 1388–1392, 2013.

[160] Elvis Dohmatob, Arthur Mensch, Gael Varoquaux, and Bertrand Thirion. Learning brain regions via large-scale online structured sparse dictionary learning. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[161] Luxin Yan, Hai Liu, Sheng Zhong, and Houzhang Fang. Semi-blind spectral deconvolution with adaptive tikhonov regularization. *Applied Spectroscopy*, 66(11):1334–1346, 2012.

[162] Yong Sheng Soh. Group invariant dictionary learning. *IEEE Transactions on Signal Processing*, 69:3612–3626, 2021.

[163] L Fahrmeir and G Tutz. *Multivariate Statistical Modelling Based on Generalized Linear Models*. 2001.

[164] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[165] B. Matern. *Spatial Variation*. Springer-Verlag, 1960.

[166] Salomon Bochner. *Lecture on Fourier Integrals*. Princeton University Press, 1959.

[167] C. Garcia-Cardona and B. Wohlberg. Convolutional dictionary learning: A comparative review and new algorithms. *IEEE Transactions on Computational Imaging*, 4(3):366–381, 2018.

[168] Ac Lozano, Grzegorz Swirszcz, and Naoki Abe. Group orthogonal matching pursuit for logistic regression. *Journal of Machine Learning Research*, 15:452–460, 2011.

[169] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. *An Introduction to Total Variation for Image Analysis*. De Gruyter, 2010.

[170] Alekh Agarwal, Anima Anandkumar, Prateek Jain, Praneeth Netrapalli, and Rashish Tandon. Learning sparsely used overcomplete dictionaries via alternating minimization. *SIAM Journal on Optimization*, 26:2775–2799, 2016.

[171] Dianne P. O'Leary. Near-optimal parameters for tikhonov and other regularization methods. *SIAM Journal on Scientific Computing*, 23(4):1161–1171, 2001.

[172] Simona Temereanca, Emery N. Brown, and Daniel J. Simons. Rapid changes in thalamic firing synchrony during repetitive whisker stimulation. *Journal of Neuroscience*, 28(44):11,153–11,164, 2008.

[173] Cristina Garcia-Cardona and Brendt Wohlberg. Convolutional dictionary learning: A comparative review and new algorithms. *IEEE Transactions on Computational Imaging*, 4(3):366–381, September 2018.

[174] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[175] Bahareh Tolooshams, Sourav Dey, and Demba Ba. Scalable convolutional dictionary learning with constrained recurrent sparse auto-encoders. In *Proc. 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2018.

[176] Hillel Sreter and Raja Giryes. Learned convolutional sparse coding. In *Proc. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2191–2195, 2018.

[177] Jeremias Sulam, Aviad Aberdam, Amir Beck, and Michael Elad. On multi-layer basis pursuit, efficient algorithms and convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[178] John R. Hershey, Jonathan Le Roux, and Felix Weninger. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv:1409.2574*, pages 1–27, 2014.

[179] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *arXiv:1912.10557*, 2019.

[180] Feng Yang, Yue M Lu, Luciano Sbaiz, and Martin Vetterli. Bits from photons: Oversampled image acquisition using binary poisson statistics. *IEEE Transactions on image processing*, 21(4):1421–1436, 2011.

[181] Joseph Salmon, Zachary Harmany, Charles-Alban Deledalle, and Rebecca Willett. Poisson noise reduction with non-local pca. *Journal of Mathematical Imaging and Vision*, 48(2):279–294, Feb 2014.

[182] Liyan Ma, Lionel Moisan, Jian Yu, and Tieyong Zeng. A dictionary learning approach for poisson image deblurring. *IEEE Transactions on medical imaging*, 32(7):1277–1289, 2013.

[183] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, July 2017.

[184] Tal Remez, Or Litany, Raja Giryes, and Alexander M. Bronstein. Class-aware fully-convolutional gaussian and poisson denoising. *CoRR*, abs/1808.06562, 2018.

[185] W. Feng, P. Qiao, and Y. Chen. Fast and accurate poisson denoising with trainable nonlinear diffusion. *IEEE Transactions on Cybernetics*, 48(6):1708–1719, 2018.

[186] Alfredo Nazábal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using VAEs. *CoRR*, abs/1807.03653, 2018.

[187] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, pages 689–698, 2018.

[188] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014.

[189] Zhaowen Wang, Ding Liu, Jianchao Yang, Wei Han, and Thomas Huang. Deep networks for image super-resolution with sparse prior. In *Proc. the IEEE International Conference on Computer Vision*, pages 370–378, 2015.

[190] Bahareh Tolooshams, Sourav Dey, and Demba Ba. Deep residual auto-encoders for expectation maximization-inspired dictionary learning. *IEEE Transactions on neural networks and learning systems*, 2020.

[191] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[192] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations*, 2016.

[193] Dror Simon and Michael Elad. Rethinking the csc model for natural images. In *Proc. Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 2271–2281, 2019.

[194] Abiy Tasissa, Emmanouil Theodosis, Bahareh Tolooshams, and Demba Ba. Dense and sparse coding: Theory and architectures. *arXiv:2006.09534*, 2020.

[195] Morteza Mardani, Qingyun Sun, Shreyas Vasawanala, Vardan Papyan, Hatef Monajemi, John Pauly, and David Donoho. Neural proximal gradient descent for compressive imaging. In *Proc. Advances in Neural Information Processing Systems 31*, pages 9573–9683, 2018.

[196] Yuanjun Gao, Evan W Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. In *Advances in Neural Information Processing Systems 29*, pages 163–171. Curran Associates, Inc., 2016.

[197] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, 2012.

[198] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[199] M. Makitalo and A. Foi. Optimal inversion of the generalized anscombe transformation for poisson-gaussian noise. *IEEE Transactions on Image Processing*, 22(1):91–103, Jan 2013.

[200] Honglak Lee, Rajat Raina, Alex Teichman, and Andrew Y. Ng. Exponential family sparse coding with applications to self-taught learning. In *Proc. the 21st International Jont Conference on Artificial Intelligence*, IJCAI, pages 1113–1119, 2009.

[201] Demba Ba, Simona Temereanca, and Emery Brown. Algorithms for the analysis of ensemble neural spiking activity using simultaneous-event multivariate point-process models. *Frontiers in Computational Neuroscience*, 8:6, 2014.

[202] Emery N. Brown, Riccardo Barbieri, Valérie Ventura, Robert E. Kass, and Loren M. Frank. The time-rescaling theorem and its application to neural spike train data analysis. *Neural Computation*, 14(2):325–346, 2002.