Sparse deconvolution with applications to spike sorting

Thesis by
Kevin Qing Shan

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2019
Defended March 4, 2019

Kevin Qing Shan
ORCID: 0000-0002-2621-1274

ACKNOWLEDGEMENTS

First, I would like to thank Thanos Siapas for his support and mentorship over all these years, and the rest of my committee—Joel Burdick, Richard Murray, and Michael Dickinson—for everything they've done to make this thesis happen.

Thanks also to Eugene Lubenov for invaluable discussions and feedback (including on this document), to Andreas Hoenselaar for being a positive role model for software development practices, to Britton Sauerbrei for his dedication to data visualization, and to Brad Hulse and Maria Papadopoulou for their conversations and commiserations, both intellectual and otherwise.

I'd also like to thank my CDS cohort—Seungil, Ivan, Anandh—and other Caltech friends—Andrew, Hannah, Denise, Max, Ioana—for being part of my life outside the lab. Finally, I am immensely grateful to my parents and my brother Kyle for their constant support and encouragment, and to my wonderful wife Sze for all of the above and more.

ABSTRACT

Chronic extracellular recording is the use of implanted electrodes to measure the electrical activity of nearby neurons over a long period of time. It presents an unparalleled view of neural activity over a broad range of time scales, offering sub-millisecond resolution of single action potentials while also allowing for continuous recording over the course of many months. These recordings pick up a rich collection of neural phenomena—spikes, ripples, and theta oscillations, to name a few—that can elucidate the activity of individual neurons and local circuits.

However, this also presents an interesting challenge for data analysis. Chronic extracellular recordings contain overlapping signals from multiple sources, requiring these signals to be detected and classified before they can be properly analyzed. The combination of fine temporal resolution with long recording durations produces large datasets, requiring efficient algorithms that can operate at scale.

In this thesis, I consider the problem of spike sorting: detecting spikes (the extracellular signatures of individual neurons' action potentials) and clustering them according to their putative source. First, I introduce a sparse deconvolution approach to spike detection, which seeks to detect spikes and represent them as the linear combination of basis waveforms. This approach is able to separate overlapping spikes without the need for source templates, and produces an output that can be used with a variety of clustering algorithms.

Second, I introduce a clustering algorithm based around a mixture of drifting t-distributions. This model captures two features of chronic extracellular recordings—cluster drift over time and heavy-tailed residuals in the distribution of spikes—that are missing from previous models. This enables us to reliably track individual neurons over longer periods of time. I will also show that this model produces more accurate estimates of classification error, which is an important component to proper interpretation of the spike sorting output.

Finally, I present a few theoretical results that may assist in the efficient implementation of sparse deconvolution.

PUBLISHED CONTENT AND CONTRIBUTIONS

Shan, K. Q., E. V. Lubenov, and A. G. Siapas (2017). "Model-based spike sorting with a mixture of drifting t-distributions". *Journal of neuroscience methods* 288, pp. 82–98. DOI: 10.1016/j.jneumeth.2017.06.017.
KQS conceived of the project, developed and analyzed the method, collected some of the validation data, and wrote the manuscript.

Shan, K. Q., E. V. Lubenov, M. Papadopoulou, and A. G. Siapas (2016). "Spatial tuning and brain state account for dorsal hippocampal CA1 activity in a non-spatial learning task". *Elife* 5, e14321. DOI: 10.7554/eLife.14321.001.
KQS performed some of the experiments, analyzed the data, and wrote the manuscript.

TABLE OF CONTENTS

Acknowledgements . . . iii
Abstract . . . iv
Published Content and Contributions . . . v
Table of Contents . . . vi
List of Illustrations . . . vii
Chapter I: Introduction . . . 1
Chapter II: Spike detection via sparse deconvolution . . . 5
2.1 Introduction . . . 5
2.2 Methods . . . 8
2.3 Results . . . 17
2.4 Discussion . . . 19
Chapter III: Spike sorting using a mixture of drifting t-distributions . . . 20
3.1 Introduction . . . 20
3.2 Mixture of drifting t-distributions (MoDT) model . . . 23
3.3 Model validation using empirical data . . . 32
3.4 Use of the MoDT model for measuring unit isolation . . . 39
3.5 Overlapping spikes . . . 46
Chapter IV: Improving regularizers for sparse deconvolution . . . 49
4.1 The nonconvex log regularizer . . . 50
4.2 Nested group regularizers . . . 59
Chapter V: Conclusion . . . 72
Bibliography . . . 75

LIST OF ILLUSTRATIONS

Number		*Page*
1.1	An overview of spike sorting	2
2.1	Spike detection as a sparse deconvolution problem	6
2.2	Issues with the convex $l_{2,1}$ regularizer	13
2.3	Comparison of optimization algorithms	15
2.4	Spike detection performance on a hybrid ground truth dataset	17
2.5	Spike removal for LFP analysis	18
3.1	Extracellular recordings contain drifting, heavy-tailed clusters	21
3.2	Scaling of computational runtime with MoDT model dimensions	28
3.3	Robust covariance estimation using the t-distribution	30
3.4	Cluster drift in well-isolated units	33
3.5	Heavy-tailed residuals in extracellular noise and well-isolated units	34
3.6	Failure modes of a stationary approach	36
3.7	Spike sorting performance	43
3.8	Comparison of unit quality metrics on hybrid ground truth datasets	43
3.9	Spike sorting performance on overlapping spikes	46
3.10	Model-based overlap reassignment	47
4.1	Application of the nonconvex log regularizer to a sparse deconvolution problem	50
4.2	An illustration of the proximal operator of the log regularizer	53
4.3	Comparison of the proximal operator to prior work	58
4.4	Group lasso regularizer applied to the spike detection deconvolution problem	60

Chapter 1

INTRODUCTION

Chronic extracellular recordings are a powerful tool for systems neuroscience, offering access to the spiking activity of neurons over long periods over time. In this technique, recording electrodes are chronically implanted in the brain to monitor the activity of nearby neurons. Unlike imaging techniques, these passive recordings do not damage the tissue through photobleaching or require expression of foreign fluorophores, enabling continuous recordings over the course of many weeks or even months. The ability to continuously monitor individual neurons over these time scales is critical to advancing our understanding of learning and memory.

However, the analysis of extracellular data requires a process known as spike sorting, in which spikes—the extracellular signatures of individual neurons' action potentials—are detected in the raw data and then assigned to putative sources (i.e., individual neurons, also known as single units). This process is summarized in Figure 1.1 and is typically broken into two stages: (1) *spike detection*, in which spikes are detected as discrete events in a noisy signal, and (2) *clustering*, an unsupervised learning problem in which the detected spikes are assigned to putative sources.

Unfortunately, prior work on this topic have two important shortcomings that prevent us from achieving our goal of continuous monitoring of spiking activity over long time scales. In this document, I develop new methods to overcome these shortcomings.

First, traditional spike detection methods do not account for overlapping spikes, i.e., spikes from different neurons that occur nearly synchronously and thus overlap in time. This is a rare occurrence, since the duration of individual spikes is short relative to the average time between spikes, so traditional techniques can achieve a reasonable overall performance despite failing to account for this overlap. However, the brain sometimes exhibits patterns of coordinated activity that increase the prevalence of overlapping spikes, and these patterns are of special interest to researchers. For example, during a hippocampal activity pattern known as a *ripple*, a sizeable fraction of pyramidal neurons all fire in close succession. These population bursts are a prominent feature of slow-wave sleep (Siapas and Wilson, 1998), appear to be involved in information transfer to other brain areas (Wierzynski et al., 2009), and sometimes exhibit surprising sequences of neural firing (Foster and Wilson, 2006).

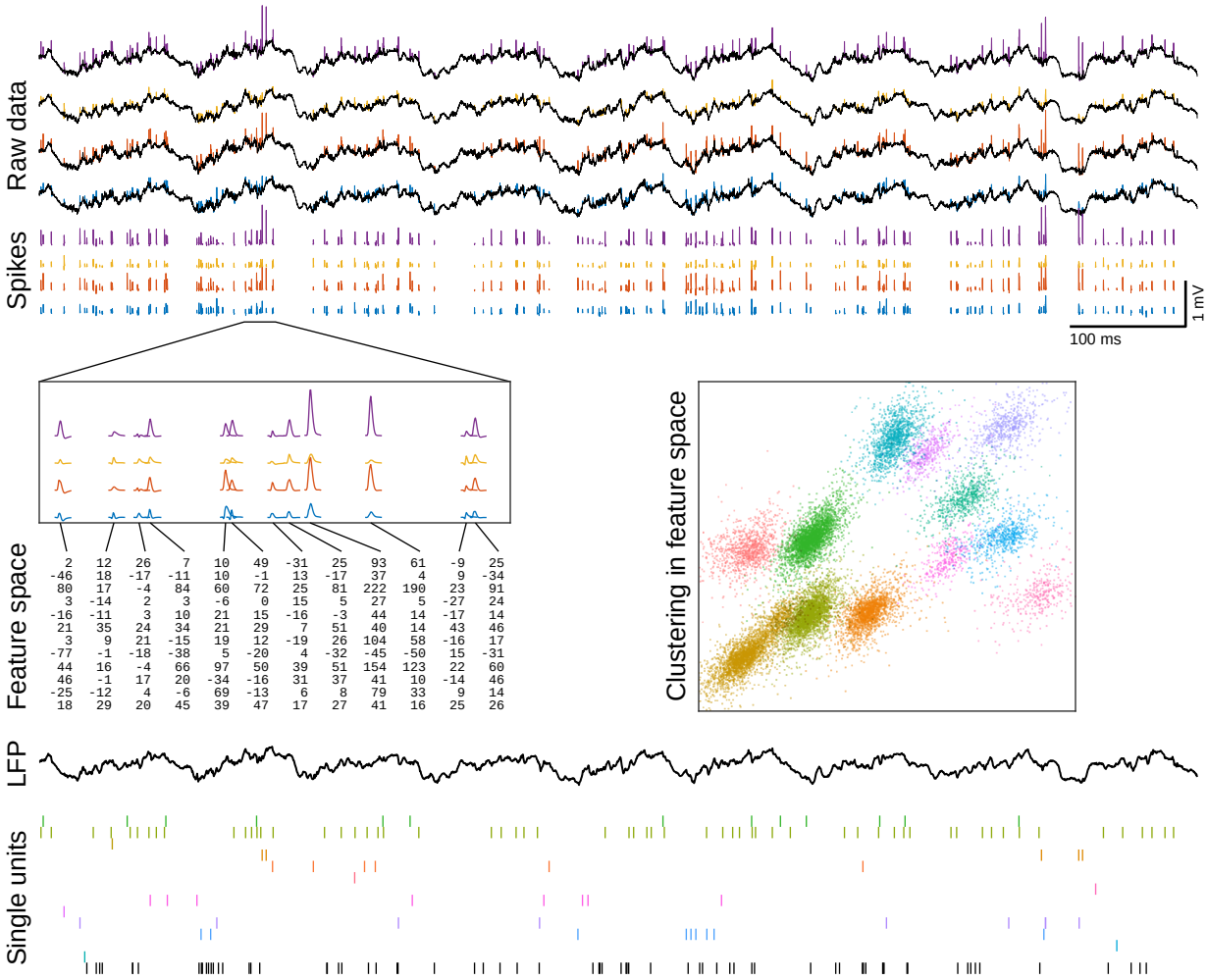

Figure 1.1: An overview of spike sorting. Spike sorting is the process of detecting spikes (the extracellular signatures of neuronal action potentials) and assigning them to putative sources. **Raw data:** Voltage traces from the four channels of a tetrode (a four-site recording probe constructed from bundled microwires) in hippocampal area CA1 of a freely-behaving rat. **Spikes:** During spike detection, we detect spikes in the raw data and extract their waveforms for downstream analysis. The shape of the spike waveforms (particularly the relative amplitude across the different channels) can be used to classify individual spikes as originating from different sources. **Feature space:** This information about each spike's waveform is represented as a point in some low-dimensional feature space (the 12-dimensional column vectors in this illustration), a format that is amenable to analysis with a variety of unsupervised clustering algorithms. **Clustering in feature space:** This step groups the spikes into clusters and produces an estimate of the misclassification error. **Single units:** Well-isolated clusters that satisfy the appropriate criteria are known as single units and may be interpreted as the spiking activity of individual neurons (shown as a raster plot here). The last row (black) shows the detected spikes that were not classified as belonging to a single unit. **LFP:** After spikes have been detected and removed from the raw data, the residual signal may be analyzed as the local field potential (LFP), which offers valuable insight into aggregate network activity.

Improving spike detection during these synchronous bursts is thus an important step to advancing our understanding of these phenomena.

To this end, Chapter 2 describes a sparse deconvolution approach to spike detection. This procedure approximates the observed signal as the convolution of a set of kernels (representing the linear subspace of typical spike waveforms) with a column-sparse feature matrix (which are the detected spikes). This approach explicitly accounts for overlapping spikes and is able to reliably separate the contribution from each spike. Unlike the class of techniques known as *template matching*, this approach does not require fitting a source template for each neuron and does not seek to assign the detected spikes to putative sources. It instead represents the detected spikes as points in a low-dimensional feature space, a format that is amenable to downstream analysis using a wide variety of clustering algorithms. This approach thereby maintains the traditional decoupling of spike detection from the clustering problem.

The second shortcoming of prior spike-sorting methods lies in the clustering of long datasets. Traditional clustering algorithms have been fairly successful when applied to recordings up to an hour, which is the duration of a typical behavioral training session. However, many behavioral tasks require multiple days to learn (Shan et al., 2016), and the process of memory consolidation—the transfer of long-term memories from the hippocampus to cortical brain areas—may require weeks (Takehara, Kawahara, and Kirino, 2003). The ability to observe changes in neural activity over these time scales is therefore critical to understanding the neural mechanisms underlying learning and memory. Although there are few experimental barriers to acquiring such long-term, continuous recordings, the analysis of such datasets presents a major challenge to traditional clustering algorithms. This is because the clusters corresponding to individual neurons drift substantially on the time scale of hours (due to a combination of electrode motion and physiological changes in the neurons), which violates the traditional assumption of cluster stationarity.

To address this issue, Chapter 3 describes a clustering algorithm based around fitting the data with a mixture of drifting t-distributions. This generative model captures two important features of chronic extracellular recordings—cluster drift over time and heavy tails in the distribution of spikes—that are missing from previous models. This model also provides accurate estimates of classification error, an important metric for proper interpretation of the spike sorting output.

Finally, Chapter 4 derives some mathematical results (proximal operators for the class of regularizers used in Chapter 2) that may assist in the efficient implementation

of sparse deconvolution.

Chapter 2

SPIKE DETECTION VIA SPARSE DECONVOLUTION

2.1 Introduction

The first step in spike sorting is detecting the spikes to be sorted. The earliest methods for spike detection simply involved bandpass filtering the signal and detecting threshold-crossings. Refinements to this technique include the use of data transformations such as the nonlinear energy operator (Mukhopadhyay and Ray, 1998) or a wavelet transform (Yang and Shamma, 1988; Nenadic and Burdick, 2005) prior to thresholding. These transformations seek to improve detection performance by accentuating certain features of neural spikes that distinguish them from the background noise. After spikes are detected, their waveforms—a short segment of data centered around each detected spike—may be extracted and then projected into a low-dimensional feature space for subsequent clustering.

While this approach works well for isolated spikes, it can fail to detect spikes that occur in close succession (less than 1 ms apart), since the overlapping waveforms may fail to trigger the detection. Even if both spikes are detected, the presence of the other spike may distort the observed waveform, leading to downstream misclassification error if the contribution of the other spike is not accounted for[1].

The issue of overlapping spikes is greatest in areas with a high density of cells, such as the hippocampal pyramidal cell layer, since this leads to a large number of neurons within the electrode's recording volume. This is further compounded by synchronous activity patterns, such as ripples, that may cause a large number of neurons to fire within a short window of time.

An alternative class of spike detection methods, known as template matching, addresses this issue by seeking to approximate the observed data as the sum of template waveforms (Bankman, Johnson, and Schneider, 1993; Pachitariu et al., 2016; Yger et al., 2018), which may be understood as a form of sparse deconvolution (Figure 2.1A). Each template corresponds to a different source, so detecting a spike and assigning it to a putative neuron occur as a single step. Although finding the

[1] I had previously developed a method to do precisely this (Shan, Lubenov, and Siapas, 2017, section 2.6). In cases where both spikes are detected, a limited amount of deconvolution may be performed during the process of dimensionality reduction (also known as feature extraction). However, this approach still struggles with spikes that are less than 0.5 ms apart, due to failure of spike detection.

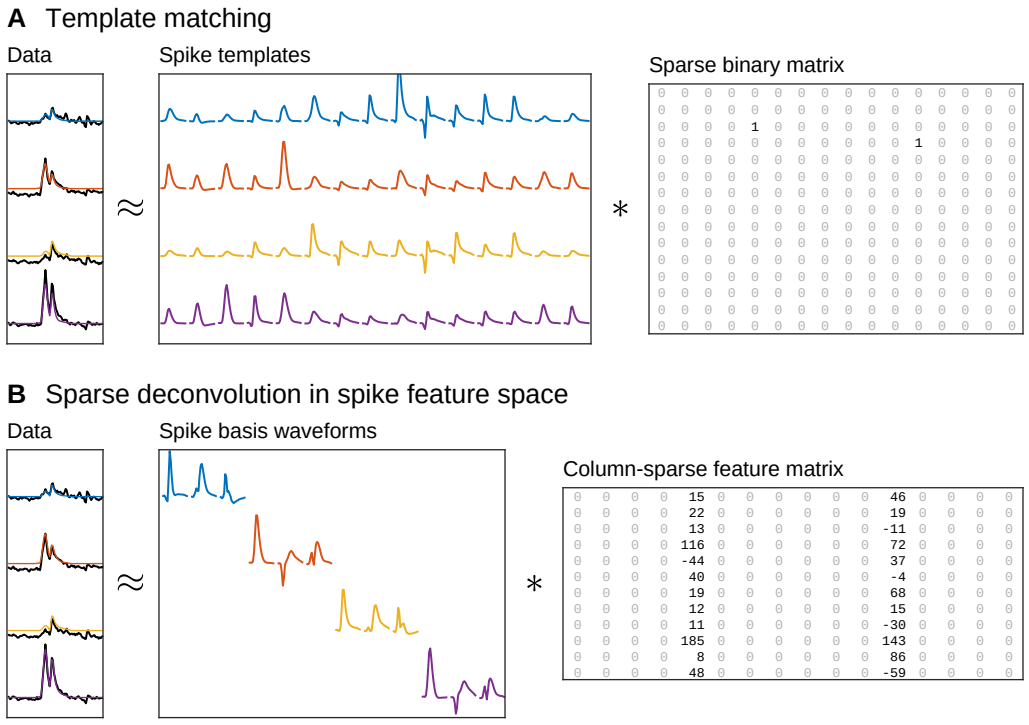

Figure 2.1: Spike detection as a sparse deconvolution problem. (A) Template matching approximates the observed data (left, black traces) as the convolution of a set of spike templates (one per source neuron) with a sparse binary matrix x, where $x[i, j]$ represents the presence of a spike from source i at time j. The product of this convolution is shown as colored traces in the left panel. Since this procedure simultaneously detects spikes and classifies them according to their putative source, it is very sensitive to the spike templates used, which must be fitted to the dataset at hand. **(B)** We will instead approximate the data as the convolution of spike basis waveforms (which define the linear subspace of neural spike waveforms) with a column-sparse feature matrix. The nonzero columns of this feature matrix can then be used for subsequent clustering in feature space, thereby decoupling the problems of spike detection and classification. In this example, each basis waveform is restricted to a single data channel; this allows for a more efficient convolution and easier interpretation of the feature coordinates.

optimal solution to this problem is NP-hard, a combination of a greedy approach and limited search seems to produce adequate results. These methods also offer improved detection sensitivity, particularly when combined with pre-whitening, since they provide a more explicit description of how neural spikes may be distinguished from the background noise.

However, template matching has some important shortcomings. First, in order to allow for an efficient implementation, the classifier boundaries between clusters are typically constrained to simple forms such as hyperplanes, which may not be appropriate when the data are highly heteroscedastic (which is often the case when

multiple neuronal cell types are present). Second, unless the spike templates are somehow known beforehand, they need to be learned from the data through some form of clustering. But clustering is a difficult problem. We may wish to run it from multiple initializations or compare across different model classes, and each of these runs may require an iterative fitting process. Since any change to the templates requires re-detecting spikes, each iteration thus involves another pass through the raw data.

In contrast, traditional spike detection only needs to be performed once and transforms 30 GB of raw data (a day's worth of recordings from a single tetrode) into 0.3 GB of spikes in feature space. This not only reduces the data size by 100x, but allows it to fit into GPU memory, which has access speeds 1000x faster than disk. This dramatically improves fitting times and enables the use of more sophisticated classifiers for spike sorting. Furthermore, this modularity—the fact that the spike detection process is agnostic to the downstream clustering step—is immensely useful from a practical standpoint, as it allows for rapid evaluation of novel clustering algorithms and allows the raw data to be relegated to offline storage.

In this chapter, I will describe a new method of spike detection that combines the modularity of traditional spike detection with the improved overlap resolution of template matching. Like template matching, this approaches spike detection as a sparse deconvolution problem (Figure 2.1B). But instead of convolving a set of source-specific spike templates with a sparse binary matrix assigning spikes to putative sources, which intertwines the problems of spike detection and clustering, we will be convolving a set of spike basis waveforms with a column-sparse feature matrix, which may then be used in the clustering algorithm of your choice.

Section 2.2 describes how to set up and solve this sparse deconvolution problem, section 2.3 evaluates its performance as a spike detection algorithm, and section 2.4 closes this chapter with some additional discussion.

Table 2.1: Mathematical notation. For a $[D \times T]$ matrix x, the notation $x[d, :]$ refers to its d-th row and $x[:, t]$ refers to its t-th column.

Dimensions	
D	Number of feature space dimensions
C	Number of data channels
T	Number of time samples
L	Kernel length
Variables	
b	$\mathbb{R}^{C \times T}$ raw data (voltage traces from C recording channels)
x	$\mathbb{R}^{D \times T}$ optimization decision variable. The nonzero columns of this matrix form the feature space representation of the detected spikes.
k_d	$\mathbb{R}^{C \times L}$ convolution kernel (spike basis waveform) for feature coordinate d
β	Nonnegative parameter that controls the relative importance of minimizing the regularizer vs. the approximation error.
Functions	
$\mathcal{A}$	$\mathbb{R}^{D \times T} \mapsto \mathbb{R}^{C \times T}$ linear operator that convolves each row $x[d, :]$ by the corresponding kernel k_d and sums them together
$\mathcal{A}^\dagger$	Adjoint of $\mathcal{A}$ with respect to the standard inner product in $\mathbb{R}^{D \times T}$ and the whitened inner product in $\mathbb{R}^{C \times T}$
$\langle \cdot, \cdot \rangle_w$	Whitened inner product between two $\mathbb{R}^{C \times T}$ data matrices
$\|\cdot\|_w$	Norm induced by the inner product $\langle \cdot, \cdot \rangle_w$
$f(\cdot)$	$\mathbb{R}^{D \times T} \mapsto \mathbb{R}$ approximation error function, equivalent to $\frac{1}{2}\|\mathcal{A}x - b\|_w^2$
$g(\cdot)$	$\mathbb{R}^{D \times T} \mapsto \mathbb{R}$ regularizer function that encourages the optimization to produce column-sparse x

2.2 Methods

In this section, we will pose spike detection as the regularized linear least-squares problem

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{A}x - b\|_w^2 + \beta g(x), \tag{2.1}$$

where we seek to approximate the raw data (b) as the convolution of spike basis waveforms (operator $\mathcal{A}$) with sparse coefficients (x). This notation is summarized in Table 2.1 and will be examined in more detail in the following subsections. First, section 2.2.1 explains the whitened error norm $\|\mathcal{A}x - b\|_w$. Next, section 2.2.2 explains the convolution operator $\mathcal{A}$ and section 2.2.3 discusses the kernels that comprise it. Finally, section 2.2.4 explains the sparsity-inducing regularizer $g(x)$, and section 2.2.5 provides a comparison of optimization algorithms.

2.2.1 Whitened inner product in data space

The neural background activity in electrophysiological recordings, which we may consider noise in our spike detection problem, does not have equal power at all frequencies. Instead, we observe much less noise power at higher frequencies: the rolloff is on the order of $1/f$, although it varies from channel to channel and may be complicated by the presence of line noise and/or high-frequency neuronal oscillations.

In order to account for this when measuring the approximation error $\|\mathcal{A}x - b\|$, we will use a whitened inner product in data space, defined as

$$\langle a, b \rangle_w = \langle \mathcal{W}a, \mathcal{W}b \rangle, \tag{2.2}$$

where the right hand side uses the standard inner product and the linear operator $\mathcal{W} : \mathbb{R}^{C \times T} \mapsto \mathbb{R}^{C \times T}$ is known as the whitening filter. Ultimately, this is equivalent to pre-whitening the data b and defining the convolution kernels k_d in a whitened data space, but treating the whitening as a separate operation allows us to discuss its role separately and leads to a more efficient implementation when $D > C$.

This whitening can take many different forms. Temporal whitening using autoregressive models is a very popular approach, but can lead to troublesome boundary issues. For this reason, I favor a symmetric finite impulse response (FIR) whitening filter designed using least squares to approximate a desired frequency response. For spike detection, the exact response at low frequencies is typically irrelevant as long as it is sufficiently attenuated, allowing us to get away with relatively short filters.

The whitening operator may also perform whitening across channels (also known as spatial whitening) to make the spike detection less sensitive to common-mode noise. This is typically applied to each time sample independently, after temporal whitening.

For the experiments in this chapter, I used the empirical power spectral density to design whitening filters for each channel. I also added a bandpass to the desired frequency response (with -6 dB cutoff frequences at 400 Hz and 8 kHz) to further discount the influence of other (non-spike) signals at lower frequencies and to avoid excessive amplification of high-frequency noise. I used the matrix square root of the channel covariance matrix as the spatial whitener.

2.2.2 Convolution

The linear operator $\mathcal{A}$ maps a given $[D \times T]$ feature matrix to a $[C \times T]$ convolution output in data space. It is responsible for convolving each feature coordinate (each row of x) with its corresponding kernel and summing them all together.

Although it is conceptually convenient to think of $\mathcal{A}$ as a $[CT \times DT]$ block Toeplitz matrix, it is impractical to implement it as such, given the data dimensions that we are considering ($T \approx 1$ million). This section will describe how to evaluate $\mathcal{A}$ and its adjoint $\mathcal{A}^\dagger$.

First, each channel of $y = \mathcal{A}x$ is given by the sum of one-dimensional convolutions,

$$y[c,:] = \sum_{d=1}^{D} k_d[c,:] * x[d,:], \tag{2.3}$$

which may be computed using the overlap-add technique to emulate the linear convolution as a sequence of circular convolutions that may then be diagonalized using fast Fourier transform (FFT) algorithms.

Our examples (e.g., Figure 2.1B) use kernels that have single-channel support: each kernel is restricted to a single channel only and $k_d[c,:] = 0$ for all other channels. This reduces the number of convolutions we need to perform and makes the coordinates of the feature space easier to interpret, although it is less efficient in terms of the number of feature space dimensions D required to achieve a desired approximation error.

Second, we will also need the adoint of $\mathcal{A}$, i.e., the linear operator $\mathcal{A}^\dagger$ such that

$$\langle \mathcal{A}x, y \rangle_w = \langle x, \mathcal{A}^\dagger y \rangle,$$

where the left hand side uses the whitened inner product discussed in section 2.2.1 and the right hand side uses the standard inner product in $\mathbb{R}^{D \times T}$. The adjoint $\mathcal{A}^\dagger$ arises in our optimization problem because the gradient of the squared error $f(x)$ is given by

$$\nabla_f(x) = \mathcal{A}^\dagger(\mathcal{A}x - b).$$

If we let $\tilde{y} = \mathcal{W}^\dagger \mathcal{W} y$, where $\mathcal{W}$ is the whitening operator described in section 2.2.1, then $x = \mathcal{A}^\dagger y$ is given by

$$x[d,:] = \sum_{c=1}^{C} k_d^\dagger[c,:] * \tilde{y}[c,:], \tag{2.4}$$

where $k_d^\dagger$ denotes the conjugate time-reversal of the kernel k_d.

2.2.3 Determining the spike basis waveforms

One of the major shortcomings of template matching is the difficulty in determining the appropriate spike templates to use. Yet our convolution operator assumes that we are given an appropriate set of spike basis waveforms. How is that any different?

The fundamental difference is that the spike basis waveforms do not need to represent the specific neurons present in the recording, but rather the general space of neural spike waveforms that may be recorded on this probe. This has two important consequences: (1) spike basis waveforms are stable over time, and (2) our choice of spike basis waveforms does not directly impact spike classification, as that is deferred to a later clustering step.

First, let us note that the spike waveforms recorded from a given neuron are not constant but will drift over time (on the order of minutes or hours, see Shan, Lubenov, and Siapas, 2017). The spike templates used in template matching therefore need to track these changes over time, requiring frequent updating. In contrast, the spike basis waveforms for a single channel depend primarily on the impedance of the electrode (which affects its transfer function as a recording device) and the overall distribution of cell types present (which affects the distribution of waveform shapes that may be observed). These properties change relatively slowly over time, on the order of days or weeks rather than minutes.

Second, unlike template matching, where the spike classifier boundaries are directly determined by the spike templates used, the spike classification performance under our approach is relatively insensitive to the choice of basis waveforms, as long as they are able to capture a reasonable fraction of the spike waveform variability. Furthermore, we can always err on the side of inclusivity and increase the dimension of the feature space. The consequence may be additional detection of non-neural events, but these are easily rejected during the clustering step.

For the experiments in this chapter, I initialized the spike basis waveforms by performing traditional spike detection (using the nonlinear energy operator as the detection heuristic) on a small batch of data (20 randomly-selected chunks of 3 seconds each) and performing a principal components analysis (PCA) of the extracted waveforms, retaining the three largest principal components on each channel to act as the spike basis waveforms. I then performed gradient descent iterations (again using batches of small, randomly-selected chunks) to fine-tune the basis waveforms.

2.2.4 Choice of regularizer

In the previous three subsections, we discussed the $f(x) = \frac{1}{2}\|\mathcal{A}x - b\|_w^2$ term of our objective function, which serves as a measure of our approximation error. However, we also want our optimization to produce solutions that are sparse, so that we may interpret the nonzero entries as detected spike events. It is the second term of our objective function, the regularizer $g(x)$, that is responsible for encouraging the optimization algorithm to produce solutions with our desired sparsity pattern.

A common choice of sparsity-inducing regularizer is the l_1 norm, also known as the lasso penalty (Tibshirani, 1996), and it corresponds to the sum of the absolute values:

$$g(x) = \sum_{t=1}^{T} \sum_{d=1}^{D} \big|x[d,t]\big|.$$

This regularizer has seen extensive use in compressive sensing (Donoho, Elad, and Temlyakov, 2006; Candes, 2008) and sparse deconvolution (Taylor, Banks, and McCoy, 1979; Chen, Donoho, and Saunders, 2001), including neuroscience applications such as deconvolution of calcium imaging data (Vogelstein et al., 2010; Pnevmatikakis et al., 2016).

However, we will need to make two modifications to this regularizer for our spike detection application. First, while the l_1 regularizer indeed produces sparse solutions, it does not care how these nonzero elements are arranged. For our application, we want the matrix x to be column-sparse, i.e., to have few columns containing nonzero entries. This can be achieved using the matrix $l_{2,1}$ norm (Udell et al., 2016), also known as the group lasso (Yuan and Lin, 2006), which is the sum of the 2-norms of the columns of x:

$$g(x) = \sum_{t=1}^{T} \big\|x[:,t]\big\|. \tag{2.5}$$

This regularizer produces solutions with few nonzero columns, but does not further incentivize sparsity within a column.

Second, even though the $l_{2,1}$ norm imposes the desired structure on the sparsity pattern, its solutions are still not sparse enough[2]. In particular, since the regularizer

[2]How does this fit in with the fairly strong sparse recovery guarantees of l_1-regularized minimization (a.k.a. basis pursuit)? An unfortunate consequence of $\mathcal{A}$ being a convolution operator is that its "dictionary items" correspond to time-shifted versions of the same waveforms and are therefore highly correlated. As a result, sparse recovery theorems that rely on $\mathcal{A}$ satisfying the *restricted isometry condition* (Needell and Vershynin, 2009) or having a low *mutual coherence* (Donoho, Elad, and Temlyakov, 2006) are unable to guarantee recovery of more than one nonzero element.

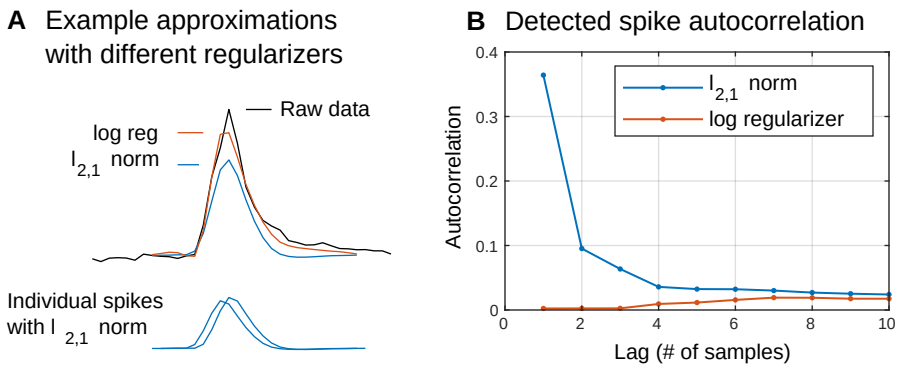

Figure 2.2: Issues with the convex $l_{2,1}$ regularizer. (A) The black trace shows a single spike from a single channel of data. Overlaid are the optimal approximations $\mathcal{A}x$ using the nonconvex log regularizer (red) and the convex $l_{2,1}$ norm (blue). Regularization with $l_{2,1}$ norm leads to two issues. First, the approximation is biased towards zero since the regularizer is always trying to shrink the feature vector norm. The log regularizer mitigates this by having a flatter slope for larger norms; note how the red trace captures more of the spike energy than the blue trace. Second, the $l_{2,1}$ norm does nothing to discourage spikes from being split across consecutive time steps. In fact, this blue trace is actually the sum of two smaller spikes detected at consecutive time samples (bottom). **(B)** Autocorrelations of the detected spike times. More than a third of the spikes detected using the $l_{2,1}$ norm (blue) are followed by another spike immediately afterwards (i.e., at a lag of 1). Such cases make it hard to interpret the deconvolution solution as a set of detected spikes. However, switching to the log regularizer (red) eliminates this problem.

is simply the sum of the column norms, there is no penalty to splitting a spike across consecutive time steps. That is, from the regularizer's point of view, there is no difference between x_1 and x_2 below:

$$x_1 = \begin{bmatrix} \cdots & \xi & 0 & \cdots \end{bmatrix}$$
$$x_2 = \begin{bmatrix} \cdots & \lambda\xi & (1-\lambda)\xi & \cdots \end{bmatrix},$$

where $\xi \in \mathbb{R}^D$ and $0 \leq \lambda \leq 1$. Even though x_1 is the sparser solution, the regularizer has no incentive to choose it, and x_2 often results in lower approximation error since it allows for a sort of linear interpolation (Figure 2.2A).

To discourage spikes from getting split up like this, we need to define the regularizer so that x_2 is more expensive than x_2, i.e., $g(x_2) > g(x_1)$. Unfortunately, this necessarily implies that the regularizer is no longer convex. To see this, consider that $x_2 = \lambda x_1 + (1-\lambda)x_3$, where

$$x_3 = \begin{bmatrix} \cdots & 0 & \xi & \cdots \end{bmatrix}.$$

Since x_3 is simply a time-shifted version of x_1, time-invariance requires that $g(x_1) = g(x_3) = \lambda g(x_1) + (1-\lambda)g(x_3)$. The condition that $g(x_2) > g(x_1)$ thus implies

that

$$g(\lambda x_1 + (1-\lambda)x_3) > \lambda g(x_1) + (1-\lambda)g(x_3),$$

therefore g cannot be convex.

A variety of nonconvex sparsity-encouraging regularizers have been proposed in the literature. One such regularizer is the log penalty (Candes, Wakin, and Boyd, 2008), and applying this to the column norms gives us our final regularizer:

$$g(x) = \sum_{t=1}^{T} \alpha \log(\left\|x[:,t]\right\| + \alpha). \tag{2.6}$$

$\alpha > 0$ is a parameter that controls the non-convexity of this regularizer; as $\alpha \to \infty$, this approaches the $l_{2,1}$ norm[3]. Switching from the convex $l_{2,1}$ norm (2.5) to the nonconvex log regularizer (2.6) eliminates the incidence of double-detected spikes (Figure 2.2B).

However, our choice of regularizer is also constrained by computational considerations. Candes, Wakin, and Boyd (2008) implemented this log-regularized approximation using a reweighted l_1 minimization scheme. This approach—which involves wrapping an outer loop around a convex version of the optimization problem—is less than ideal. Instead, we will approach this nonconvex minimization directly by deriving a closed-form expression for the proximal operator of our columnwise log regularizer (2.6) in Chapter 4. The existence of a simple proximal operator enables us to solve our optimization problem (2.1) using a variety of large-scale optimization algorithms, which I will discuss in the next section.

Finally, let me close this section with a few remarks on nonconvex optimization. Since it is nonconvex, we are not guaranteed to converge to the global minimum, and may instead get caught in local minima. Based on numerical experiments, it seems that we can improve the quality of our solutions by (1) starting with the convex solution by setting $\alpha = \infty$, and (2) slowly ramping down α over the course of many iterations in a process loosely analogous to simulated annealing. I found that this approach produced sparser solutions with less approximation error than the reweighted l_1 minimization of Candes, Wakin, and Boyd (2008), and reached this solution in far fewer iterations.

[3] I will note that equation (2.6) differs slightly from previous work by the addition of factor of α in front of the log. This is not a substantive change, since it is equivalent to modifying β in (2.1), but it does lend the convenient property that the derivative $g'(0) = 1$ for all α. This produces a family of regularizers as seen in Figure 4.1, inset.

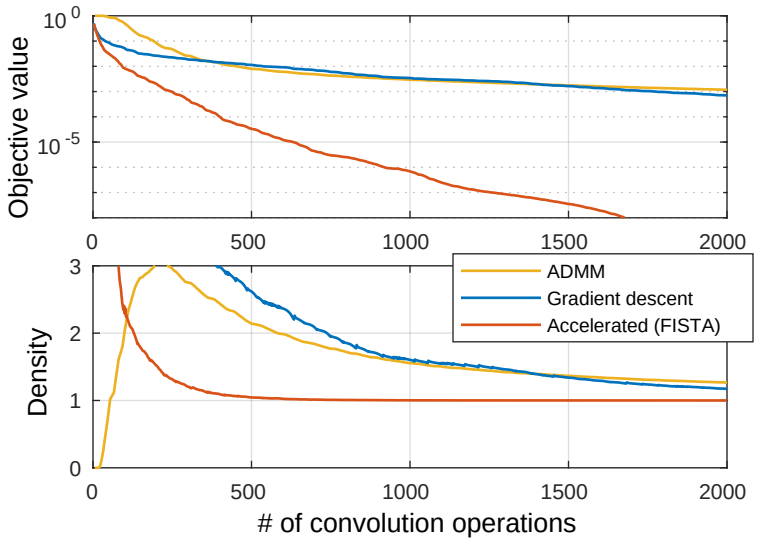

Figure 2.3: Comparison of optimization algorithms. Comparison of three algorithms—alternating directions method of multipliers (ADMM, yellow), proximal gradient descent (blue), and a form of accelerated proximal gradient descent (FISTA, red)—on a convex deconvolution problem. Runtime is measured in terms of the number of convolution operations, which is the most computationally intensive step. **Top:** Relative objective value $\frac{J(x)-J(x^\star)}{J(x_0)-J(x^\star)}$, where $J(x)$ is the objective function (2.1), x_0 is the starting point, and $x^\star$ is the optimal solution as determined by running the algorithms for several thousand more iterations. **Bottom:** Relative density $\text{nnz}(x)/\text{nnz}(x^\star)$, where $\text{nnz}(x)$ is the number of nonzero entries in x.

2.2.5 Optimization algorithm

Figure 2.3 compares the performance[4] of three popular optimization algorithms: alternating direction method of multipliers (ADMM, yellow), proximal gradient descent (blue), and a form of accelerated gradient descent known as the fast iterative shrinkage-thresholding algorithm (FISTA, red).

ADMM (Boyd et al., 2011) is a very popular optimization algorithm that has shown promising results in a variety of difficult optimization problems (Swaminathan and Murray, 2014; Horowitz, Papusha, and Burdick, 2014). It has attracted considerable attention for deconvolution applications (Bristow, Eriksson, and Lucey, 2013; Heide, Heidrich, and Wetzstein, 2015; Wohlberg, 2016; Wang et al., 2018) because the proximal operator of the approximation error $f(x)$ has a simple expression in frequency domain. However, since the proximal operator for the regularizer $g(x)$ still must be evaluated in time domain, this does not actually reduce the number of FFT

[4] As a disclaimer, note that these results are for a different sparse deconvolution problem, involving a larger set of kernels spanning a wide range of frequencies, and using the convex l_1 regularizer. Given the superiority of the accelerated algorithm in these tests, I decided not to re-implement the other algorithms after switching to the spike detection problem with the nonconvex regularizer.

operations per iteration as compared to gradient-based methods, and furthermore precludes the use of the overlap-add technique to replace a large FFT with several smaller transforms. Ignoring this last concern, this procedure requires the equivalent of two convolution operations per iteration.

Proximal gradient descent (Parikh and Boyd, 2014) is an extension of standard gradient descent techniques to cases where the objective function may be written as the sum of a smooth term $f(x)$ and a potentially nonsmooth term $g(x)$ that admits a simple proximal operator. This requires at least two convolutions per iteration, but sometimes more due to backtracking[5], and averaged 2.3 convolutions per iteration.

FISTA (Beck and Teboulle, 2009) is a form of accelerated gradient descent (see Becker, Candès, and Grant, 2011, for an excellent overview), which adds a momentum-like term that can help speed up convergence. I tried a variety of accelerated gradient methods (Nesterov, 2013; Lan, Lu, and Monteiro, 2011; Auslender and Teboulle, 2006) and found their performance to be essentially identical (note that computing the gradient is far more expensive than evaluating prox_g in this problem). FISTA was the easiest to implement and the most memory-efficient. This averaged 3.8 convolutions per iteration.

Despite requiring more convolutions per iteration, FISTA still converges much faster than the other algorithms tested (Figure 2.3, top). More importantly, it arrives at a sparse solution much sooner than the others (Figure 2.3, bottom). Here we see that the accelerated algorithm produces a solution with near-optimal sparsity within 500 operations, while ADMM and gradient descent still contain twice as many nonzero entries. Even after 2000 operations, these non-accelerated methods still contain substantially more nonzero entries than optimal.

Using FISTA as the optimization algorithm, I could handle problem sizes up to $T = 1$ million before running out of GPU memory. Since typical chronic datasets

[5]Backtracking is a way of automatically adjusting the step size. If a certain condition is not met, then we backtrack, i.e., reduce the step size (by increasing our Lipschitz constant estimate L) and try again. While we are on this topic, note that the usual backtracking termination criterion

$$f(x_{i+1}) \leq f(x_i) + \langle \nabla_f(x_i), x_{i+1} - x_i \rangle + \frac{L_i}{2} \|x_{i+1} - x_i\|^2$$

suffers from numerical cancellation issues since $f(x_i)$ and $f(x_{i+1})$ may be much larger than the other terms in this expression. But since $f(x) = \frac{1}{2}\|\mathcal{A}x - b\|_w^2$, this is equivalent to

$$\|\mathcal{A}(x_{i+1} - x_i)\|_w^2 \leq L_i \|x_{i+1} - x\|^2.$$

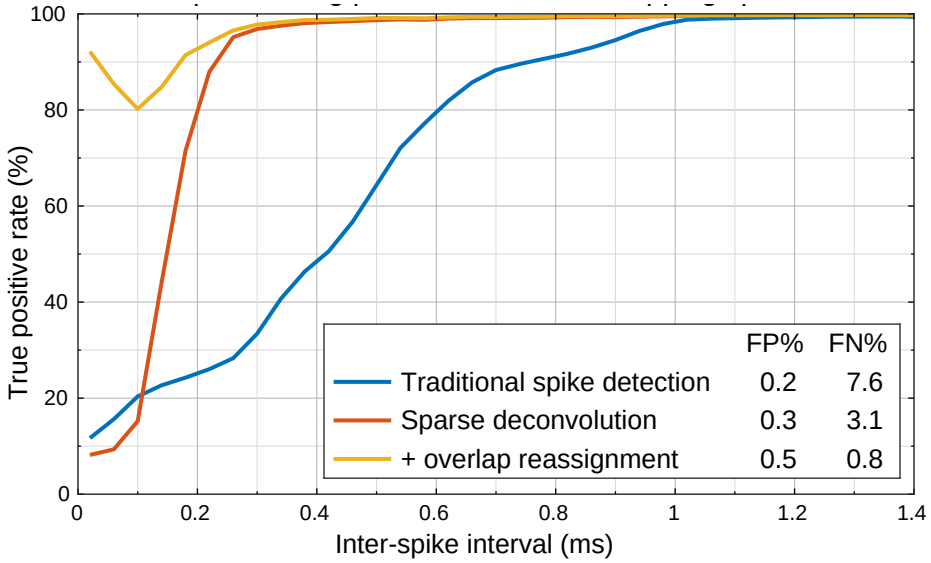

Figure 2.4: Spike detection performance on a hybrid ground truth dataset. Fraction of donor spikes that were detected and assigned to the correct source (true positive rate) as a function of the time between spikes (inter-spike interval). The overall false discovery rate (FP%) and false negative rate (FN%) for each method are reported in the legend.

range from 100 million to a few billion samples in duration, I processed these datasets in chunks and merged them together.

2.3 Results

To evaluate the spike detection performance, I created a "hybrid ground truth" dataset (section 3.4.3) by adding known "donor" spikes to a relatively quiet recording. We can then perform spike detection and clustering and measure the number of donor spikes that were detected and assigned to the correct source (Figure 2.4).

Traditional spike detection using the nonlinear energy operator as the detection heuristic and PCA for dimensionality reduction (blue) performs well on non-overlapping spikes, but struggles with spikes that are less than 1 ms apart. Since this dataset contains a high incidence of overlapping spikes, this results in an overall false negative rate of 7.6%.

The sparse deconvolution approach presented in this chapter (red) is able to reliably deconvolve spikes down to 0.3 ms (for reference, the data shown in figure 2.1 shows a pair of spikes separated by 0.3 ms), reducing the false negative rate by more than half. Remarkably, it is able to deconvolve these spikes without any clustering or other characterization of the spike sources.

Beyond this, it becomes increasingly difficult to distinguish two overlapping spikes

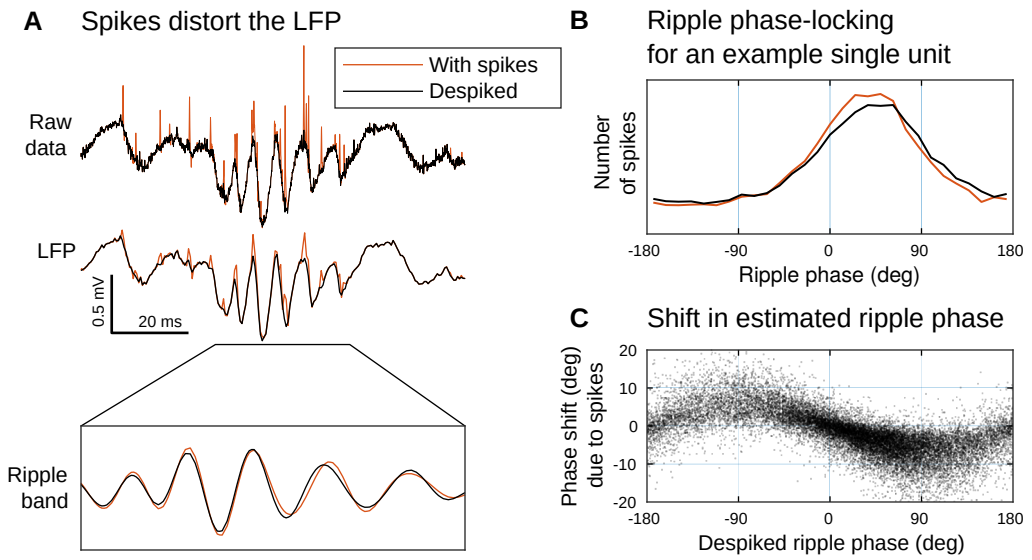

Figure 2.5: Spike removal for LFP analysis. Failing to remove spikes may bias subsequent analysis of the local field potential (LFP). **(A)** *Top:* After detecting spikes using our approach, these spikes can be removed from the original raw data (red) to produce a despiked signal (black). *Middle:* This broadband data is lowpass filtered to obtain an estimate of the LFP, a measure of aggregate network activity. However, if spikes are not removed prior to filtering, then they will distort our LFP estimate. *Bottom:* A close-up after filtering in the ripple band (100–200 Hz). Note that failing to remove spikes has changed the amplitude and phase of the observed ripple. **(B)** Ripple phase-locking analysis for a single unit (a putative CA1 interneuron), showing the histogram of spikes according to the phase of the ripple oscillation, as estimated from despiked LFP (black) and a with-spikes LFP (red). Note that the peak of the with-spikes histogram is narrower and shifted towards zero. **(C)** Shift in the estimated ripple phase due to the presence of spikes. Each dot corresponds to a single spike from the unit in panel B, and shows the phase shift $\theta_{\text{with-spikes}} - \theta_{\text{despiked}}$ (i.e., the difference between the despiked and with-spikes estimates of ripple phase) as a function of the ripple phase at which this spike was observed. This shows that the with-spikes LFP consistently biases the estimate of ripple phase towards zero.

from one very large spike. Resolving closely-overlapping spikes inevitably requires some model of the individual sources (i.e., some idea of what the spikes produced by each putative neuron will look like). One approach, described in section 3.5, augments the fitted model with additional components that correspond to overlap between pairs of neurons at various lags. This overlap reassignment procedure (yellow) is able to correctly reassign most of these closely-overlapping spikes based only on their feature space representation (i.e., without recourse to the raw data), bringing the overall false negative below 1%.

This deconvolution-based spike detection has another useful application: spike removal. Traditionally, the broadband signal (spikes and all) is simply lowpass

filtered and downsampled to form an estimate of the local field potential (LFP). However, spikes still contain a fair amount of energy at lower frequencies, and this lowpass filtering does not completely remove their influence on the recorded signal (Figure 2.5A). Instead, our spike detection method allows us to analyze the residual $b - \mathcal{A}x$ as a "despiked" version of the recorded signal. This is particularly relevant for the analysis of ripples (LFP oscillations in the range of 100–200 Hz), where the activity of ripple-phase-locked neurons may produce consistent biases in the apparent phase and amplitude of ripple oscillations (Figure 2.5B and C).

2.4 Discussion

In this chapter, I have presented a spike detection algorithm that follows the modular approach of traditional spike detection—producing a feature-space representation of the detected spikes without requiring any information about the clusters present in the recording—while offering improved resolution of overlapping spikes.

At the moment, the main drawback of this approach is that it is quite slow, running only slightly faster than realtime on tetrode data (an hour-long recording takes nearly one hour to process). Although the current software implementation has some room for improvement, that alone cannot produce the 10 or 100x speedup that would be necessary for large-scale deployment of this technique. Instead, a greedy approach such as orthogonal matching pursuit (Pati, Rezaiifar, and Krishnaprasad, 1993) may be more appropriate.

Finally, I will note that the generality of this technique—the fact that, aside from the choice of spike basis waveforms, nothing ties this approach to the specific problem of spike detection—allows us to consider using this approach for the detection and clustering of other transient events.

Chapter 3

SPIKE SORTING USING A MIXTURE OF DRIFTING T-DISTRIBUTIONS

Shan, K. Q., E. V. Lubenov, and A. G. Siapas (2017). "Model-based spike sorting with a mixture of drifting t-distributions". *Journal of neuroscience methods* 288, pp. 82–98. DOI: 10.1016/j.jneumeth.2017.06.017.

This chapter is based on previously-published material. In preparing this chapter, I have condensed and rearranged the sections to be more amenable to selective reading. I have also expanded section 3.5 to include results using a sparse deconvolution approach to spike detection.

Section 3.1 provides some additional background on the spike sorting problem. Section 3.2 describes the MoDT model and performs some benchmarking of our software implementation of the model fitting algorithm. Section 3.3 evaluates the model on several thousand hours of chronic tetrode recordings to show that it fits the empirical data substantially better than a mixture of Gaussians. Section 3.4 then demonstrates that the MoDT-based estimate of misclassification error is more accurate than previous unit isolation metrics, and Section 3.5 discusses the issue of temporally overlapping spikes.

3.1 Introduction

Chronic extracellular recordings offer access to the spiking activity of neurons over the course of days or even months. However, the analysis of extracellular data requires a process known as spike sorting, in which extracellular spikes are detected and assigned to putative sources. Despite many decades of development, there is no universally-applicable spike sorting algorithm that performs best in all situations.

Approaches to spike sorting can be divided into two categories: model-based and non-model-based (or non-parametric). In the model-based approach, one constructs a generative model (e.g., a mixture of Gaussian distributions) that describes the probability distribution of spikes from each putative source. This model may be used for spike sorting by comparing the posterior probability that a spike was generated by each source. Fitting of such models may be partially or fully automated using

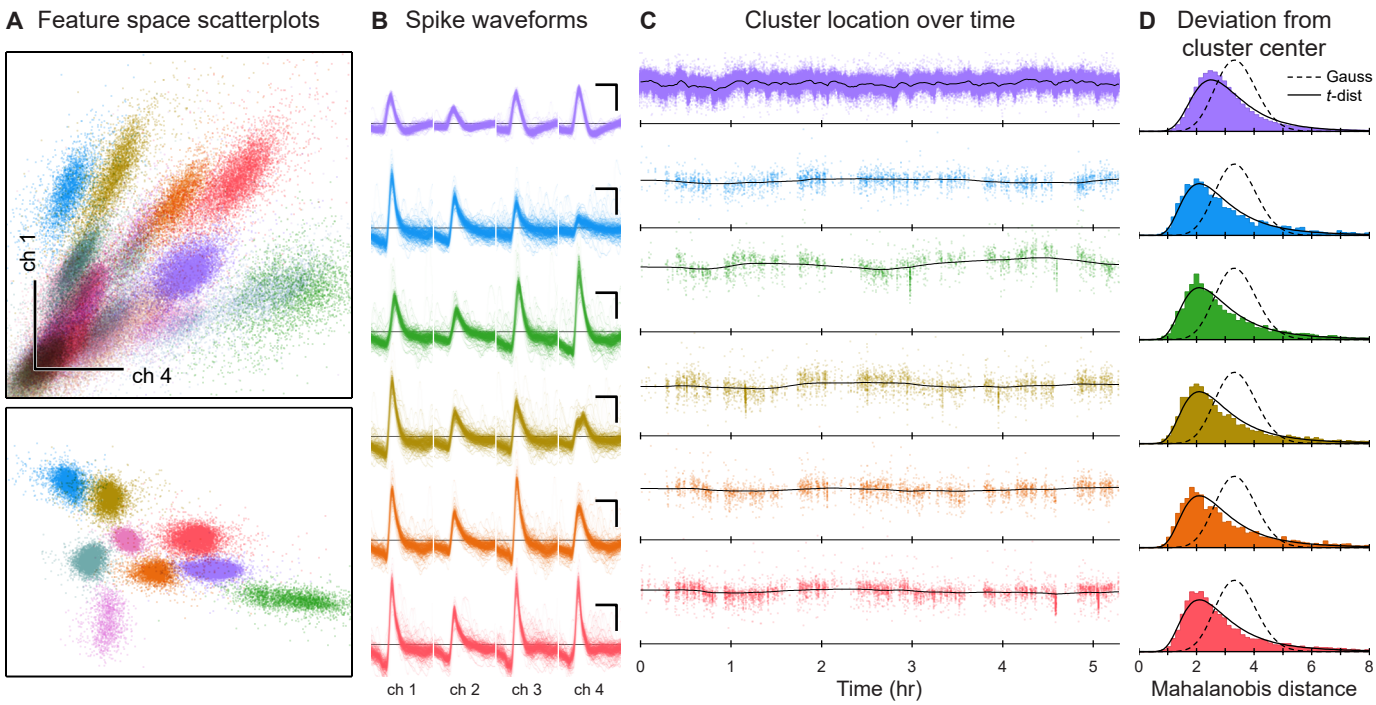

Figure 3.1: Extracellular recordings contain drifting, heavy-tailed clusters. (**A**) Scatterplots of spikes in feature space, color-coded by putative identity. Spike waveforms recorded on 4 tetrode channels were projected onto a 12-dimensional feature space using 3 principal components from each channel. Top: scatterplot of the first principal component from channels 1 and 4. Bottom: a different projection of the data, showing only the best-isolated single units. (**B**) Spike waveforms (inverted polarity) for six example units. Scale bar: 200 µV, 0.5 ms. (**C**) Cluster drift in feature space. y-axis shows one of the 12 feature space dimensions. Black line indicates the cluster center fitted using the MoDT model. (**D**) Distribution of the non-squared Mahalanobis distance (δ) from the fitted cluster center to the observed spikes. Lines indicate the theoretical distributions for Gaussian and t-distributed spikes.

maximum likelihood or Bayesian methods, and the model also provides an estimate of the misclassification error.

In the non-parametric approach, spike sorting is treated solely as a classification problem. These classification methods may range from manual cluster cutting to a variety of unsupervised learning algorithms. Regardless of the method used, scientific interpretation of the sorted spike train still requires reliable, quantitative measures of unit isolation quality. Often, these heuristics either explicitly (Hill, Mehta, and Kleinfeld, 2011) or implicitly (Schmitzer-Torbert et al., 2005) assume that the spike distribution follows a mixture of Gaussian distributions.

However, a mixture of Gaussians does not adequately model the cluster drift and heavy tails that are observed in experimental data (Figure 3.1). Cluster drift is

a slow change in the shape and amplitude of recorded waveforms (Figure 3.1C), usually ascribed to motion of the recording electrodes relative to the neurons (Snider and Bonds, 1998; Lewicki, 1998). This effect may be small for short recordings ($<$ 1 hour), but can produce substantial errors if not addressed in longer recordings (Figure 3.6). Even in the absence of drift, spike residuals have heavier tails than expected from a Gaussian distribution, and may be better fit using a multivariate t-distribution (Figure 3.1D and Figure 3.5; see also Shoham, Fellows, and Normann, 2003; Pouzat et al., 2004).

To address these issues, we model the spike data as a mixture of drifting t-distributions (MoDT). This model builds upon previous work that separately addressed the issues of cluster drift (Calabrese and Paninski, 2011) and heavy tails (Shoham, Fellows, and Normann, 2003), and we have found the combination to be extremely powerful for modeling and analyzing experimental data. We also discuss the model's robustness to outliers, provide a software implementation of the fitting algorithm, and discuss some methods for reducing errors due to spike overlap.

We used the MoDT model to perform spike sorting on 34,850 tetrode-hours of chronic tetrode recordings (4.3 billion spikes) from the rat hippocampus, cortex, and cerebellum. Using these experimental data, we evaluate the assumptions of our model and provide recommended values for the model's user-defined parameters. We also analyze how the observed cluster drift may impact the performance of spike sorting methods that assume stationarity. Finally, we evaluate the accuracy of MoDT-based estimates of misclassification error and compare this to the performance of other popular unit isolation metrics in the presence of empirically-observed differences in firing rate and spike variability.

Table 3.1: Mathematical notation. Lowercase bold letters ($\boldsymbol{y}_n, \boldsymbol{\mu}_{kt}$) denote D-dimensional vectors, and uppercase bold letters ($\boldsymbol{C}_k, \boldsymbol{Q}$) denote $D \times D$ symmetric positive definite matrices.

Dimensions	
D	Number of dimensions
N	Number of spikes
K	Number of clusters
T	Number of time frames
Given data	
$\boldsymbol{y}_n$	Observed spike n
t_n	Time frame in which spike n occurred
w_n	Weighting of spike n (multiplier applied to log-likelihood)
User-defined constants	
ν	t-distribution degrees-of-freedom parameter
$\boldsymbol{Q}$	Drift regularization parameter
Fitted model parameters	
α_k	Mixing proportion for cluster k
$\boldsymbol{\mu}_{kt}$	Location parameter for cluster k in time frame t
$\boldsymbol{C}_k$	Scale parameter for cluster k
Latent variables introduced by EM procedure	
z_{nk}	Posterior probability that spike n belongs to cluster k
u_{nk}	Scaling variable introduced in formulating the t distribution as a Gaussian-Gamma compound distribution

3.2 Mixture of drifting t-distributions (MoDT) model

In this section, I will introduce the mixture of drifting t-distributions (MoDT) model (section 3.2.1), describe an EM algorithm for fitting this model (section 3.2.2), and analyze the computational scaling of this algorithm (section 3.2.3). Section 3.2.4 contains some stray remarks about how the multivariate t-distribution may be used for robust covariance estimation. Finally, section 3.2.5 discusses some potential extensions to this MoDT model.

3.2.1 Model description

Spike sorting begins with spike detection and feature extraction. During these preprocessing steps, spikes are detected as discrete events in the extracellular voltage trace and represented as points $\boldsymbol{y}_n$ in some D-dimensional feature space.

The standard mixture of Gaussians (MoG) model treats this spike data $\boldsymbol{y}_n$ as samples

drawn from a mixture distribution with PDF given by

$$f_{\text{MoG}}(\boldsymbol{y}_n; \phi) = \sum_{k=1}^{K} \alpha_k f_{\text{mvG}}(\boldsymbol{y}_n; \boldsymbol{\mu}_k, \boldsymbol{C}_k),$$

where $\phi = \{\ldots, \alpha_k, \boldsymbol{\mu}_k, \boldsymbol{C}_k, \ldots\}$ is the set of fitted parameters, K is the number of mixture components, α_k are the mixing proportions, and $f_{\text{mvG}}(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{C})$ is the PDF of the multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{C}$:

$$f_{\text{mvG}}(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{C}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{C}|^{1/2}} \exp\left[-\frac{1}{2}\delta^2(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{C})\right].$$

For notational convenience, let δ^2 denote the squared Mahalanobis distance

$$\delta^2(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{C}) = (\boldsymbol{y} - \boldsymbol{\mu})^\intercal \boldsymbol{C}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}).$$

We make two changes to this model. First, we replace the multivariate Gaussian distribution with the multivariate t-distribution. The PDF for this distribution, parameterized by location $\boldsymbol{\mu}$, scale $\boldsymbol{C}$, and degrees-of-freedom ν, is given by

$$f_{\text{mvt}}(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{C}, \nu) \frac{1}{(\nu\pi)^{D/2}|\boldsymbol{C}|^{1/2}} \frac{\Gamma(\frac{\nu+D}{2})}{\Gamma(\frac{\nu}{2})} \left[1 + \frac{1}{\nu}\delta^2(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{C})\right]^{-\frac{\nu+D}{2}}.$$

Second, we break up the dataset into T time frames (we used a frame duration of 1 minute) and allow the cluster location $\boldsymbol{\mu}$ to change over time. The mixture distribution becomes

$$f_{\text{MoDT}}(\boldsymbol{y}_n; \phi, \nu) = \sum_{k=1}^{K} \alpha_k f_{\text{mvt}}(\boldsymbol{y}_n; \boldsymbol{\mu}_{kt_n}, \boldsymbol{C}_k, \nu),$$

where $t_n \in \{1, \ldots, T\}$ denotes the time frame for spike n. We use a common ν parameter for all components and have chosen to treat it as a user-defined constant. The fitted parameter set is thus $\phi = \{\ldots, \alpha_k, \boldsymbol{\mu}_{k1}, \ldots, \boldsymbol{\mu}_{kT}, \boldsymbol{C}_k, \ldots\}$.

In order to enforce consistency of the component locations across time, we introduce a prior on the location parameter that penalizes large changes over consecutive time steps. This prior has a joint PDF proportional to

$$f_{\text{prior}}(\boldsymbol{\mu}_{k1}, \ldots, \boldsymbol{\mu}_{kT}) = \prod_{t=2}^{T} f_{\text{mvG}}(\boldsymbol{\mu}_{kt} - \boldsymbol{\mu}_{k(t-1)}; \boldsymbol{0}, \boldsymbol{Q}), \tag{3.1}$$

where $\boldsymbol{Q}$ is a user-defined covariance matrix that controls how much the clusters are expected to drift.

3.2.2 EM algorithm for model fitting

Assuming independent spikes and a uniform prior on the other model parameters, we can obtain the maximum *a posteriori* (MAP) estimate of the fitted parameters ϕ by maximizing the log-posterior, which is equivalent (up to an additive constant) to the following:

$$L(\phi) = \sum_{n=1}^{N} w_n \log f_{\text{MoDT}}(\boldsymbol{y}_n; \phi) + \sum_{k=1}^{K} \log f_{\text{prior}}(\boldsymbol{\mu}_{k1}, \ldots, \boldsymbol{\mu}_{kT}).$$

Note that we have introduced a weight w_n for each spike. This allows us to fit the model to a weighted subset of the data while remaining consistent with the full dataset (Feldman, Faulkner, and Krause, 2011).

As with most mixture distributions, it is intractable to optimize $L(\phi)$ directly. However, by introducing additional latent random variables, we obtain a "complete-data" log-posterior $L_c(\phi, Z, U)$ that allows us to decompose the problem and optimize it using an expectation-maximization (EM) algorithm (McLachlan and Peel, 2000).

In the E-step, we compute the expected value of L_c assuming that these latent variables follow their conditional distribution given the observed data and the fitted parameters $\hat{\phi}$ from the previous EM iteration. The conditional expectations of these latent variables are given by:

$$z_{nk} = \frac{\hat{\alpha}_k f_{\text{mvt}}(\boldsymbol{y}_n; \hat{\boldsymbol{\mu}}_{kt_n}, \hat{\boldsymbol{C}}_k, \nu)}{\sum_\kappa \hat{\alpha}_\kappa f_{\text{mvt}}(\boldsymbol{y}_n; \hat{\boldsymbol{\mu}}_{\kappa t_n}, \hat{\boldsymbol{C}}_\kappa, \nu)}, \tag{3.2}$$

$$u_{nk} = \frac{\nu + D}{\nu + \delta^2(\boldsymbol{y}_n; \hat{\boldsymbol{\mu}}_{kt_n}, \hat{\boldsymbol{C}}_k)}. \tag{3.3}$$

The z_{nk} correspond to the posterior probability that spike n was produced by component k, and may thus be interpreted as a soft-assignment of spikes to clusters (i.e., putative neurons). The u_{nk} arises from the formulation of the t-distribution as a Gaussian-Gamma compound distribution and may be interpreted as a scaling variable that "Gaussianizes" the multivariate t-distribution. In the Gaussian case (the limit of a t-distribution as $\nu \to \infty$), we have $u_{nk} = 1$ for all spikes. For finite ν, note that u_{nk} decreases as the Mahalanobis distance δ increases.

Next we can compute the conditional expectation of $L_c(\phi, Z, U)$ over these latent variables. Following Peel and McLachlan (2000), we find that this is equivalent (up

to an additive constant) to

$$J(\phi;\hat{\phi}) = \sum_{n=1}^{N} w_n \sum_{k=1}^{K} z_{nk}\left[\log\alpha_k - \frac{1}{2}\log|\boldsymbol{C}_k| - \frac{1}{2}u_{nk}\delta^2(\boldsymbol{y}_n;\boldsymbol{\mu}_{kt_n},\boldsymbol{C}_k)\right] + \sum_{t=2}^{T} -\frac{1}{2}\delta^2(\boldsymbol{\mu}_{kt} - \boldsymbol{\mu}_{k(t-1)};\boldsymbol{0},\boldsymbol{Q}).$$

In the M-step, we maximize $J(\phi,\hat{\phi})$ with respect to the fitted parameters. The optimal value for the mixing proportions α is simply a weighted version of the mixture of Gaussians (MoG) M-step update:

$$\arg\max_{\alpha_k} J(\phi;\hat{\phi}) = \frac{\sum_n w_n z_{nk}}{\sum_n w_n}. \tag{3.4}$$

The optimal value for the cluster scale parameter $\boldsymbol{C}$ is also similar to the MoG case, but each spike is additionally scaled by u_{nk}:

$$\arg\max_{\boldsymbol{C}_k} J(\phi;\hat{\phi}) = \frac{\sum_n w_n z_{nk} u_{nk}(\boldsymbol{y}_n - \boldsymbol{\mu}_{kt_n})(\boldsymbol{y}_n - \boldsymbol{\mu}_{kt_n})^\intercal}{\sum_n w_n z_{nk}}. \tag{3.5}$$

For the cluster location parameters $\boldsymbol{\mu}$, note that $J(\phi,\hat{\phi})$ is quadratic with respect to $\boldsymbol{\mu}$ and its maximum occurs where the gradient $\nabla_{\boldsymbol{\mu}} J(\phi,\hat{\phi}) = 0$. We can therefore find the optimal $\boldsymbol{\mu}$ by solving the following linear system of equations:

$$\nabla_{\boldsymbol{\mu}_k} J(\phi,\hat{\phi}) = \begin{bmatrix}\boldsymbol{b}_{k1}\\ \boldsymbol{b}_{k2}\\ \vdots\\ \boldsymbol{b}_{kT}\end{bmatrix} - \boldsymbol{A}\begin{bmatrix}\boldsymbol{\mu}_{k1}\\ \boldsymbol{\mu}_{k2}\\ \vdots\\ \boldsymbol{\mu}_{kT}\end{bmatrix} = 0, \tag{3.6}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{M}_{k1} + \boldsymbol{Q}^{-1} & -\boldsymbol{Q}^{-1} & & \\ -\boldsymbol{Q}^{-1} & \boldsymbol{M}_{k2} + 2\boldsymbol{Q}^{-1} & -\boldsymbol{Q}^{-1} & \\ & -\boldsymbol{Q}^{-1} & \ddots & \ddots \\ & & \ddots & \boldsymbol{M}_{kT} + \boldsymbol{Q}^{-1} \end{bmatrix}$$

and

$$\boldsymbol{M}_{kt} = \boldsymbol{C}_k^{-1} \sum_{n\,:\,t_n = t} w_n z_{nk} u_{nk}$$

$$\boldsymbol{b}_{kt} = \boldsymbol{C}_k^{-1} \sum_{n\,:\,t_n = t} w_n z_{nk} u_{nk}\, \boldsymbol{y}_n.$$

EM fitting of the MoDT model thus consists of iterative evaluation of equations (3.2) through (3.6). A few remarks on this procedure:

1. Most of the z_{nk} end up very close to zero, and ignoring these spikes in the M-step can reduce the complexity of that operation. However, the sparsity needs to be very high ($K > 20$, as a rule of thumb) to outweigh the efficiency advantages of the highly-optimized numerical routines for dense linear algebra. We found that applying a threshold on z_{nk} produced more accurate results than "hard EM" (in which each spike is assigned to only one cluster), which tends to underestimate the covariance of highly-overlapping clusters.
2. The scaling variable u_{nk} acts as an additional weighting term in the optimization of $\boldsymbol{\mu}$ and $\boldsymbol{C}$. Since u_{nk} decreases as spike n gets far away from cluster k, any outliers are automatically discounted during the fitting process. As a result, the fitted parameters are considerably more robust to the presence of outliers than in the Gaussian case (Lange, Little, and Taylor, 1989; Peel and McLachlan, 2000).
3. The optimal value of $\boldsymbol{\mu}$ depends on the value of $\boldsymbol{C}$ and vice-versa. Although the standard EM algorithm calls for maximizing $J(\phi, \hat{\phi})$ over all ϕ, the convergence of a generalized EM algorithm requires only that we improve upon the previous $\hat{\phi}$ (Dempster, Laird, and Rubin, 1977). Therefore we need not simultaneously optimize $\boldsymbol{\mu}$ and $\boldsymbol{C}$, but may instead update them one at a time.
4. Although equation (3.6) involves solving a $DT \times DT$ linear system, its sparsity structure allows us to solve for $\boldsymbol{\mu}$ with a complexity that scales linearly with T. For example, $\boldsymbol{M}_{kt}$ and $\boldsymbol{b}_{kt}$ appear in the *information filter*, an alternative paramaterization of the Kalman filter (see, e.g., Anderson and Moore, 1979). The acausal problem can then be solved using a backwards pass (Rauch, Tung, and Striebel, 1965). However, we have found that it is faster to solve equation (3.6) using standard numerical linear algebra routines for solving banded positive semi-definite matrices (i.e., LAPACK `dpbsv`).

3.2.3 Software implementation and computational scaling

A MATLAB implementation of this EM algorithm, along with a demo script, is available at `https://github.com/kqshan/MoDT`. In this section, we measure the runtime on a desktop workstation with an Intel Core i5-7500 CPU, 32 GB of memory, and an NVidia GeForce GTX 1080 graphics card, running MATLAB R2017a (64-bit) on Ubuntu 16.04.2 with CUDA toolkit 8.0, using double-precision arithmetic.

Table 3.2: Computational runtime for model fitting. Time required to perform 20 EM iterations on the sample dataset shown in Figure 3.1 (D=12, K=26, N=1.9 million). `fitgmdist` is a mixture of Gaussians fitting routine that is part of the MATLAB Statistics and Machine Learning Toolbox. `modt` is a MATLAB implementation of our EM algorithm that may be downloaded from `https://github.com/kqshan/MoDT`. In addition to fitting the richer MoDT model, it supports two additional features (data weights and GPU computing) that can dramatically reduce fitting times.

Model type	Algorithm description	Runtime (s)
MoG	`fitgmdist`	123.43
MoG	`modt` in Gaussian mode	103.53
MoDT	`modt`	104.56
MoDT	`modt` on GPU	7.06
MoDT	`modt`, 5% subset	5.74
MoDT	`modt`, 5% subset, GPU	1.24

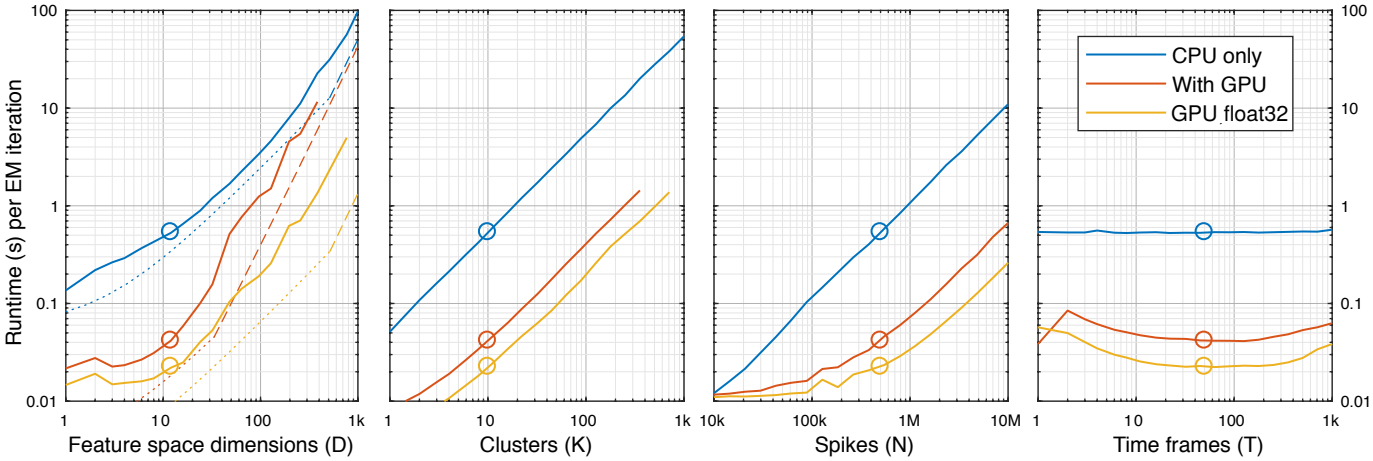

Figure 3.2: Scaling of computational runtime with model dimensions. Starting from a baseline of D = 12, K = 10, N = 500,000, T = 50, we varied each dimension and measured the runtime on CPU (blue) and GPU (red). We also measured GPU runtime using single-precision (32-bit floating point) arithmetic (yellow). For D, we show the theoretical limits imposed by the hardware's computing power (dashed line) and memory throughput (dotted line). Peak memory usage is $(5K + 4D)N$ elements, and the GPU line ends when we run out of GPU memory (8 GB).

Our implementation offers a mild speedup over the MATLAB built-in mixture-of-Gaussians fitting routine, despite fitting a more complex model (Table 3.2). In addition, it supports the use of a weighted training subset, which offers a proportional reduction in runtime at the expense of model accuracy, and supports the use of GPU computing using the NVidia CUDA computing platform.

How does this runtime scale with the model dimensions D, K, N, and T? The

most computationally intensive operations are computing the Mahalanobis distance (D^2KN), updating the cluster location $\boldsymbol{\mu}$ ($DKN + D^3KT$), and updating the scale parameter $\boldsymbol{C}$ (D^2KN). Since the number of spikes is typically much larger than the number of time frames or dimensions ($N \gg DT$), we expect the fitting time to scale as D^2KN overall. To test these scaling laws, we measured the runtime while varying each model dimension (Figure 3.2).

Surprisingly, we found that the CPU runtime scaled almost linearly with D. This is because the CPU's memory throughput (17 GB/s), rather than its computing power (218 GFLOPS), is the limiting factor[1] when $D < 500$, and memory access scales linearly with D.

The GPU's higher memory throughput (320 GB/s) affords a substantial speedup on small D. Like many consumer-grade GPUs, this device's single-precision computing power (8.2 TFLOPS) is substantially higher than its double-precision capacity (257 GFLOPS), and switching to single-precision arithmetic dramatically increases performance on compute-limited tasks.

As expected, we found the runtime scales linearly with K and N, and the effect of T is negligible. The GPU shows similar trends, but with reduced efficiency at small N and T due to poor utilization of the hardware resources.

Finally, we measured the peak memory usage to be approximately 5 $N \times K$ matrices and 4 $D \times N$ matrices, for a total of $8(5K + 4D)N$ bytes (double-precision). Fitting larger datasets may require using a weighted subset of spikes and/or performing the optimization in batches.

3.2.4 Robust covariance estimation using the t-distribution

In this section, let us take a brief intermission from our consideration of drifting, heavy-tailed data and look at the case where the underlying data are stationary and Gaussian, but unmodeled noise or artifacts may be present. In such cases, standard estimates of the data covariance may be unduly influenced by these outliers (Figure 3.3A). Since the data covariance is a critical component of multivariate statistics, it is important to have a covariance estimator that is robust to the presence of outliers.

One very popular robust covariance estimator is the minimum covariance determinant (MCD) estimator (Rousseeuw and Driessen, 1999). In this approach, the data points

[1] In other words, the *arithmetic intensity* of this operation is fairly low.

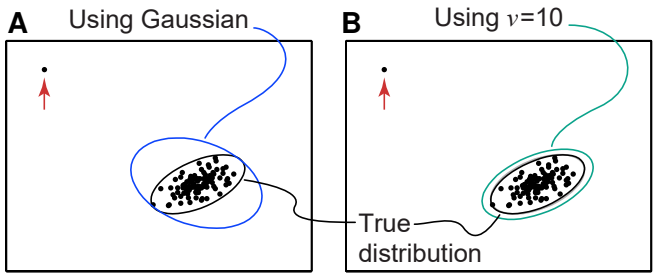

Figure 3.3: Robust covariance estimation using the t-distribution. The t-distribution may also be used to derive robust estimates of Gaussian parameters. In this example, we generated 100 points from a Gaussian distribution and added a single outlier (red arrow). **(A)** This has stretched out the estimated covariance when using a Gaussian fit. **(B)** The fitted t-distribution is largely unaffected by the outlier. Equation (3.7) can then be used to convert the fitted t-distribution scale parameter into an equivalent Gaussian covariance parameter.

are weighted according to their distance from the estimated center—specifically, points with a Mahalanobis distance below some threshold are given a weight of 1 and points beyond that threshold are given a weight of 0—and then the weighted covariance is multiplied by a correction factor to achieve consistency with an non-truncated Gaussian distribution. Finding the appropriate set of points to include in this estimate requires an iterative procedure, preferably with multiple random initializations, and may be very slow to converge when there are many data points.

As we noted in the previous section, fitting the multivariate t-distribution also discounts any points that are far away from the estimated center. In Figure 3.3B, we have fit this same dataset using a t-distribution, and we can see that it automatically rejects the influence of the outlier. However, this has also caused us to overestimate the distribution tails: note how the fitted t-distribution's 99% confidence ellipse (green) is inflated relative to the true ellipse (black).

Rather than using the t-distribution directly, can we use the fitted parameters $\boldsymbol{\mu}$ and $\boldsymbol{C}$ to derive robust estimates of the Gaussian mean and covariance? As the number of samples $N \to \infty$, the fitted $\boldsymbol{\mu}$ converges to the Gaussian distribution's true mean, but the fitted $\boldsymbol{C}$ is a biased estimator of the Gaussian distribution's covariance parameter. However, we can compute a correction factor by solving for β in

$$\int_0^\infty \frac{\nu + D}{\beta\nu + x} x^{D/2} e^{-x/2}\, dx = 2^{D/2}\Gamma(D/2)D, \tag{3.7}$$

and then use $\beta\boldsymbol{C}$ as a robust estimate for the Gaussian covariance. The corresponding 99% confidence ellipse is also plotted in Figure 3.3B as a light grey ellipse, but it is visually obscured by the 99% confidence ellipse of the true distribution (black).

Compared to the MCD approach, fitting the t-distribution is much faster to converge and is less susceptible to local minima. This may be attributed to its use of "soft" weights u_{nk} rather than a hard threshold. Although the theoretical properties of a t-distribution-based robust covariance estimator remain to be characterized, it may be a promising approach for larger datasets.

3.2.5 Model extensions

The MoDT model we have presented consists of three components: a spike distribution model, a drift regularizer, and an EM fitting algorithm. These components may be modified or extended in several ways.

For example, we modeled the spike distribution using a t-distribution, which is elliptically symmetric. However, some neurons fire bursts in which subsequent spikes exhibit a reduced amplitude, producing a skewed distribution that has a longer tail in one direction. This one-dimensional skew may be modeled using a restricted multivariate skew t-distribution, which can be fitted using an EM algorithm (Lee and McLachlan, 2014).

We also used a very simple form of drift regularization, but this could be replaced with a more sophisticated model. High-density probes may benefit from a model that explicitly accounts for correlated changes in cluster location due to physical motion of a rigid multi-site probe. This would improve tracking of neurons with a low firing rate.

Finally, the EM algorithm has been widely studied and improved upon in many ways. The basic algorithm is is well-suited for large-scale data processing and is amenable to parallel computing on GPU hardware (Figure 3.2) or distributed computing using high-level data flow engines (Meng et al., 2016). As we consider applying this model to higher-dimensional data, a variety of algorithmic approximations may also be useful to consider, such as masked EM (Kadir, Goodman, and Harris, 2014), approximating $\boldsymbol{C}$ as the sum of low-rank and an isotropic component (Magdon-Ismail and Purnell, 2010), and partial E- or M-step updates (Neal and Hinton, 1998).

The MoDT model thus offers a modular framework that may be adapted to experimental needs.

3.3 Model validation using empirical data

In this section, I evaluate how well the MoDT model characterizes real data. Using a large collection of *in vivo* extracellular recordings from a variety of brain areas (section 3.3.1), I show that empirical data does indeed contain drifting clusters (section 3.3.2) and heavy-tailed residuals (section 3.3.3), and I use these data to provide recommendations for the corresponding user-defined parameters in the MoDT model. I also use these data to consider the consequences of using a stationary model for spike sorting (section 3.3.4). Finally, I discuss the implications on spike sorting of chronic recordings (section 3.3.5).

3.3.1 How these data were collected

To evaluate the MoDT model for spike sorting, we collected 34,850 tetrode-hours (34 terabytes) of chronic tetrode data by implanting 10 Long-Evans rats with 24-tetrode arrays targeting areas of the hippocampus, cortex, and cerebellum. Additional details regarding the data acquisition, spike detection, feature extraction, human-guided clustering, and computational infrastructure are described by Shan, Lubenov, and Siapas (2017), sections 2.4–2.7.

Overall, we detected and sorted 4.3 billion spikes, of which 852 million (representing 89,127 unit-hours) were deemed to come from 20,630 putative single units.

However, in our subsequent analysis we are faced with the following challenge: we wish to characterize the empirical properties of single units so that we can decide what assumptions to make during spike sorting, but we need to perform spike sorting in order to obtain single units to characterize. To break this circular dependency, we focused our analysis on the best-isolated units because these are the least sensitive to the spike sorting procedure used. This also ensures that we are studying the natural tails of the spike distribution rather than artificially truncating them via the spike classifier boundaries.

Therefore we applied a more conservative set of selection criteria than usual (Shan, Lubenov, and Siapas, 2017, section 2.9), which narrowed this down to 4,432 high-amplitude, well-isolated single units, accounting for 338 million spikes over 32,890 unit-hours.

3.3.2 Cluster drift in empirical data

We quantified the cluster drift by measuring the distance from each unit's current location (determined using a 40-minute moving average) to its location at the start

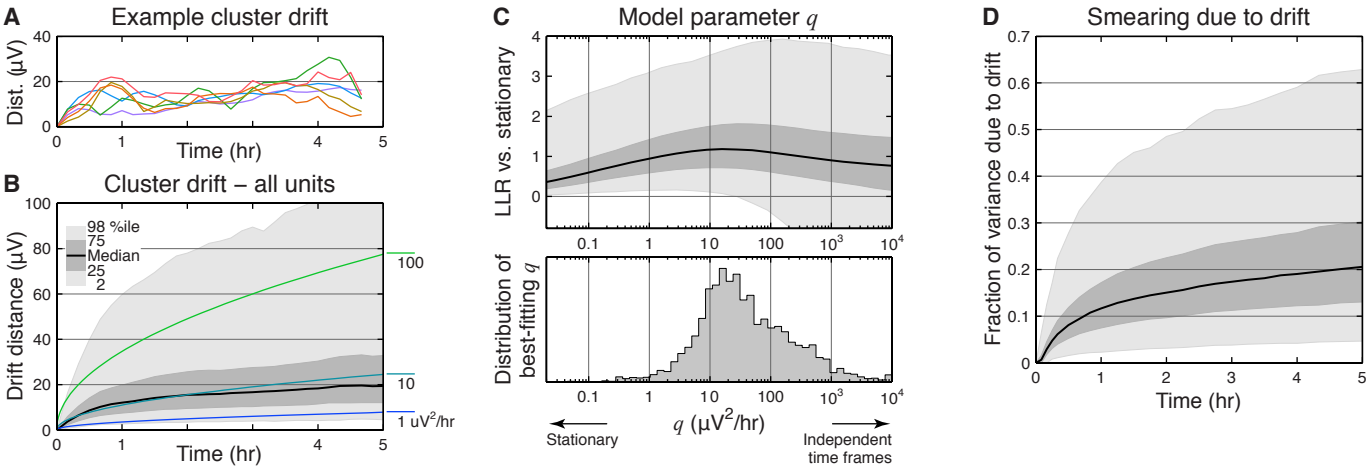

Figure 3.4: Cluster drift in well-isolated units. (**A**) Cluster drift of the 6 example units from Figure 3.1. Distances are measured from the unit's current location (determined using a 40-minute moving average) to its location at the start of the dataset. (**B**) Cluster drift of all well-isolated units in our analysis. Shaded regions indicate quantiles across units. Colored lines show expected drift distances for 3 different drift rates. (**C**) Effect of changing the model's drift regularization parameter $\boldsymbol{Q}$. We only considered isotropic matrices $\boldsymbol{Q} = q\boldsymbol{I}$. Top: Test-set log-likelihood ratio (LLR per spike) comparing the drifting vs. stationary model for all units. Bottom: Overall distribution of the best-fitting q for each unit. (**D**) If drift is not accounted for, it produces an apparent "smearing" of the spike distribution in feature space. This panel shows the fraction of spike variability that can be accounted for by cluster drift.

of the recording. Individual clusters may move closer or farther away from where they started (Figure 3.4A), but over all units, the average distance increases and the distribution spreads out (Figure 3.4B). These distances are measured in feature space units (μV).

Note that the cluster location prior in equation (3.1) corresponds to a Gaussian random walk with a constant rate of drift. However, the observed distribution of drift distances is much broader than expected from such a process, and so this aspect of the MoDT model should be treated as a regularizer rather than an attempt to accurately model the underlying phenomena.

The MoDT model parameter $\boldsymbol{Q}$ is a user-defined constant that controls this regularization. Figure 3.4C shows the effect of changing this parameter over a wide range of values. The log-likelihood ratio (LLR) is a measure of the MoDT model's quality of fit compared to a stationary alternative; values greater than zero indicate that the MoDT model provided a better fit. In this analysis, we measured the LLR on a holdout test set, since increasing the allowable drift always improves the quality of fit on the training set.

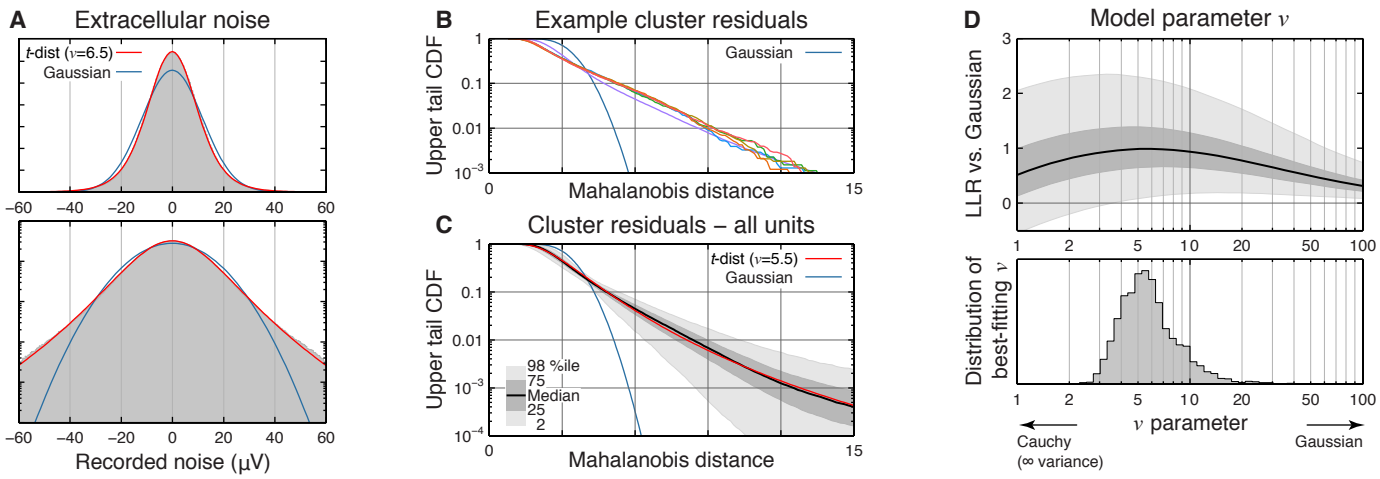

Figure 3.5: Heavy-tailed residuals in extracellular noise and well-isolated units. (**A**) Distribution of extracellular noise for an example tetrode channel during periods where no spikes were detected. Red and blue lines show a t-distribution and Gaussian fit, respectively. Bottom panel shows the same histogram with a logarithmic y-scale. (**B**) Upper tail CDF (fraction of a unit's spikes that lie beyond a given Mahalanobis distance from the cluster center) for the 6 example units from Figure 3.1. (**C**) Upper tail CDF for all well-isolated units. Shaded areas indicate quantiles across units. Theoretical distributions for a t-distribution and Gaussian are shown for reference. (**D**) Effect of changing the model's ν parameter, which controls the heavy-tailedness of the distribution. Top: Log-likelihood ratio (LLR per spike) comparing the t-distribution vs. Gaussian model for all units. Bottom: Overall distribution of best-fitting ν for each unit.

In this analysis we considered only isotropic matrices $\boldsymbol{Q} = q\boldsymbol{I}$, where q is a positive scalar and $\boldsymbol{I}$ is the identity matrix. When $q = 0$, the MoDT model is equivalent to a stationary (non-drifting) mixture model. As we increase q, we allow more drift in the model.

The optimal value of q varies across units (Figure 3.4C, bottom) and depends on the stability of the tetrode and the firing rate of the unit. Since we use the same value of q for all units, we chose a relatively low value (2 μV^2/hr), which is lower than optimal for many units but still outperforms a stationary model for the vast majority of units. This produces a smoothed estimate that may not follow all of the fluctuations in cluster location, but is still able to capture slower trends (see, e.g., Figure 3.1C). Despite this excessive smoothing, we still find that cluster drift accounts for 12–30% of the spike variability observed in longer recordings (Figure 3.4D).

3.3.3 Heavy-tailed residuals in empirical data

We also quantified the heavy-tailed distributions of the spike clusters. First, we note that these heavy tails are present even in the extracellular background noise when no

spikes are detected (Figure 3.5A). This is consistent with the data shown by previous spike sorting studies, including those that have considered the Gaussian distribution to be an adequate approximation (Fee, Mitra, and Kleinfeld, 1996; Pouzat et al., 2004; Prentice et al., 2011).

However, modeling the spike residuals as a Gaussian distribution dramatically underestimates the fraction of spikes that are located away from the cluster center (Figure 3.5B,C). Again, the observed distribution is more consistent with a t-distribution than a Gaussian.

In the MoDT model, the parameter ν is a user-defined constant that controls the heavy-tailedness of the assumed spike distribution. At $\nu = 1$, it corresponds to a Cauchy distribution, which has infinite variance. As $\nu \to \infty$, it approaches a Gaussian distribution. The Gaussian version of the MoDT model is equivalent to the "Mixture of Kalman filters" (Calabrese and Paninski, 2011).

We found that most units were best fit with ν in the range 3–20 (Figure 3.5D), with some differences between brain areas and cell types. For comparison, Shoham, Fellows, and Normann (2003) reported a range of 7–15 for single-electrode recordings in macaque motor cortex. We performed spike sorting using $\nu = 7$ as this provided a good approximation to both limits of the observed range.

3.3.4 Consequences of using a stationary model

Cluster drift is a well-known feature of chronic recordings, and many techniques have been proposed to address this phenomenon. A common approach is to break the recording into chunks, perform spike sorting on each chunk independently, and finally link the clusters across time (Bar-Hillel, Spiro, and Stark, 2006; Tolias et al., 2007; Wolf and Burdick, 2009; Shalchyan and Farina, 2014; Dhawale et al., 2017).

This approach comes with a tradeoff: short chunks may not contain enough spikes from low-firing neurons, but long chunks suffer more from the effects of drifting clusters. We characterized this tradeoff by breaking our recordings into chunks of varying duration, re-fitting each chunk with a stationary model, and analyzing the result. We identified three common failure modes of this approach (Figure 3.6): (A) fragmented units due to non-clusterable chunks, (B) loss of isolation between units, and (C) splitting of single units.

Unit fragmentation occurs when a unit cannot be linked across chunks. The proposed linking algorithms do not link units over a gap in activity, so a single non-clusterable chunk will break the chain of linked units. We evaluated this by counting how many

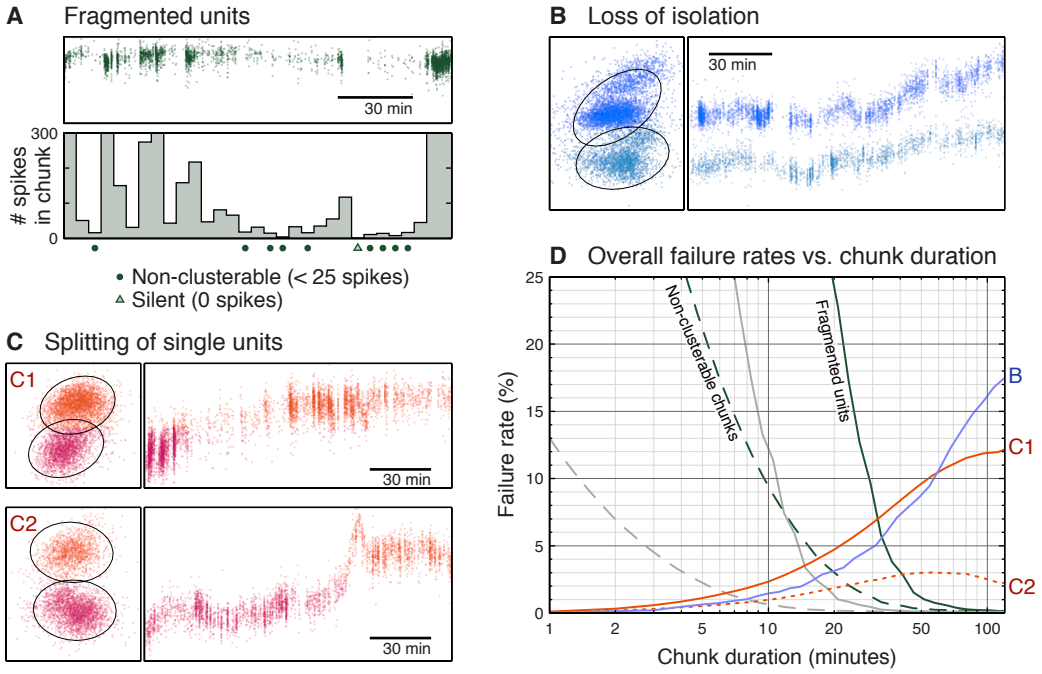

Figure 3.6: Failure modes of a stationary approach. An alternative approach to handling cluster drift is to break the recording into chunks, perform spike sorting on each chunk independently, and finally link the spike clusters over time. We identified three common failure modes of this approach. (**A**) In order to link a unit over time, each chunk must contain enough of that unit's spikes to form a cluster (we used a threshold of 25 spikes). If any chunk is non-clusterable, then the unit cannot be successfully linked and will become fragmented. Panel D shows the overall prevalence of these failures for varying chunk durations (dashed and solid green lines). The light grey lines (dashed and solid) repeat this analysis with the clusterability threshold set at 1 spike. (**B**) Drift causes clusters to become smeared out over time. If we analyze these irregularly-shaped clusters as stationary distributions, then some units will appear to overlap even though they remain well-isolated over time. This loss of isolation artificially reduces the yield of good units. (**C**) Drift may produce a multi-modal density distribution. As a result, the clustering algorithm may split a single unit into two clusters (C1). In some cases, these two clusters may be quite well-isolated from each other (C2). (**D**) Overall prevalence of these failure modes for varying chunk durations.

spikes a given unit fired within each chunk, and we considered any chunk with fewer than 25 spikes to be non-clusterable for that unit (Figure 3.6A). Figure 3.6D shows the overall fraction of non-clusterable chunks (dashed green line) and the fraction of units that are thus fragmented (solid green line).

Longer chunks are therefore needed to ensure that each chunk contains enough spikes to prevent unit fragmentation. However, longer chunks expose us to more cluster drift, which can cause a loss of isolation and splitting of single units.

Loss of isolation occurs when two drifting clusters occupy the same region of feature space at different times, which appears as cluster overlap under a stationary analysis (Figure 3.6B). Figure 3.6D (line B) shows the fraction of our well-isolated units would have failed to meet our quality threshold if we had instead used a stationary model to evaluate the unit isolation.

Cluster drift may also produce a multi-modal distribution that leads to a single unit being split into two clusters (Figure 3.6C). We quantified this effect by identifying cases where the Bayes information criterion (BIC) would justify splitting a cluster into two (Figure 3.6D, line C1). In some cases, the resulting clusters are well-isolated from one another (less than 5% overlap; Figure 3.6D, line C2) and would likely require timing information to identify them as a spuriously split unit.

These tradeoffs are faced by any approach, whether model-based or not, that performs spike sorting on each chunk independently. Although the MoDT model also uses discrete time frames, it avoids this tradeoff by aggregating data across frames: it uses the same cluster scale matrix $\boldsymbol{C}$ for all time frames and incorporates a drift regularizer that effectively smoothes the estimated cluster location $\boldsymbol{\mu}$ over time. As a result, it is able to track units regardless of how few spikes it may fire in a given time frame, which enables us to use sufficiently short time frames (1 minute) that the effects of drift are negligible.

Note that some other approaches also avoid this tradeoff. These also use models that allow for gradual parameter variation and disallow splitting or merging of clusters over time (Pouzat et al., 2004; Franke et al., 2010; Calabrese and Paninski, 2011; Carlson et al., 2013).

3.3.5 Implications for spike sorting of chronic recordings

Continuous recordings over the course of days or weeks can be a powerful tool for studying the long-term dynamics of neural firing properties (Harris et al., 2016).

This requires us to track single units over long periods of time, and model-based clustering using the MoDT model is well-suited for this task.

In particular, establishing when a neuron is silent is often just as important as knowing when it is firing. However, this involves demonstrating an absence of spikes in the region of feature space where we expected to see them, and this fundamentally requires a model-based approach.

This is why we did not consider the possibility of linking over non-clusterable chunks in our earlier analysis (section 3.3.4). Although the proposed linking algorithms could be modified to link over a non-clusterable chunk, doing so poses a problem when using the sorted spike trains to draw conclusions about neural activity. Linking a unit over a non-clusterable chunk would imply that it was silent during this period. However, one's inability to cluster a unit in a given chunk does not certify that it was silent; it could have fired insufficient spikes to warrant its own cluster or it could have been spuriously merged into another cluster.

In contrast, the MoDT model effectively interpolates the cluster's expected location between consecutive "sightings" of the unit, giving us a reasonable guarantee that the lack of spikes assigned to this unit in the intervening period is indeed due to its silence. Although linking is still necessary for continuous recordings, the MoDT model simplifies the linking process by allowing us to perform spike sorting in segments up to 10 hours in duration. The use of longer segments ensures that all units will fire enough spikes to be clustered, reduces the number of segments that need to be linked, and enables the use of overlapping segments. For example, a week-long recording can be broken into 21 ten-hour segments with an overlap of 2 hours each. We can then establish cluster correspondences based on the spike assignments of the overlapping data.

3.4 Use of the MoDT model for measuring unit isolation

Consider the following spike sorting algorithm: each spike is randomly assigned to an arbitrary cluster until there are no more spikes left. This algorithm is completely unsupervised, computationally efficient, and amenable to parallelization. It is also very bad at spike sorting. But how do we know?

A common validation step is to run the algorithm on some labeled ground truth data, i.e., extracellular recordings in which one of the neurons has been simultaneously recorded using an intra- or juxta-cellular pipette that is able to isolate the action potentials of a single neuron. We can then evaluate the spike sorting algorithm based on the correspondence between the known spike train and the spike sorted output.

However, these ground truth datasets are quite rare—such paired recordings are experimentally challenging to perform—and may not be representative of the experimental conditions that you wish to record from. Differences in electrode type and preparation will affect the spectral power density of the recorded signal, differences in brain area will change the statistics of the background noise and multi-unit activity, and differences in the specific cell types in the vicinity of the electrode will affect the separability and ease of clustering. How can we tell if our algorithm's output can still be trusted under changing experimental conditions?

Answering this question requires reliable, quantitative measures of unit quality[2]. Despite the proliferation of spike sorting methods, considerably less attention has been paid to this problem. I will review some of these quality metrics (section 3.4.1), discuss how the MoDT model may be used in this context (section 3.4.2), explain the "hybrid ground truth" approach to producing labeled datasets (section 3.4.3), and use this to compare the performance of these unit quality metrics (section 3.4.4). Finally, I will discuss some of the additional considerations involved in applying these metrics to chronic data (section 3.4.5).

3.4.1 Background: unit isolation metrics

Regardless of the algorithm used, the output of spike sorting will be spike trains that have been grouped into clusters. However, not all of these clusters will correspond to single units (i.e., the spiking activity of individual neurons). Spike detection and sorting are not perfect, and a given cluster may contain spurious spikes (false positives) or may not capture all of the spikes from a given neuron (false negatives).

[2]The spike sorted output (groups of spikes that supposedly correspond to individual neurons) are also known as "single units" and so "unit quality" refers to these. This is often used synonymously with "unit isolation", since good isolation between clusters is a major component of overall quality.

Depending on the scientific question being addressed, our subsequent analysis may be more or less sensitive to the presence of such errors. Reliable, quantitative measures of unit quality are therefore critically important for the proper interpretation of the spike sorting output.

Schmitzer-Torbert et al. (2005) introduced two such metrics of unit quality—isolation distance and L-ratio—which have since found widespread use. These are estimates of how well individual clusters are isolated from the rest, and are based on the Mahalanobis distance δ_{nk} from cluster k to spike n. If there are N_k spikes in cluster k, then its isolation distance is the N_kth smallest value of δ_{nk}^2 among the spikes not assigned to that cluster. L-ratio is defined as L/N_k, where L is the sum, over all spikes n not assigned to cluster k, of the complementary CDF of a χ_D^2 distribution evaluated at δ_{nk}^2. This summand can be interpreted as the P-value, using the Mahalanobis distance as the test statistic, under the null hypothesis that the given spike came from a Gaussian distribution fitted to the spikes assigned to cluster k.

Hill, Mehta, and Kleinfeld (2011) proposed a more direct quantification of the number of expected false positives and negatives. By fitting a generative model, such as a mixture of Gaussians, we can estimate the false positives and negatives due to misclassification errors: spikes assigned to a cluster that should have been assigned to a different cluster[3]. We will be using FP+FN as a shorthand for misclassification error.

K-means consensus (Fournier et al., 2016) is a non-model-based approach in which K-means is used to partition the data based on their Euclidean distance in feature space. This is repeated multiple times from random initializations, and the estimated misclassification error is computed from the fraction of a given cluster's spikes that have been co-partitioned with other clusters' spikes.

Finally, we propose using a t-distribution to quantify the misclassification error. We have previously shown that the t-distribution is a good fit for the tails of the spike distribution (section 3.3.3), which suggests that it may be a good candidate for estimating the overlap between clusters. We will therefore be comparing these five metrics—isolation distance, L-ratio, FP+FN from a Gaussian distribution,

[3]Hill, Mehta, and Kleinfeld (2011) also describe steps for visual inspection and introduce estimates of false negatives due to other sources, such as failure of spike detection and censoring from overlapping spikes. These are very important steps and are a critical component in determining which clusters correspond to putative single units. However, we will not discuss these further since they are not affected by our choice of model.

FP+FN from K-means consensus, and FP+FN from a t-distribution—using hybrid ground-truth data.

3.4.2 Measuring unit isolation using the MoDT model

The MoDT model may be fitted to previously spike-sorted data by using the given spike assignments, rather than the z_{nk} computed in the E-step, during the M-step update. If the spike sorting algorithm can provide soft assignments (i.e., each spike has a probability of belonging to each cluster rather than being fully assigned to a single cluster), then these probabilities can be substituted directly as $\hat{z}_{nk}$. For hard assignments, the equivalent posterior is

$$\hat{z}_{nk} = \begin{cases} 1 & \text{if spike } n \text{ was assigned to cluster } k \\ 0 & \text{otherwise.} \end{cases}$$

Model fitting still requires iterative evaluation of equations (3.3)–(3.6), but typically converges in fewer than 10 iterations since $\hat{z}_{nk}$ is fixed.

After fitting the model parameters (α_k, $\boldsymbol{\mu}_{kt}$, $\boldsymbol{C}_k$), the z_{nk} defined by equation (3.2) provides a model-based estimate of the probability that spike $\boldsymbol{y}_n$ was produced by each of the source clusters. Summing these z_{nk} provides the expected number of misclassified spikes. Following Hill, Mehta, and Kleinfeld (2011), we define the false positive (FP) fraction and the false negative (FN) ratio for cluster k as

$$\text{FP\%} = \frac{1}{|\mathcal{N}_k|} \sum_{n \in \mathcal{N}_k} \sum_{\kappa \neq k} z_{n\kappa}$$

$$\text{FN\%} = \frac{1}{|\mathcal{N}_k|} \sum_{n \notin \mathcal{N}_k} z_{nk},$$

where $\mathcal{N}_k$ is the set of spikes assigned to cluster k. In the hypothesis testing literature, the FP fraction is also known as the false discovery rate. The FN ratio does not have a similar analogue, and it may be greater than one.

The MoDT model also provides a natural generalization of Gaussian-based unit isolation metrics to the drifting case. By setting $\nu = \infty$, the fitted $\boldsymbol{\mu}_{kt}$ and $\boldsymbol{C}_k$ correspond to the time-varying cluster mean and the cluster covariance, respectively[4]. These were used to compute Mahalanobis distances for the isolation distance and L-ratio metrics in Figure 3.8.

[4] Alternatively, ν could be set to a finite value and the procedure described in section 3.2.4 used to derive robust estimates of the Gaussian mean and covariance.

3.4.3 Hybrid ground truth datasets

To validate the performance of our spike sorting toolchain, we generated "hybrid ground truth" datasets by injecting known spikes into an acceptor dataset (Rossant et al., 2016). In order to more realistically capture the waveform variability, we used the actual spike waveforms from the original data rather than synthesizing them from the mean waveform, but otherwise followed the procedure described by Rossant et al.

We selected 45 well-isolated units to form our base set of donor units. However, this selection is unavoidably biased towards units that are easy to cluster using our current method. In order to more fully characterize the space of possible units, we generated additional units by modifying the spike amplitude (by scaling the spike waveforms), firing rate (by dropping a subset of spikes), and drift rate (by temporally compressing the spike train) of these base units.

We thus obtained 450 donor units that ranged in spike peak amplitude from 50–700 μV, firing rate from 0.003–30 Hz, and drift rate from 0.06–5000 μV^2/hr. For each donor, we selected an acceptor dataset from the same tetrode but several days earlier or later. We then spike-sorted each of these 450 4-hour datasets, identified the cluster that best corresponds to the injected spike set, and measured the number of false positives and negatives. Unsurprisingly, the results varied widely and depended on the amplitude and firing rate of the injected units (Figure 3.7A).

While the error rate is undeniably correlated to attributes such as the spike amplitude, firing rate, and drift rate of the injected units, these properties are a poor predictor of spike sorting performance on a case-by-case basis (Figure 3.7B).

These results underscore the difficulty of making general claims about spike sorting performance. Given the variety of experimental conditions—brain area, cell type, probe geometry, electrode impedance, presence of artifacts, etc.—it is difficult if not impossible to guarantee that a particular spike sorting algorithm or parameter set will achieve a given performance specification in all circumstances.

Instead, performance must be evaluated on a case-by-case basis, and in the absence of ground truth, we must rely on quantitative estimates of unit quality. Indeed, our proposed measure of misclassification error accurately estimates the true error on these hybrid ground truth datasets (Figure 3.7C).

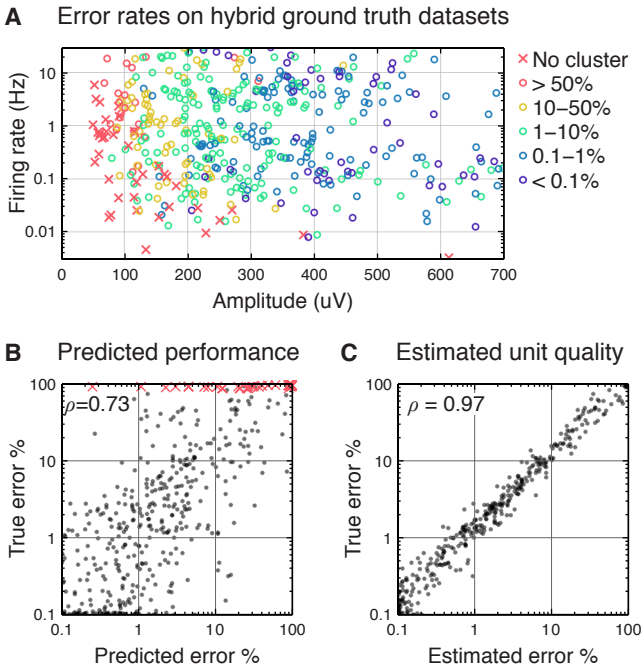

Figure 3.7: Spike sorting performance. (**A**) Total error rates (false positives + false negatives) on 450 hybrid ground truth datasets. The location of each dot indicates the amplitude and firing rate of the injected unit, and its color indicates the total error rate for that unit. Red X's indicate cases where no single cluster corresponds to the injected unit. (**B**) Although performance is correlated to many attributes of the injected unit, it is difficult to predict the spike sorting performance based on these attributes alone. ρ is Spearman's rank correlation. (**C**) The misclassification error (section 3.4.2), combined with estimates of false negatives due to failed spike detection and censoring from overlapping spikes (Hill, Mehta, and Kleinfeld, 2011), provides an accurate estimate of the true error rate.

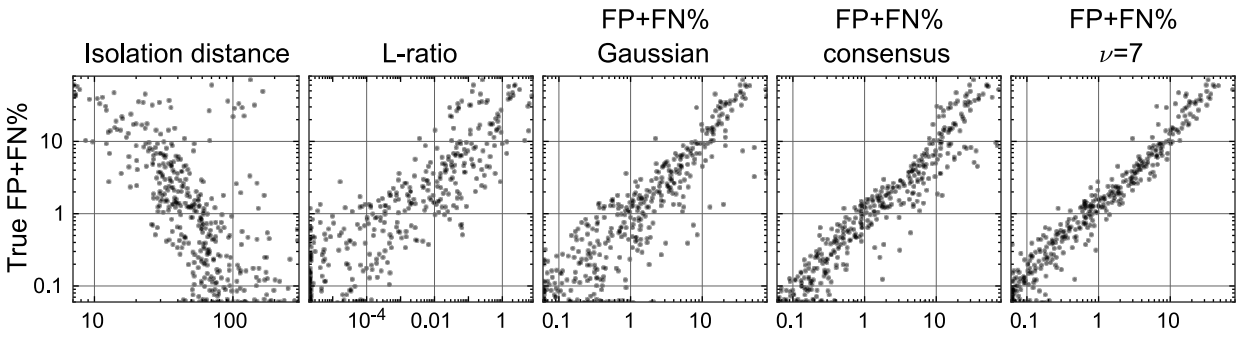

Figure 3.8: Comparison of unit quality metrics on hybrid ground truth datasets. Using 450 hybrid ground truth datasets, we evaluated how well five unit isolation metrics—isolation distance, L-ratio, total misclassification error (FP+FN) based on a Gaussian model, FP+FN based on K-means consensus, and FP+FN based on a $\nu = 7$ t-distribution model—were able to estimate the true misclassification error. These metrics are defined in section 3.4.1.

3.4.4 Comparative analysis of unit isolation metrics

Using these 450 hybrid ground truth datasets, we evaluated how well these five unit isolation metrics—isolation distance, L-ratio, FP+FN based on a Gaussian model, FP+FN using K-means consensus, and FP+FN using a $\nu = 7$ t-distribution model—were able to estimate the true misclassification error (Figure 3.8). Note that we are considering only misclassification errors and not false negatives due to spike detection or censoring.

This analysis shows that isolation distance is a poor metric of unit isolation. Reviewing its definition reveals an important flaw: the contaminating cluster is completely ignored if it contains fewer spikes than the cluster being measured. In such cases, the isolation distance is determined by the location of the second-nearest cluster, and may be arbitrarily large. As a result, a large isolation distance does not imply a low misclassification error, particularly for units with many spikes. Even under ideal conditions, the relationship between the isolation distance and the error rate depends heavily on the dimensionality of the feature space, making it difficult to compare quality thresholds across experimental settings.

The L-ratio shares many of these same shortcomings: the correspondence between L-ratio and error rate is pretty weak, and this relationship is also highly dependent on the dimensionality of the feature space.

The remaining metrics—estimated misclassification error based on a Gaussian model, K-means consensus, or a $\nu = 7$ t-distribution model—are easier to interpret and offer better performance. Of these, the t-distribution model provided the most accurate estimates. Additional analysis indicates that it is robust to variations in the underlying distribution of the clusters being measured (Shan, Lubenov, and Siapas, 2017, Figure C.1) and is able to provide accurate estimates of the false positives and negatives separately (Shan, Lubenov, and Siapas, 2017, Figure C.2).

3.4.5 Measuring unit isolation quality in chronic recordings

The analysis of long recordings also requires unit quality metrics that can handle drift. The MoDT model accomplishes this by explicitly tracking the clusters over time. The use of a t-distribution also provides a natural robustness to outliers (Figure 3.3) and produces accurate estimates of misclassification error over a wide range of conditions (Figure 3.8).

However, the MoDT model is still a highly structured model. Each cluster is elliptically symmetric with a predetermined tail distribution, and the drift regularization

discourages sudden changes in the cluster's location. It is only through slow drift over time that we can trace out an irregularly-shaped cluster in feature space (e.g., Figure 3.6B). In contrast, non-parametric approaches allow clusters to take on arbitrary shapes, which may require additional review to ensure that they correspond to biophysically plausible spike distributions.

Furthermore, it is important to acknowledge that unit isolation quality is a time-varying quantity. Drift may cause two clusters to be well-separated at one point in time, but begin to overlap later. If subsequent analyses are restricted to a particular subset of the overall recording, then the unit isolation measures should be based on those epochs as well. Model-based approaches accommodate this requirement by providing a continuous estimate of misclassification error, which may then be integrated over the appropriate epochs.

Finally, we would like to caution that isolation quality is only one aspect of unit quality overall. Hill, Mehta, and Kleinfeld (2011) describe a number of additional quality measures. For example, estimating false negatives due to spike detection is an equally important yet frequently overlooked metric. This is especially important in the presence of cluster drift, as fluctuations in spike amplitude may affect detection efficiency, which could manifest as apparent changes in firing rate.

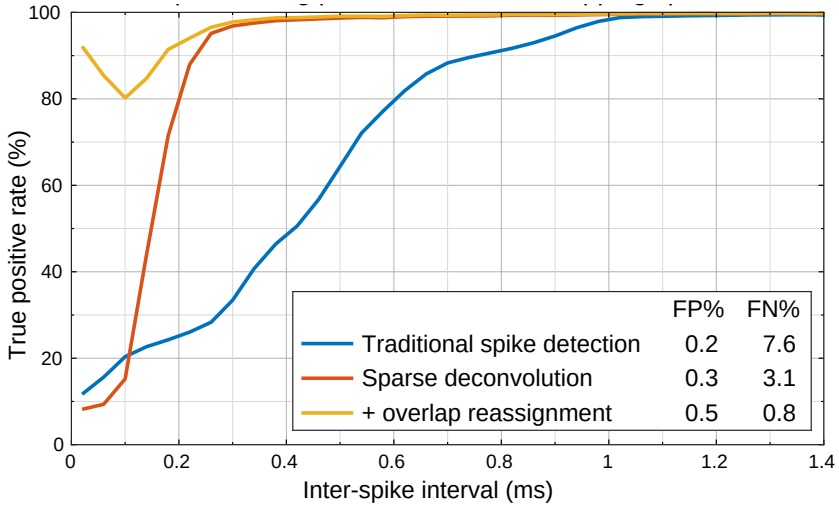

Figure 3.9: Spike sorting performance on overlapping spikes. Fraction of hybrid ground truth[5] donor spikes that were detected and assigned to the correct source (true positive rate) as a function of the time between spikes (inter-spike interval). The overall false discovery rate (FP%) and false negative rate (FN%) for each method are reported in the legend. Traditional spike detection uses the nonlinear energy operator for spike detection and principal components analysis (PCA) for feature extraction. Sparse deconvolution is the spike detection approach described in Chapter 2. Overlap reassignment is the described in section 3.5.1.

3.5 Overlapping spikes

If two neurons fire near-simultaneously, their spikes will overlap and produce a waveform that is the sum of both waveforms. We discussed this earlier in section 2.3 in the context of spike detection. There, we showed that our spike detection method could reliably deconvolve spikes that are separated by as little as 0.3 ms. Overall, it offers a substantial improvement over traditional spike detection without sacrificing performance on non-overlapping spikes (Figure 3.9).

However, this method still cannot deconvolve very closely-overlappig spikes; it simply is not capable of distinguishing between the sum of two simultaneous spikes and a single large spike. These then show up as outliers during the clustering process. Fortunately, thanks to the t-distribution's robustness to outliers, these outliers do not substantially affect the model fitting (Figure 3.3) and can be ignored during the interactive clustering process.

[5] The hybrid ground truth dataset analyzed in figures 2.4 and 3.9 was created slightly differently from the datasets described in section 3.4.3. Instead of choosing a donor unit and acceptor dataset from the same tetrode on different days, I selected 12 simultaneously-recorded donor units (from different tetrodes) in order to preserve the temporal structure of the spikes. The average firing rates of the donor units ranged from 0.15 to 26.4 Hz, with a combined firing rate of 76.4 Hz. The acceptor dataset was relatively quiet, with a spike detection rate of 1.5 Hz.

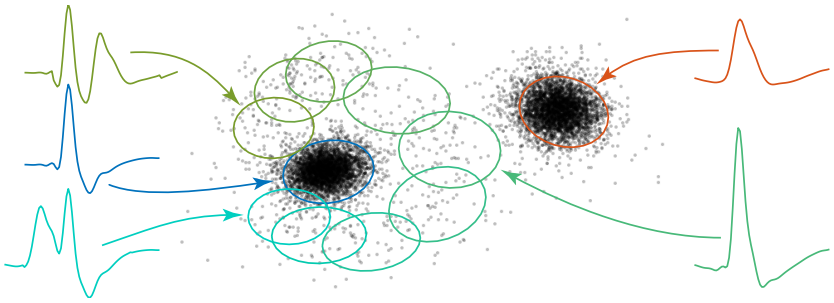

Figure 3.10: Model-based overlap reassignment. Feature space scatterplot containing some instances of overlapping spikes. Blue and red ellipses show the model clusters for single spikes; overlapping spikes appear as outliers. By augmenting the model with overlap clusters (green ellipses), we can reassign these overlapping spikes to the appropriate units.

Afterwards, a number of options are available. If the estimated number of false negatives due to spike overlap (Hill, Mehta, and Kleinfeld, 2011) is sufficiently low, then it may be adequate to leave the spike assignments as-is. Unless otherwise specified, this is the method used for the analyses in section 3.3.

Alternatively, we can augment the fitted model with additional components corresponding to these overlapped waveforms (Figure 3.10). This is described in more detail in the next section. Any spikes assigned to these overlap clusters are then reassigned to their source units. In our test dataset, this correctly reassigned more than 80% of overlapping spikes (Figure 3.9), yielding an overall false negative rate of less than 1%.

A third option is to transform the fitted cluster centers $\boldsymbol{\mu}_{kt}$ back into waveform space for use in a template matching algorithm. This approach is described in more detail in (Shan, Lubenov, and Siapas, 2017).

3.5.1 Model-based overlap reassignment

In this section, we describe how we augmented the model with overlap clusters. To generate the overlap cluster corresponding to units k_1 and k_2 firing with a particular temporal offset, we first need to determine their temporal offsets τ_1, τ_2 relative to the detected spike time. To do this, we construct an overlap waveform by overlapping their individual mean waveforms $\boldsymbol{W}\boldsymbol{\mu}_{k_1}$ and $\boldsymbol{W}\boldsymbol{\mu}_{k_2}$, where $\boldsymbol{W}$ is the $P \times D$ matrix corresponding to the D basis waveforms. We then pass this through the spike detection algorithm to determine the center of the detected spike. For example, all of the overlap waveforms in Figure 3.10 ended up aligned to the blue spike. Note that this is just an approximation; it doesn't take into account the cluster drift over time or the waveform variability, but it is easy to implement.

This gives us the transformation matrices

$$\boldsymbol{U}_1 = \boldsymbol{W}^\intercal \boldsymbol{T}_{\tau_1} \boldsymbol{W}$$
$$\boldsymbol{U}_2 = \boldsymbol{W}^\intercal \boldsymbol{T}_{\tau_2} \boldsymbol{W},$$

where $\boldsymbol{T}_\tau$ is the $P \times P$ matrix corresponding to a temporal shift by τ samples. These can be used to derive the model parameters of the overlap cluster:

$$\boldsymbol{\mu} = \boldsymbol{U}_1 \boldsymbol{\mu}_{k_1} + \boldsymbol{U}_2 \boldsymbol{\mu}_{k_2}$$
$$\boldsymbol{C} = \boldsymbol{U}_1 \boldsymbol{C}_{k_1} \boldsymbol{U}_1^\intercal + \boldsymbol{U}_2 \boldsymbol{C}_{k_2} \boldsymbol{U}_2^\intercal$$
$$\alpha = \alpha_{k_1} \alpha_{k_2} \beta.$$

β is the probability of any two units firing with the given temporal offset, and depends on the overall spike detection rate.

However, the number of possible cluster combinations is quite large (on the order of K^2) and is compounded by the number of temporal offsets that we need to consider (we used 11 offsets from −0.2 to +0.2 ms). We pruned the number of overlap clusters by evaluating the posterior probability (z_{nk}) that a spike located at the overlap cluster's center belongs to that cluster vs. one of the base clusters, and keeping only the overlap clusters with the largest posterior. On our benchmarking computer (section 3.2.3), it took 2.9 seconds to generate and prune the overlap clusters (from a base model with $K = 17$) and 68.8 seconds to process the test dataset (4.2 million spikes) with 750 overlap clusters.

This method works best when the overlapping spikes occur simultaneously (inter-spike interval = 0 in Figure 3.9), because that provides the best conditions for feature extraction. When the spikes are slightly offset, the contribution of the second spike may be nearly orthogonal to the waveform bases, resulting in poor discrimination.

Chapter 4

IMPROVING REGULARIZERS FOR SPARSE DECONVOLUTION

In Chapter 2, we set up sparse deconvolution as the regularized least-squares problem

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{A}x - b\|^2 + \beta g(x), \tag{4.1}$$

where x is the latent feature vector being optimized, $\mathcal{A}$ is a linear operator representing convolution with a known set of kernels, b is the given data vector we wish to approximate, and β is a tuning parameter that controls the relative importance of the sparsity-inducing regularizer $g(x)$.

The simplest and most popular sparsity-inducing regularizer is the l_1 norm, the sum of the absolute values of the vector elements. Also known as the lasso penalty, this regularizer was originally developed for factor selection in regression problems (Tibshirani, 1996), but has become extremely popular for sparse approximation problems such as compressive sensing (Donoho, Elad, and Temlyakov, 2006; Candes, 2008) and deconvolution (Taylor, Banks, and McCoy, 1979; Chen, Donoho, and Saunders, 2001), including for neuroscience applications such as deconvolution of calcium imaging data (Vogelstein et al., 2010; Pnevmatikakis et al., 2016).

However, the l_1 norm does not always produce the desired results, as we saw in section 2.2.4, so in this chapter I will discuss two modifications that were necessary for effective deconvolution, and were not adequately addressed by the prior work.

Section 4.1 discusses the nonconvex log regularizer, which produces sparser solutions with less approximation error. However, efficient optimization using this regularizer requires a closed-form expression for its proximal operator, and the prior work did not provide an accurate expression for this operator. So in this section I will derive the correct expression for the log regularizer's proximal operator.

Section 4.2 discusses regularizers that are organized into nested groups of features. These nested groups allow us to specify the desired sparsity structure for a deconvolution solution. Under some conditions, the proximal operator of a nested group regularizer corresponds to the composition of elementary proximal operators, and I will show that this is true for a more general class of regularizers than previously reported, including the nonconvex log regularizer discussed in section 4.1.

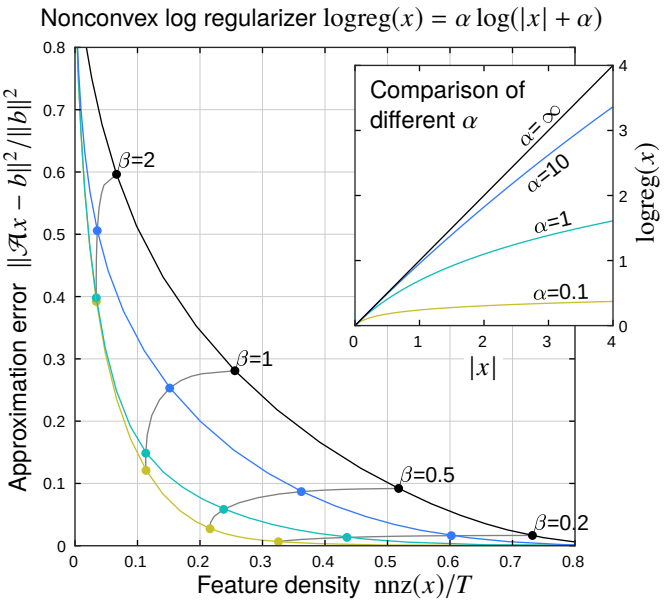

Figure 4.1: Application of the nonconvex log regularizer to a sparse deconvolution problem. The log regularizer $\mathrm{logreg}(x) = \alpha \log(|x| + \alpha)$ is a nonconvex sparsity-inducing regularizer. **Inset:** This shows the shape of $\mathrm{logreg}(x)$ for different values of α. Note that the slope is 1 at $x = 0$ and decreases for larger x. In particular, the slope is $\frac{1}{2}$ at $x = \alpha$. As $\alpha \to \infty$, $\mathrm{logreg}(x) \to |x|$. **Main panel:** This shows how the choice of regularizer affects the approximation error and sparsity of the solution to a sparse deconvolution problem (see text). Let us first consider the $\alpha = \infty$ case (black line), which is equivalent to regularization using the convex l_1 norm. As we vary β, the parameter that weights the regularizer in the overall objective, we move along some frontier that trades off sparsity for reduced approximation error. By reducing α, however, we arrive at solutions that are both sparser *and* have less approximation error.

4.1 The nonconvex log regularizer

In this section, we will investigate the use of a nonconvex log regularizer as the sparsity-inducing regularizer $g(x)$ in the sparse deconvolution problem

$$\underset{x}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{A}x - b\|^2 + \beta g(x). \tag{4.2}$$

Although other authors have used a slightly different notation, I will define the log regularizer as

$$\mathrm{logreg}(x) \triangleq \alpha \log(|x| + \alpha). \tag{4.3}$$

Including the factor of α in front makes the derivative $\mathrm{logreg}'(0_+) = 1$ for all α (Figure 4.1, inset), and gives us $\mathrm{logreg}(x) \to |x|$ as $\alpha \to \infty$. The overall regularizer in (4.2) is simply the sum over each element of the feature vector

$$g(\boldsymbol{x}) = \sum_i \mathrm{logreg}(x_i). \tag{4.4}$$

This regularizer was analyzed by Candes, Wakin, and Boyd (2008), who demonstrated that it may produce sparser solutions with less approximation error. I have observed this same improvement in my own sparse deconvolution problems[1] (Figure 4.1), and can confirm that the log regularizer produces much better solutions for the spike detection problem (Figure 2.2 in Section 2.2.4). Since the slope of this regularizer flattens out for larger x (Figure 4.1 inset), it does not bias solutions towards zero as much, and the concavity of this regularizer discourages single events from being decomposed into the sum of two correlated events.

Defining a regularizer is all well and good, but how do we go about optimizing it? Candes, Wakin, and Boyd (2008) used reweighted l_1 minimization, which entails wrapping an outer loop around a convex l_1 minimization. In this section, we will instead derive the proximal operator for this regularizer, which enables us to minimize it directly. Section 4.1.1 provides some background on proximal operators and reviews some prior work on this regularizer. Section 4.1.2 then derives the proximal operator for this regularizer, which is then restated more concisely in Section 4.1.3.

4.1.1 Background: proximal operators and prior work

The proximal operator of a cost function $g(x)$ is defined as

$$\text{prox}_g(y) \triangleq \arg\min_x \; g(x) + \frac{1}{2}\|x - y\|^2. \tag{4.5}$$

In essence, the proximal operator seeks to minimize $g(x)$ while also staying close to the input point y. Algorithms that rely on this proximal operator (Parikh and Boyd, 2014) have become the *de facto* standard for large-scale nonsmooth optimization, making the existence of a simple proximal operator a necessity for practical implementation.

For example, in FISTA (Beck and Teboulle, 2009), the accelerated proximal gradient descent algorithm that we used in Chapter 2, each iteration k of the algorithm requires performing the minimization

$$\begin{aligned} x^{(k+1)} &= \arg\min_x \; f(y^{(k)}) + \langle \nabla_f(y^{(k)}), x - y^{(k)} \rangle + \beta g(x) + \frac{L^{(k)}}{2}\|x - y^{(k)}\|^2 \\ &= \arg\min_x \; \frac{\beta}{L^{(k)}} g(x) + \frac{1}{2}\left\|x - \left(y^{(k)} - \frac{1}{L^{(k)}}\nabla_f(y^{(k)})\right)\right\|^2. \end{aligned}$$

[1] Specifically, this problem analyzed in Figure 4.1 is a form of sparse time-frequency analysis. The kernels are a set of Gaussian-windowed complex sinusoids spanning many different frequencies and window lengths with a high degree of redundancy. This is somewhat analogous to a continuous wavelet transform using Morlet/Gabor wavelets, except that we are relying on the sparse deconvolution to select the best wavelet σ parameter on a case-by-case basis.

Since our choice of $g(x)$ is the sum of logreg over the elements x_i, and the squared norm $\|x - y\|^2$ is likewise separable over x_i, we can find $x^{(k+1)}$ by performing the minimization for each element independently:

$$x_i^{(k+1)} = \arg\min_{x_i} \; \tilde{\beta}\,\mathrm{logreg}(x_i) + \frac{1}{2}(x_i - \tilde{y}_i)^2,$$

where $\tilde{\beta} = \beta / L^{(k)}$ and $\tilde{y} = y^{(k)} - \frac{1}{L^{(k)}} \nabla_f(y^{(k)})$. In other words, we need the proximal operator of

$$h(x) = \tilde{\beta}\,\mathrm{logreg}(x) = \tilde{\beta}\alpha \log(x + \alpha). \tag{4.6}$$

As we will show in the next section, the cost function (4.6) admits a simple proximal operator. This is discussed by Malioutov and Aravkin (2014), but they only provided an expression for a local minimum and not necessarily the global minimum. They rationalize this by arguing that the reweighted minimization of Candes, Wakin, and Boyd (2008) also would have converged to this local minimum.

More concerning, however, is the fact that their expression is incorrect for certain regions of parameter space (specifically, when $\alpha > \tilde{\beta}$). The authors make no mention of this[2], and while it may not have been an issue for their choice of optimization algorithm, which encounters only fixed values of α and $\tilde{\beta}$, our use of backtracking (which adjusts $L^{(k)}$ and therefore $\tilde{\beta}$ on every iteration) and an annealing schedule for α (which gradually reduces α from ∞ over the course of multiple iterations) requires that we have an expression for prox_h that is correct for all values of α and $\tilde{\beta}$.

4.1.2 Deriving the proximal operator for the log regularizer

In this section, we will derive the proximal operator for

$$h(x) \triangleq \beta\alpha \log(|x| + \alpha). \tag{4.7}$$

That is, if we introduce the proximal objective function (Figure 4.2, left)

$$J(x; y) \triangleq h(x) + \frac{1}{2}(x - y)^2, \tag{4.8}$$

then we wish to find an expression for the proximal operator (Figure 4.2, right)

$$\mathrm{prox}_h(y) \triangleq \arg\min_x J(x; y). \tag{4.9}$$

[2]This incorrect expression for prox_h was also repeated by Bayram (2015) and Li, Ding, and Li (2015), and these authors also failed to pass along Malioutov and Aravkin's warning that the expression is only a local minimum.

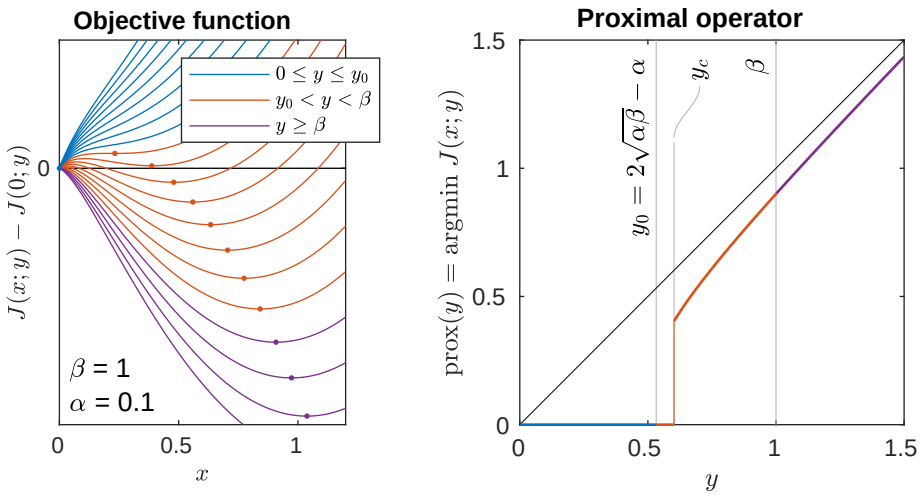

Figure 4.2: An illustration of the proximal operator of the log regularizer. This figure illustrates how we derive the proximal operator of the log regularizer $h(x) = \beta\alpha \log(|x| + \alpha)$. In this example, $\alpha = 0.1$ and $\beta = 1$. **Left:** The proximal objective function $J(x; y)$ for different values of y. Local minima are marked with dots. **Right:** $\text{prox}_h(y)$, i.e. the value of x that minimizes $J(x; y)$ for each value of y. Both panels are color-coded to show the different regimes of this objective function. For small y, $J(x; y)$ has no stationary points and our only minimum is the boundary point $x = 0$ (blue). As y increases, we gain a stationary point at some $\xi > 0$, giving us two distinct local minima (red): one at the boundary $x = 0$ and one at the stationary point $x = \xi$. At some critical value y_c, the global minimum switches from 0 to ξ. For large y, $x = 0$ is no longer a local minimum, leaving only the nonzero minimum (purple).

We will make the following assumptions regarding α and β:

- $\alpha > 0$. If $\alpha = 0$, then $h(x) = 0$ for all x.
- $\beta > 0$. If $\beta = 0$, then $h(x) = 0$ for all x.

In the remainder of this section, we will derive an expression for prox_h by establishing the following:

- If $y < 0$, then $\text{prox}_h(y) = -\text{prox}_h(-y)$.
- If $0 \le y \le y_0$, where $y_0 = 2\sqrt{\alpha\beta} - \alpha$, then $\text{prox}_h(y) = 0$.
- If $y_0 < y \le \beta$, then:
  - If $\alpha \ge \beta$, then $\text{prox}_h(y) = 0$.
  - If $\alpha < \beta$, then let y_c be the solution to $\phi(y_c) = 0$, defined in (4.18), and:
    - If $y \le y_c$, then $\text{prox}_h(y) = 0$.
    - If $y > y_c$, then $\text{prox}_h(y) = \xi_2(y)$, where ξ_2 is defined in (4.16).
- If $y > \beta$, then $\text{prox}_h(y) = \xi_2(y)$.

For the first point (regarding $y < 0$), note that $h(-x) = h(x)$ implies that $J(-x; -y) = J(x; y)$ and hence $\operatorname{prox}_h(-y) = -\operatorname{prox}_h(y)$. Therefore, without loss of generality, we will assume that $y \geq 0$ in the remainder of this section.

This allows us to restrict our search to $x \geq 0$. We can show this by contradiction: if $x < 0$, then $h(x) > h(0)$ since h is uniquely minimized at zero and $(x-y)^2 > (0-y)^2$ since $y \geq 0$, therefore $J(x; y) > J(0; y)$, indicating that this $x < 0$ cannot be a minimizer of J.

Now let's get an expression for the stationary points of J, i.e. the points ξ where the derivative $J'(\xi; y) = 0$. Since we are only considering $x \geq 0$, the derivative of J with respect to x is given by

$$J'(x; y) = \frac{\alpha\beta}{x + \alpha} + x - y. \tag{4.10}$$

A little bit of algebraic manipulation yields

$$J'(x; y) = \frac{1}{x + \alpha}\left(x^2 + (\alpha - y)x + \alpha(\beta - y)\right) \tag{4.11}$$

$$= \frac{1}{x + \alpha}\left(\left(x + \frac{\alpha - y}{2}\right)^2 - \frac{d}{4}\right), \tag{4.12}$$

where

$$d \triangleq (\alpha - y)^2 - 4\alpha(\beta - y). \tag{4.13}$$

If $d < 0$, then (4.12) shows that $J'(x; y) > 0$ for all $x \geq 0$, and $x = 0$ must be the unique global minimum. If $d = 0$, then we have an inflection point at $x = \frac{1}{2}(y - \alpha)$, but $J'(x; y) > 0$ everywhere else, so $x = 0$ is still the unique global minimum.

If $d \geq 0$, then we can factorize (4.12) further into

$$J'(x; y) = \frac{(x - \xi_1)(x - \xi_2)}{x + \alpha}, \tag{4.14}$$

where

$$\xi_1 \triangleq \frac{1}{2}(y - \alpha - \sqrt{d}) \tag{4.15}$$

$$\xi_2 \triangleq \frac{1}{2}(y - \alpha + \sqrt{d}). \tag{4.16}$$

Inspecting the sign of (4.14) shows that if $\xi_1 \geq 0$, then it is a local maximum, and if $\xi_2 \geq 0$, then it is a local minimum.

To summarize what we've shown so far:

- If $d \leq 0$, then $x = 0$ is the global minimum.
- If $d > 0$ and $\xi_2 \geq 0$, then ξ_2 is a local minimum.

For the next part, it will help to identify a few regimes of our problem (the color-coded regions in Figure 4.2). If we let

$$y_0 \triangleq 2\sqrt{\alpha\beta} - \alpha, \tag{4.17}$$

then note that $y_0 \leq \beta$ (since $\beta - y_0 = (\sqrt{\beta} - \sqrt{\alpha})^2 \geq 0$), which allows us to partition the real line into 3 regimes:

1. $y \leq y_0$.
   Rearranging our definition of d (4.13), we find that $d = (y + \alpha)^2 - 4\alpha\beta$, so if $y + \alpha \leq y_0 + \alpha = 2\sqrt{\alpha\beta}$, then $d \leq 0$ and hence $x = 0$ is the global minimum.
2. $y_0 < y \leq \beta$.
   Using a similar argument as above, we can note that $d > 0$, so ξ_2 is real and could potentially be a local minimum. Let's save further discussion for later.
3. $y > \beta$.
   We've already established that $d > 0$ since $y > \beta \geq y_0$. Substituting the fact that $\beta - y < 0$ into our definition of d (4.13), we find that $d > (\alpha - y)^2$ and hence $\sqrt{d} > |y - \alpha|$. Substituting this into our definition of ξ_2 (4.16) shows that $\xi_2 > \frac{1}{2}(y - \alpha + |y - \alpha|) \geq 0$, and hence ξ_2 is a local minimum.
   Furthermore, the derivative (4.10) gives us $J'(0; y) = \beta - y < 0$, which means that $x = 0$ is a local maximum and therefore ξ_2 must be the global minimum.

We will need to further subdivide the middle regime ($y_0 < y \leq \beta$) into two cases: $\alpha \geq \beta$ and $\alpha < \beta$.

If $\alpha \geq \beta$, then we have $y \leq \beta \leq \alpha$, which means that both $\alpha - y \geq 0$ and $\beta - y \geq 0$. Substituting this into (4.11) shows that

$$J'(x; y) \geq \frac{x^2}{x + \alpha}$$

and hence $J'(x; y) > 0$ for all $x > 0$, implying that $x = 0$ is the global minimum.

This last case ($y_0 < y \leq \beta$ and $\alpha < \beta$) is the trickiest and the rest of this section is devoted to it. We will show that $\xi_2 > 0$, meaning that it is a valid local minimum, and then show that there exists a threshold y_c that can be used to determine whether $x = 0$ or $x = \xi_2$ is the global minimum.

First, let's show that $\xi_2 > 0$.

$$\xi_2 = \frac{y - \alpha + \sqrt{d}}{2} \overset{(1)}{>} \frac{y - \alpha}{2} \overset{(2)}{>} \frac{y_0 - \alpha}{2} = \sqrt{\alpha\beta} - \alpha = \sqrt{\alpha}(\sqrt{\beta} - \sqrt{\alpha}) \overset{(3)}{>} 0,$$

where the marked inequalities follow from the previously-established facts that (1) $d > 0$, (2) $y > y_0$, and (3) $\alpha < \beta$. This means that ξ_2 is a local minimum. However, since $J'(0; y) = \beta - y \geq 0$, our boundary point $x = 0$ may also be a local minimum. We therefore need to determine whether $J(0; y)$ or $J(\xi_2; y)$ is smaller.

To this end, let us define

$$\phi(y) \triangleq J(\xi_2(y); y) - J(0; y), \tag{4.18}$$

where the notation $\xi_2(y)$ simply reminds us that our expression for ξ_2 depends on y. Taking the derivative of this, we get

$$\begin{aligned}
\phi'(y) &= J'(\xi_2(y); y)\, \xi_2'(y) + \frac{\partial}{\partial y} J(\xi_2(y); y) - \frac{\partial}{\partial y} J(0; y) \\
&= \begin{bmatrix} \quad 0 \quad \end{bmatrix} \xi_2'(y) + \begin{bmatrix} \; y - \xi_2(y) \; \end{bmatrix} - \begin{bmatrix} \quad y \quad \end{bmatrix} \\
&= -\xi_2(y).
\end{aligned} \tag{4.19}$$

Here, we have used the fact that $J'(\xi_2; y) = 0$ since ξ_2 is a stationary point, and that $\partial J(x; y)/\partial y = y - x$. Since we have already established that $\xi_2(y) > 0$, this means that $\phi(y)$ is strictly monotonically decreasing in y over the interval $[y_0, \beta]$.

Next, we will show that $\phi(y_0) > 0$ while $\phi(\beta) < 0$. We had previously shown that $x = 0$ is the unique global minimum when $y = y_0$, and since $\xi_2(y_0) = \sqrt{\alpha\beta} - \alpha \neq 0$, we must have $J(0; y_0) < J(\xi_2(y_0); y_0)$ and hence $\phi(y_0) > 0$.

To show that $\phi(\beta) < 0$, note that $\xi_1(\beta) = 0$ whereas $\xi_2(\beta) = \beta - \alpha > 0$, which means that $J'(x; \beta) = \frac{x(x - \xi_2)}{x + \alpha} < 0$ for all x in the nonempty interval $(0, \xi_2)$. Since $\xi_2 > 0$, integrating $J'(x; \beta)$ from 0 to ξ_2 gives us $J(\xi_2; \beta) - J(0; \beta) < 0$ and hence $\phi(\beta) < 0$.

To summarize the last few paragraphs: $\phi(y_0) > 0$, $\phi(y)$ is monotonically decreasing over the interval $[y_0, \beta]$, and $\phi(\beta) < 0$. Therefore there must exist exactly one $y_c \in (y_0, \beta)$ such that $\phi(y_c) = 0$. This y_c acts as a threshold such that $x = 0$ is the unique global minimum for all $y < y_c$ and $x = \xi_2$ is the unique global minimum for all $y > y_c$. At $y = y_c$, the two are tied, and we can arbitrarily choose $x = 0$ as the minimum.

Unfortunately, I do not have a closed-form expression for y_c and we will need to find it numerically by solving for $\phi(y_c) = 0$. However, the fact that $\phi(y)$ is

monotonic and we know that $y_c \in (y_0, \beta)$ allows us to use very simple methods like bisection. Since the derivative $\phi'(y) = -\xi_2(y)$ is known and bounded away from zero, Newton's method can also be very effective. Furthermore, in a problem like sparse deconvolution, the cost of computing y_c is negligible compared to the cost of applying the proximal operator to the millions of entries in the feature vector $\boldsymbol{x}$.

4.1.3 Statement of the proximal operator for the log regularizer

For readers who skipped the previous section, we are considering the cost function

$$h(x) = \beta\alpha \log(|x| + \alpha), \tag{4.20}$$

where $\alpha, \beta > 0$, and we wish to derive an expression for the proximal operator

$$\text{prox}_h(y) = \arg\min_x h(x) + \frac{1}{2}(x - y)^2. \tag{4.21}$$

First, let us define the following:

$$y_0 = 2\sqrt{\alpha\beta} - \alpha \tag{4.22}$$

$$d(y) = (y - y_0)(y + \alpha + 2\sqrt{\alpha\beta}\,) \tag{4.23}$$

$$\xi(y) = \frac{1}{2}(y - \alpha + \sqrt{d(y)}\,) \tag{4.24}$$

$$= y - \frac{2\alpha\beta}{y + \alpha + \sqrt{d(y)}}. \tag{4.25}$$

The two expressions for $\xi(y)$ are mathematically equivalent, but the second form was slightly better-behaved in my numerical tests.

Next, let us define the constant

$$y_c = \begin{cases} \text{(see below)} & \alpha < \beta \\ \beta & \alpha \geq \beta. \end{cases} \tag{4.26}$$

In the case where $\alpha < \beta$, then y_c is the solution to $\phi(y_c) = 0$, where

$$\phi(y) = h(\xi(y)) - h(0) + \frac{1}{2}(\xi(y) - y)^2 - \frac{1}{2}y^2. \tag{4.27}$$

Note that $y_c \in (y_0, \beta)$ and that $\phi'(y) = -\xi(y) < 0$ in this interval, so numerical methods such as bisection or Newton's method would be appropriate here. In my numerical experiments, I found that Newton's method typically converged to machine epsilon within 6 iterations.

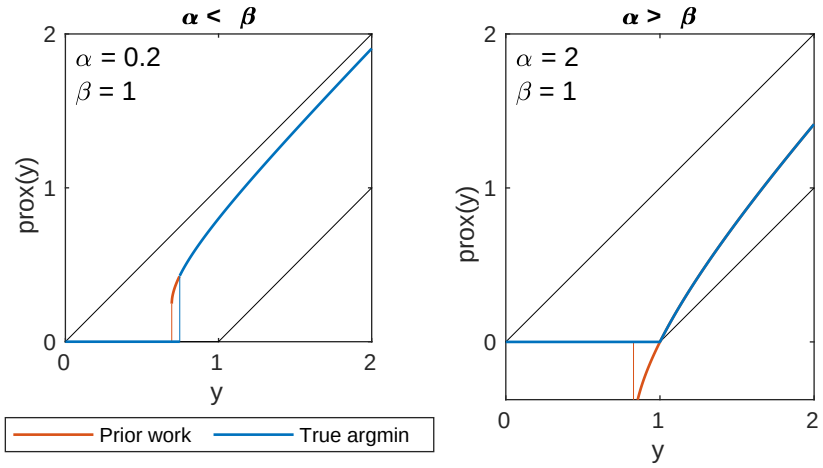

Figure 4.3: Comparison of the proximal operator to prior work. This figure compares our expression for the proximal operator (4.28) to the expression reported in the prior work (Malioutov and Aravkin, 2014). **Left:** When $\alpha < \beta$, there is a region in which there are two local minima (Figure 4.2A). The prior work switches to the nonzero minimum immediately, even though it is not the global minimum until y exceeds some higher threshold. **Right:** When $\alpha > \beta$, the prior work reports a negative result for some values of y, which is incorrect.

Then the proximal operator is given by

$$\text{prox}_h(y) = \begin{cases} 0 & |y| \leq y_c \\ \text{sgn}(y)\xi(|y|) & |y| > y_c. \end{cases} \tag{4.28}$$

Malioutov and Aravkin (2014) arrived at a similar expression, but they used y_0 instead of y_c, which leads to the inaccuracies shown in Figure 4.3.

4.2 Nested group regularizers

As a regularizer, the l_1 norm produces solutions with few active (i.e., nonzero) elements but does not care how these active elements are arranged. As we saw in Section 2.2.4, this is not appropriate for applications such as spike detection, where we may wish to impose a different sparsity structure on the solution. In this section, we will discuss how a generalization of the "group lasso" may be used to impose the desired sparsity structure.

Like the traditional lasso, the group lasso regularizer was originally developed for factor selection in regression problems (Yuan and Lin, 2006). It partitions the elements of x into groups G_i and takes the l_2 norm of each group

$$g(x) = \sum_i \beta_i \|x_{G_i}\|. \tag{4.29}$$

This produces solutions with few active groups, but does not further incentivize sparsity within a group. This can be also seen as a generalization of other norms that have been used for regularization: if the β_i are all the same and each group contains the same number of elements, then x may be reshaped into a matrix and this reduces to the matrix $l_{2,1}$ norm. If each group contains only one element of x, then this reduces to the l_1 norm.

The group lasso regularizer thus provides a mechanism by which we may impose some sparsity structure on the solution. For example, consider a single-channel spike detection problem (Figure 4.4A), where we've reshaped the x vector into a $T \times D$ matrix (T time steps, D spike basis waveforms)[3]. We expect spikes to be sparse over time, but once a spike is detected at time t, we don't expect the basis loading vector $x[t, :]$ to be sparse. We can produce this sparsity structure by creating a lasso group for each row of this matrix. Furthermore, since the l_2 norm is invariant under orthonormal transformations, this also guarantees that the sparsity pattern of our solution is invariant to orthonormal transformations of the D basis waveforms.

This extends naturally to the multi-channel case (Figure 4.4B), where we've reshaped the x vector into a $T \times DC$ matrix. We still expect spikes to be sparse over time, so we have a lasso group for each row of the matrix. Additionally, we now expect spikes to have a sparse spatial footprint (i.e., to be detected on a small number of channels),

[3] Apologies for transposing the matrix; Chapter 2 set up the features as a $D \times T$ matrix and opted for column-sparsity rather than row-sparsity.

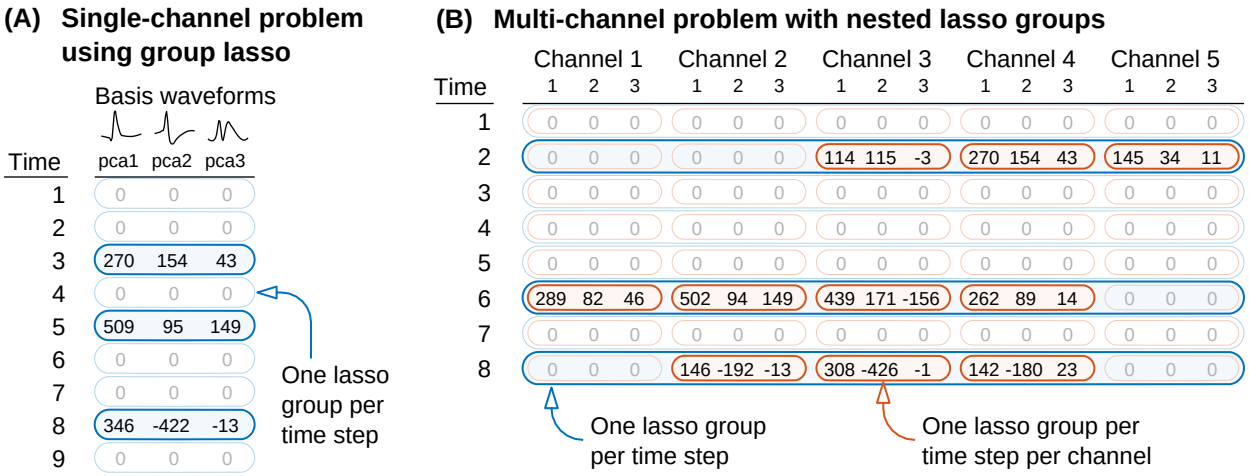

Figure 4.4: Group lasso regularizer applied to the spike detection deconvolution problem. The group lasso regularizer (equation 4.29) produces solutions that have few active groups, but does not further incentivize sparsity within a group. **(A)** In the single-channel case, we want our solution to be row-sparse, so we add a lasso group for every time step. **(B)** In the multi-channel case, we can encourage the detected spikes to have a sparse spatial footprint by adding a lasso group for each channel at every time step. These new lasso groups (red) are nested within the old lasso groups (blue), allowing for a simple proximal operator.

so we now add a lasso group for each of the channel subsets at every time step[4]. As before, this regularizer remains invariant to orthonormal transformations of the D basis waveforms for each channel.

Note that these new per-channel lasso groups (red) are nested within the cross-channel lasso groups (blue). It turns out that this nested structure is critically important for ensuring that this regularizer still has a simple proximal operator, and the next few subsections of this document are devoted to deriving this proximal operator. Section 4.2.1 discusses some prior work on nested and otherwise overlapping group regularizers. Section 4.2.2 introduces the concept of a *coordinatewise minimized at zero* (CMZ) function, which is exploited in Section 4.2.3 to derive rules for combining elementary functions into more complicated regularizers while maintaining a simple proximal operator. Finally, Section 4.2.4 uses these propositions to derive the proximal operator for the nested group lasso regularizer shown in Figure 4.4B.

[4]We did not include this spatial sparsity in Chapter 2, since we were primarily concerned with tetrode data, for which this spatial sparsity does not make sense. However, this may be of concern for data acquired using high-density probes.

4.2.1 Prior work

The main result of this section—that a regularizer constructed of nested l_2 norms (also known as "hierarchical" or "tree-structured" regularizer) admits a simple proximal operator—was previously derived by Jenatton et al. (2011) and, for a special case dubbed the "sparse group lasso", by Simon et al. (2013). However, both sets of authors relied heavily on the convexity of the regularizer and so it is not clear whether their results would apply in a nonconvex setting, such as with the log regularizer that we described in Section 4.1. Therefore[5] we will re-derive those results without assuming that the regularizer is convex. The proofs will be very different, but the results (the expressions for the proximal operators) will end up being very similar.

But before we continue, it is worth discussing some additional related work. Besides the l_2 norm, other forms of nested regularizers are also convenient to optimize: nested l_∞ norms also admit a simple proximal operator (Zhao, Rocha, and Yu, 2009; Jenatton et al., 2011), and nested l_1 norms can be converted into an equivalent non-overlapping regularizer.

For the more general case of overlapping groups (where they may not be nested in a hierarchical structure), the problem becomes more difficult. Yuan, Liu, and Ye (2011) consider applying the group lasso regularizer to non-nested groups, but are unable to find a simple expression for the proximal operator, instead reformulating it as a set of smaller constrained smooth optimization problems. Alternatively, Jacob, Obozinski, and Vert (2009) propose a reformulation of the group lasso regularizer that handles the overlapping case slightly differently and admits a simple proximal operator, but at the cost of a larger problem size (see also Rao et al., 2016). This can be applied to the multi-channel spike detection problem to create a point-set topology on the channels that participate in a single spike, which may be of interest for some applications.

4.2.2 Coordinatewise minimized at zero (CMZ) functions

This section and the next are pretty dense. The proofs are rather tedious and I can't promise that they'll offer any useful insight into the structure of the problem. Based on feedback from readers, I would advise skipping ahead to Section 4.2.4 to decide for yourself whether the payoff is worth the effort.

Let us take an additive approach to deriving the proximal operator for the nested

[5] This is a bit of revisionist history; in reality, I was simply unaware of their results when I set out to determine whether the log regularizer would be compatible with nested groups.

group regularizer. Assuming that we have a regularizer h with a simple proximal operator, how can we combine this with some elementary function f so that the combination still has a simple proximal operator?

It turns out that we will need some restrictions on h; specifically, we will be assuming that it is coordinatewise minimized at zero (CMZ), defined below.

Definition 1 (Coordinatewise minimized at zero (CMZ) and CMZ nondecreasing)**.** *A function $g : \mathbb{R}^n \mapsto \mathbb{R}$ is* coordinatewise minimized at zero *if for any $x \in \mathbb{R}^n$ and any coordinate index $i \in \{1, \ldots, n\}$, g is minimized by setting the coordinate $x_i = 0$ (holding the other coordinates $x_{j \neq i}$ fixed):*

$$\forall x, i \qquad 0 \in \arg\min_{x_i} g(\begin{bmatrix} x_1 \cdots x_{i-1} & x_i & x_{i+1} \cdots x_n \end{bmatrix}).$$

In other words, if we construct $\tilde{x}$ so that $\tilde{x}_i = 0$ but $\tilde{x}_j = x_j$ for $j \neq i$, then $g(\tilde{x}) \leq g(x)$. Additionally, g is CMZ nondecreasing *if $g(\alpha x + (1 - \alpha)\tilde{x})$ is nondecreasing in α for $\alpha \geq 0$.*

Note that the l_p norms $\|x\|_p$ are all CMZ nondecreasing, as are any nondecreasing function of these norms, such as $\frac{1}{2}\|x\|_2^2$. However, not all norms are CMZ; as a counterexample, consider

$$g(x) = x^\intercal \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}^{-1} x \qquad g(\begin{bmatrix} 1 \\ 1 \end{bmatrix}) = 0.4 \qquad g(\begin{bmatrix} 0 \\ 1 \end{bmatrix}) = 0.6.$$

Conversely, not all CMZ functions are norms; consider the l_0 "norm" or the log regularizer described in Section 4.1, both of which are CMZ but are not norms.

Before we continue, we will need to discuss some notation regarding the proximal operator. For a function $g : \mathbb{R}^n \mapsto \mathbb{R}$, let

$$J_g(x; y) \triangleq g(x) + \frac{1}{2}\|x - y\|_2^2. \tag{4.30}$$

This is the objective function being minimized by the proximal operator

$$\text{prox}_g(y) \triangleq \arg\min_x J_g(x; y). \tag{4.31}$$

g is often assumed to be convex, in which case J_g is strongly convex and therefore has a unique minimizer. However, if g is not convex, then there may be multiple x that minimize $J_g(x; y)$. To accommodate this case, this subsection and the next will treat $\text{prox}_g(y)$ as a set-valued function, so a statement like $\text{prox}_g(y) = \text{prox}_h(z)$

indicates that the set of minimizers of $J_g(x; y)$ is the same as the set of minimizers of $J_h(x; z)$. This is a slight abuse of notation, since the rest of this document (and the optimization literature in general) expects $\text{prox}_g(y)$ to be a regular (single-valued) function. In that context, if multiple x can minimize $J_g(x; y)$, then $\text{prox}_g(y)$ will select one arbitrarily.

We will always assume that a minimizer exists (i.e., $\text{prox}_g(y)$ is non-empty); this is guaranteed if g is convex or if g is CMZ and lower semicontinuous (Proposition 1).

Proposition 1. *If g is CMZ and lower semicontinuous, then $\text{prox}_g(y)$ is non-empty.*

Proof. The CMZ property implies that $g(0) \leq g(x)$ for all x, and therefore $J_g(x; y)$ is bounded below by $\phi(x; y) \triangleq g(0) + \frac{1}{2}\|x - y\|_2^2$. Note that ϕ is continuous and strongly convex, so its sublevel sets are closed and bounded (and therefore compact).

Consider the sublevel set $\mathcal{L}_0 \triangleq \{x : \phi(x; y) \leq \phi(0; y)\}$, which contains the point $x = 0$ by construction. Since g is lower semicontinuous, J_g is also lower semicontinuous, and by the extreme value theorem must attain a minimum within the compact set $\mathcal{L}_0$, i.e., there exists $x^\star \in \mathcal{L}_0$ such that $J_g(x^\star; y) \leq J_g(x; y)$ for all $x \in \mathcal{L}_0$. Furthermore,

$$\forall x \notin \mathcal{L}_0,\ J_g(x; y) \geq \phi(x; y) \overset{x \notin \mathcal{L}_0}{>} \phi(0; y) = J_g(0; y) \overset{0 \in \mathcal{L}_0}{\geq} J_g(x^\star; y).$$

so $x^\star$ is a global minimizer of $J_g(x; y)$. □

4.2.3 Composition of CMZ regularizers

This section presents 3 propositions regarding the composition of 3 specific functions with an arbitrary CMZ function. It is not meant to be a comprehensive list, just the minimum necessary to implement the nested group lasso presented in Figure 4.4B.

Propositions 3 and 4 deal with the case where the regularizer g can be written as the sum $g(x) = h(x) + f(x)$. If h is CMZ and f is the l_1 norm (Proposition 3) or a nonnegativity constraint (Proposition 4), then $\text{prox}_g(y) = \text{prox}_h(\text{prox}_f(y))$.

Proposition 5 deals with the case where the regularizer g can be written as a function of the l_2 norms of non-overlapping groups, i.e., $g(x) = h(\textsc{GroupNorms}(x))$. If h is CMZ, then $\text{prox}_g(y) = \textsc{ScaleGroups}(y, \text{prox}_h(\textsc{GroupNorms}(y)))$, where $\textsc{ScaleGroups}(y, a)$ scales the groups of y so that the resulting group norms are a.

These propositions also provide sufficient conditions under which g is also CMZ, allowing us to continue combining it with additional terms. This will allow us to

implement the nested group lasso, including the case where a nonconvex regularizer is applied to the top-level lasso groups. The details of this implementation are described in Section 4.2.4.

The rest of this section contains the propositions and their proofs. But first, Lemma 2 presents a general property of CMZ functions. It will be used extensively in the proofs of Propositions 3–5 to show that certain regions of solution space may be safely ignored.

Lemma 2 (The proximal operator of a CMZ function is weakly sign-consistent)**.** *If g is CMZ, then for any $x^\star \in \mathrm{prox}_g(y)$,*

$$\forall i,\ \mathrm{sgn}(x_i^\star) \in \{0\} \cup \{\mathrm{sgn}(y_i)\},$$

where $\mathrm{sgn}(\alpha) = 1$ *if* α *is strictly positive,* -1 *if strictly negative, and* 0 *if zero. In other words, $x^\star$ is weakly sign-consistent with y, i.e., $y_i \geq 0 \implies x_i^\star \geq 0$ and $y_i \leq 0 \implies x_i^\star \leq 0$.*

Proof by contradiction. Suppose that $x^\star$ is not weakly sign-consistent with y, i.e., there exists at least one coordinate such that $x_i^\star < 0 \leq y_i$ or $x_i^\star > 0 \geq y_i$. Let us consider an alternative $\tilde{x}$ such that $\tilde{x}_i = 0$ but $\tilde{x}_j = x_j^\star$ for $j \neq i$. Then

$$\begin{aligned} J_g(x^\star; y) - J_g(\tilde{x}; y) &= g(x^\star) - g(\tilde{x}) + \frac{1}{2}\sum_k (x_k^\star - y_k)^2 - (\tilde{x}_k - y_k)^2 \\ &= g(x^\star) - g(\tilde{x}) + \frac{1}{2}(x_i^\star - y_i)^2 - \frac{1}{2}(0 - y_i)^2 \\ &\overset{\text{(a)}}{\geq} \frac{1}{2}(x_i^\star)^2 - y_i x_i^\star \overset{\text{(b)}}{\geq} \frac{1}{2}(x_i^\star)^2 \overset{\text{(c)}}{>} 0. \end{aligned}$$

The inequalities follow from our assumptions that (a) g is CMZ, (b) either $x_i^\star < 0 \leq y_i$ or $x^\star > 0 \geq y_i$, and (c) $x_i^\star \neq 0$. This yields $J_g(x^\star; y) > J_g(\tilde{x}; y)$, which contradicts our assumption that $x^\star$ minimizes J_g. □

Proposition 3 (CMZ + weighted l_1 norm)**.** *For $x \in \mathbb{R}^n$, let $f(x) = \sum_i \beta_i |x_i|$ with $\beta_i \geq 0$. For reference, $z = \mathrm{prox}_f(y)$ is given by $z_i = \mathrm{sgn}(y_i)\max(0, |y_i| - \beta_i)$.*

If $h : \mathbb{R}^n \mapsto \mathbb{R}$ is CMZ, then $g(x) = h(x) + f(x)$ is also CMZ and

$$\mathrm{prox}_g(y) = \mathrm{prox}_h(\mathrm{prox}_f(y)). \tag{4.32}$$

Additionally, if h is CMZ nondecreasing, then g is CMZ nondecreasing as well.

Proof. First, note that f is CMZ nondecreasing because it is coordinatewise separable and the contribution from each coordinate $\beta_i|x_i|$ is nondecreasing as we move away from zero. If h is CMZ, then g is the sum of two CMZ functions and therefore CMZ. Likewise, if h is CMZ nondecreasing, then g is CMZ nondecreasing as well.

The proof of (4.32) is a bit longer, so I will outline it first. Let $z \triangleq \operatorname{prox}_f(y)$. We will define a set $C \subset \mathbb{R}^n$ and show that it satisfies the following properties:

$$\operatorname{prox}_h(z) \subseteq C \tag{4.33}$$

$$\forall x \in C,\ J_h(x; z) = J_g(x; y) + \text{constants} \tag{4.34}$$

$$\operatorname{prox}_g(y) \subseteq C. \tag{4.35}$$

In other words, this set C contains the unconstrained minimizers of both J_h and J_g, and within this set, the values of the objective functions J_h and J_g differ only by a constant that doesn't depend on x. Taken together, these imply (4.32) because

$$\operatorname{prox}_h(z) \overset{(4.33)}{=} \operatorname*{arg\,min}_{x \in C} J_h(x; z) \overset{(4.34)}{=} \operatorname*{arg\,min}_{x \in C} J_g(x; y) \overset{(4.35)}{=} \operatorname{prox}_g(y).$$

First, (4.33) follows directly from Lemma 2 because we will define C as the set of all x that are weakly sign-consistent with z:

$$C \triangleq \{x : \forall i,\ \operatorname{sgn}(x_i) \in \{0\} \cup \{\operatorname{sgn}(z_i)\}\}. \tag{4.36}$$

Next, (4.34) follows from the remarkable property that for all $x \in C$, the inner product $\langle x, y - z\rangle = f(x)$. Recall that $z = \operatorname{prox}_f(y)$ and hence

$$z_i = \operatorname{sgn}(y_i) \max(0, |y_i| - \beta_i)$$

$$y_i - z_i = \begin{cases} y_i & z_i = 0 \\ \beta_i \operatorname{sgn}(z_i) & \text{otherwise.} \end{cases}$$

For any $x \in C$, x will be weakly sign-consistent with z, therefore $\operatorname{sgn}(z_i)x_i = \operatorname{sgn}(x_i)x_i$ and $(z_i = 0) \implies (x_i = 0)$, hence

$$\forall x \in C, \quad x_i(y_i - z_i) = \begin{cases} 0 & z_i = 0 \\ \beta_i \operatorname{sgn}(x_i)x_i & \text{otherwise} \end{cases} = \beta_i|x_i|.$$

Summing over the coordinates i, we have $\langle x, y - z\rangle = f(x)$ and thus

$$\begin{aligned}\forall x \in C, \quad J_h(x; z) &= h(x) + \frac{1}{2}\|x - y\|_2^2 + \langle x - y, y - z\rangle + \frac{1}{2}\|y - z\|_2^2 \\ &= h(x) + \frac{1}{2}\|x - y\|_2^2 + \langle x, y - z\rangle + \text{constants} \\ &= h(x) + \frac{1}{2}\|x - y\|_2^2 + f(x) + \text{constants} \\ &= J_g(x; y) + \text{constants}.\end{aligned}$$

Finally, (4.35) mostly follows from Lemma 2. Let $x^\star \in \text{prox}_g(y)$. Since g and f are both CMZ, both $x^\star$ and z must be weakly sign-consistent with y, therefore we cannot have $x_i^\star < 0 < z_i$ or $x_i^\star > 0 > z_i$. In order to conclude that $x^\star \in C$, we will also need to rule out the possibility that $x_i^\star \neq 0 = z_i$, which we will do in the next paragraph.

Suppose this were true. Let us construct $\tilde{x}$ such that $\tilde{x}_i = 0$ and $\tilde{x}_{j\neq i} = x^\star_{j\neq i}$. Since f is separable, let $f_i(\xi) \triangleq \beta_i|\xi|$ so that $f(x) = \sum_i f_i(x_i)$, and note that $z_i = \arg\min_\xi J_{f_i}(\xi; y_i)$ is the unique minimizer since f_i is convex (and hence J_{f_i} is strongly convex). Then

$$\begin{aligned}J_g(x^\star; y) - J_g(\tilde{x}; y) &= h(x^\star) - h(\tilde{x}) + \sum_k J_{f_k}(x_k^\star; y_k) - J_{f_k}(\tilde{x}_k; y_k) \\ &= h(x^\star) - h(\tilde{x}) + J_{f_i}(x_i^\star; y_i) - J_{f_i}(0; y_i) \\ &\overset{(a)}{\geq} J_{f_i}(x_i^\star; y_i) - J_{f_i}(0; y_i) \overset{(b)}{>} 0.\end{aligned}$$

The inequalities follow from our assumptions that (a) h is CMZ and (b) $z_i = 0$ uniquely minimizes J_{f_i} and $x_i^\star \neq 0$. This yields $J_g(x^\star; y) > J_g(\tilde{x}; y)$, contradicting $x^\star \in \text{prox}_g(y)$.

To summarize, we defined $z \triangleq \text{prox}_g(y)$ and $C \subset \mathbb{R}^n$ as the set of all x that are weakly sign-consistent with z (4.36). Then (4.33) $\impliedby$ (h is CMZ), (4.34) $\impliedby$ ($\forall x \in C, \langle x, y - z\rangle = f(x)$), and (4.35) $\impliedby$ (f, g, h are CMZ and f is coordinatewise separable and convex). Together, these imply that $\text{prox}_g(y) = \text{prox}_h(\text{prox}_f(y))$. We will use this again in Proposition 4. □

Proposition 4 (CMZ + nonnegativity constraint)**.** *For $x \in \mathbb{R}^n$, let $f(x) = \sum_{i\in\mathcal{I}} \mathbb{1}_{\geq 0}(x_i)$, where the indicator function $\mathbb{1}_{\geq 0}(x_i)$ is 0 if $x_i \geq 0$ or $+\infty$ otherwise, and $\mathcal{I}$ is the set of coordinates that the nonegativity constraint applies to. For reference, $z = \text{prox}_f(y)$ is given by $z_i = \max(0, y_i)$ for $i \in \mathcal{I}$ and $z_i = y_i$ for $i \notin \mathcal{I}$.*

If $h : \mathbb{R}^n \mapsto \mathbb{R}$ is CMZ, then $g(x) = h(x) + f(x)$ is also CMZ and

$$\text{prox}_g(y) = \text{prox}_h(\text{prox}_f(y)). \tag{4.37}$$

Additionally, if h is CMZ nondecreasing, then g is CMZ nondecreasing as well.

Proof. First, note that f is CMZ nondecreasing because it is coordinatewise separable and the contribution from each coordinate $\mathbb{1}_{\geq 0}(x_i)$ is nondecreasing as we move away from zero. If h is CMZ, then g is the sum of two CMZ functions and therefore CMZ. Likewise, if h is CMZ nondecreasing, then g is CMZ nondecreasing as well.

The proof of (4.37) follows the same outline as that of Proposition 3. Let $z \triangleq \text{prox}_f(y)$ and let $C \triangleq \{x : \forall i,\ \text{sgn}(x_i) \in \{0\} \cup \{\text{sgn}(z_i)\}\}$. We will show that

$$\text{prox}_h(z) \overset{\text{(a)}}{=} \arg\min_{x \in C} J_h(x; z) \overset{\text{(b)}}{=} \arg\min_{x \in C} J_g(x; y) \overset{\text{(c)}}{=} \text{prox}_g(y).$$

As we discussed in the proof of Proposition 3, these set equality relationships ultimately follow from (a) h is CMZ, (b) $\forall x \in C, \langle x, y - z\rangle = f(x)$, and (c) f, g, h are CMZ and f is coordinatewise separable and convex. We have already shown the necessary conditions for (a) and (c) and will show (b) in the next paragraph.

First, note that the definition of z gives us either $z_i = y_i$ (in which case $y_i - z_i = 0$) or $z_i = 0$ (in which case $x_i = 0$ for any $x \in C$), therefore $x_i(y_i - z_i) = 0$ and hence $\langle x, y - z\rangle = 0$. Next, note that $z_i \geq 0$ for all $i \in \mathcal{I}$, so for any $x \in C$, we must also have $x_i \geq 0$ for all $i \in \mathcal{I}$ and hence $f(x) = 0$. We therefore have $\forall x \in C, \langle x, y - z\rangle = 0 = f(x)$. □

Proposition 5 (CMZ with nested group l_2 norm)**.** *For $x \in \mathbb{R}^n$ and a set of m groups G_i that partition the coordinates of x, let $f : \mathbb{R}^n \mapsto \mathbb{R}^m_+$ compute the l_2 norm of each group. In other words, if $a = f(x)$, then $a_i = \|x_{G_i}\|_2$. Elsewhere in this document, $f(x)$ is also known as* GroupNorms*(x).*

Let us also define a linear function $\phi_y : \mathbb{R}^m \to \mathbb{R}^n$ such that $x = \phi_y(a)$ is given by

$$x_{G_i} \triangleq \begin{cases} 0 & y_{G_i} = 0 \\ \frac{y_{G_i}}{\|y_{G_i}\|_2} a_i & \textit{otherwise}. \end{cases} \tag{4.38}$$

Elsewhere in this document, $\phi_y(a)$ is also known as ScaleGroups*(y, a).*

If $h : \mathbb{R}^m \mapsto \mathbb{R}$ is CMZ and $g(x) = h(f(x))$, then

$$\text{prox}_g(y) = \phi_y(\text{prox}_h(f(y))). \tag{4.39}$$

Additionally, if h is CMZ nondecreasing, then g is CMZ nondecreasing as well.

Proof. First, note that each coordinate of the vector-valued $f(x)$ is CMZ nondecreasing in x, so if h is also CMZ nondecreasing then the composition $g(x) = h(f(x))$ must be CMZ nondecreasing as well. Note that this requires h to be nondecreasing (otherwise g may not even be CMZ) but that is not necessary for the rest of this proof.

For the main result, our first step will be to show that the optimal value of the proximal problem $\min_x J_g(x; y)$ is the same as that of $\min_a J_h(a; b)$, where $b \triangleq f(y)$. To begin, let us rewrite the proximal problem as a nested optimization:

$$\begin{aligned}\min_x J_g(x; y) &= \min_{a \in \mathbb{R}^m_+} \min_{x: f(x)=a} h(f(x)) + \frac{1}{2}\|x - y\|_2^2 \\ &= \min_{a \in \mathbb{R}^m_+} h(a) + \frac{1}{2} \min_{x: f(x)=a} \|x - y\|_2^2. \end{aligned} \tag{4.40}$$

Our assumption that the groups G_i partition the coordinates of x yields $\|x - y\|_2^2 = \sum_i \|x_{G_i} - y_{G_i}\|_2^2$, allowing us to separate the inner minimization problem (4.40) over i:

$$x^\star \in \operatorname*{arg\,min}_{x: f(x)=a} \|x - y\|_2^2 \quad \iff \quad \forall i, \;\; x^\star_{G_i} \in \operatorname*{arg\,min}_{x_{G_i}: \|x_{G_i}\|_2 = a_i} \|x_{G_i} - y_{G_i}\|_2^2.$$

Each sub-problem is simply minimizing the distance from a sphere to a point, and so

$$\forall a_i \geq 0, \quad \operatorname*{arg\,min}_{x_{G_i}: \|x_{G_i}\|_2 = a_i} \|x_{G_i} - y_{G_i}\|_2^2 = \begin{cases} \{u : \|u\|_2 = a_i\} & y_{G_i} = 0 \\ \{a_i \frac{y_{G_i}}{\|y_{G_i}\|_2}\} & \text{otherwise,} \end{cases} \tag{4.41}$$

achieving a minimum value of $(a_i - \|y_{G_i}\|_2)^2$. Substituting this into (4.40) with $b \triangleq f(y)$,

$$\min_x J_g(x; y) = \min_{a \in \mathbb{R}^m_+} h(a) + \frac{1}{2}\|a - b\|_2^2 = \min_{a \in \mathbb{R}^m_+} J_h(a; b).$$

Since h is CMZ, the nonnegativity constraint on a is unnecessary as long as b is nonnegative (Proposition 4), and so we have

$$\min_x J_g(x; y) = \min_a J_h(a; b). \tag{4.42}$$

How does this help? If we are given an $a^\star \in \operatorname{prox}_h(b)$ and are able to construct some $\hat{x}$ such that $J_g(\hat{x}; y) = J_h(a^\star; b)$, then (4.42) implies that $\hat{x}$ minimizes J_g, i.e., $\hat{x} \in \operatorname{prox}_g(y)$.

One more thing: let us return to the inner minimization problem of (4.40) and denote the set of minimizers (described in equation 4.41) as $C_a \triangleq \arg\min_{x: f(x)=a} \|x - y\|_2^2$.

We already noted that for all $x \in C_a$, we have $\|x - y\|_2^2 = \|a - b\|_2^2$. Since $C_a \subseteq \{x : f(x) = a\}$, we also have $g(x) = h(f(x)) = h(a)$, and adding these together we get

$$\forall a \in \mathbb{R}_+^m, x \in C_a, \quad J_g(x; y) = J_h(a; b). \tag{4.43}$$

Using (4.42) and (4.43), we will show that for any $a^\star \in \text{prox}_h(b)$, we have $\phi_y(a^\star) \in \text{prox}_g(y)$ and therefore $\phi_y(\text{prox}_h(b)) \subseteq \text{prox}_g(y)$. The function $\phi_y : \mathbb{R}^m \mapsto \mathbb{R}^n$ was defined in (4.38). Conversely, we will also show that for any $x^\star \in \text{prox}_g(y)$, there exists some $a^\star \in \text{prox}_h(b)$ such that $\phi_y(a^\star) = x^\star$, therefore $\text{prox}_g(y) \subseteq \phi_y(\text{prox}_h(b))$. This pair of subset relationships thus imply that $\text{prox}_g(y) = \phi_y(\text{prox}_h(b))$, our main result.

First, given an $a^\star \in \text{prox}_h(b)$ we will show that $\phi_y(a^\star) \in \text{prox}_g(y)$. Note that

$$y_{G_i} = 0 \overset{b_i = \|y_{G_i}\|_2}{\iff} b_i = 0 \overset{\text{Lemma 2}}{\implies} a_i^\star = 0$$

and therefore $\{u : \|u\|_2 = a_i^\star\} = \{0\}$ for any i such that $y_{G_i} = 0$. Substituting this into (4.41) yields the definition of ϕ_y (equation 4.38), hence $C_{a^\star} = \{\phi_y(a^\star)\}$ and therefore

$$\phi_y(a^\star) \in C_{a^\star} \overset{(4.43)}{\implies} J_g(\phi_y(a^\star); y) = J_h(a^\star; b) \overset{(4.42)}{\implies} \phi_y(a^\star) \in \text{prox}_g(y)$$

Conversely, given an $x^\star \in \text{prox}_g(y)$, we will show $f(x^\star) \in \text{prox}_h(b)$ and $\phi_y(f(x^\star)) = x^\star$. First, note that $x^\star \in C_{f(x^\star)}$, otherwise any $\tilde{x} \in C_{f(x^\star)}$ will have $J_g(\tilde{x}; y) < J_g(x^\star; y)$, a contradiction. Then

$$x^\star \in C_{f(x^\star)} \overset{(4.43)}{\implies} J_g(x^\star; y) = J_h(f(x^\star); b) \overset{(4.42)}{\implies} f(x^\star) \in \text{prox}_h(b).$$

We previously showed that $C_{a^\star}$ is the singleton set $\{\phi_y(a^\star)\}$ for any $a^\star \in \text{prox}_h(b)$. Applying that to this situation, we therefore have $C_{f(x^\star)} = \{\phi_y(f(x^\star))\}$ and hence $x^\star = \phi_y(f(x^\star))$. □

4.2.4 Proximal operator for nested groups with a nonconvex regularizer

The multi-channel group lasso regularizer presented in Figure 4.4B may be written as

$$g(x) = \sum_t \left\| x[t, :, :] \right\| + \sum_t \sum_c \left\| x[t, c, :], \right\|$$

where we have reshaped x into a $[T \times C \times D]$ matrix (T time steps, C channels, D spike basis waveforms per channel). The first term $\|x[t,:,:]\|_2$ corresponds to the blue lasso groups (one per time step), and it should be interpreted as the vector l_2 norm applied to a "flattened" vector of length CD. The second term $\|x[t,c,:]\|_2$ corresponds to the red lasso groups (one per time step per channel).

Hopefully it is apparent from this setup that the $x[t,c,:]$ groups are nested within the $x[t,:,:]$ groups. To make this example a bit more realistic, we wish to apply the nonconvex log regularizer to the outermost group (the one-per-time-step group) and we will introduce weighting parameters β_t and β_c on these groups:

$$g(x) = \beta_t \sum_t \operatorname{logreg}\left(\left\|x[t,:,:]\right\|\right) + \beta_c \sum_t \sum_c \left\|x[t,c,:]\right\|. \tag{4.44}$$

Next, we will need to rewrite this a little to make it compatible with the propositions of the previous section. Given a multidimensional array, let GroupNorms return the l_2 norm over the last dimension. So we can start to rewrite (4.44) by noting that

$$\begin{aligned}\|x[t,c,:]\| &= \textsc{GroupNorms}(x)[t,c] \\ \|x[t,:,:]\| &= \|(\textsc{GroupNorms}(x)[t,:])\|\end{aligned}$$

and hence we can write $g(x)$ as

$$\begin{aligned}g(x) &= g_1(\textsc{GroupNorms}(x)) \\ g_1(\xi) &= \beta_t \sum_t \operatorname{logreg}\left(\left\|\xi[t,:]\right\|\right) + \beta_c \sum_t \sum_c \xi[t,c].\end{aligned}$$

Applying Proposition 5 then gives us

$$\begin{aligned}\operatorname{prox}_g(y) &= \textsc{ScaleGroups}(y, \xi^\star) \\ \xi^\star &= \operatorname{prox}_{g_1}(\textsc{GroupNorms}(y))\end{aligned}$$

Formal definitions of ScaleGroups and GroupNorms are given in the statement of Proposition 5.

Now we need the proximal operator of g_1. Since $\xi[t,c] \geq 0$, we can rewrite g_1 as

$$\begin{aligned}g_1(x) &= g_2(x) + \beta_c \|x\|_1 \\ g_2(x) &= \beta_t \sum_t \operatorname{logreg}\left(\left\|x[t,:]\right\|\right).\end{aligned}$$

Applying Proposition 3 and noting that the proximal operator of the l_1 norm is $\text{prox}_{\beta\|\cdot\|_1}(y) = \textsc{Shrink}_\beta(y)$, we have

$$\text{prox}_{g_1}(y) = \text{prox}_{g_2}(\textsc{Shrink}_{\beta_c}(y))$$

Now we need the proximal operator of g_2. Once again, we can use the group norm

$$\begin{aligned} g_2(x) &= g_3(\textsc{GroupNorms}(x)) \\ g_3(\xi) &= \beta_t \sum_t \text{logreg}(\xi[t]) \end{aligned}$$

and then Proposition 5 to yield

$$\begin{aligned} \text{prox}_{g_2}(y) &= \textsc{ScaleGroups}(y, \xi^\star) \\ \xi^\star &= \text{prox}_{g_3}(\textsc{GroupNorms}(y)). \end{aligned}$$

Finally, $g_3(x)$ is the same regularizer that we analyzed in Section 4.1, so we can stop here. But hopefully it is clear from this exercise how we could continue in the case where we had more nested groups to consider.

As far as the regularizer (4.44), we can compile our results into the following algorithm:

$$\begin{aligned} \psi_0 &= y \\ \psi_1 &= \text{prox}_{\beta_c\|\cdot\|_1}(\textsc{GroupNorms}(\psi_0)) \\ \psi_2 &= \text{prox}_{\beta_t \text{ logreg}}(\textsc{GroupNorms}(\psi_1)) \\ \xi_2 &= \psi_2 \\ \xi_1 &= \textsc{ScaleGroups}(\psi_1, \xi_2) \\ \xi_0 &= \textsc{ScaleGroups}(\psi_0, \xi_1) \\ x &= \xi_0. \end{aligned}$$

This is a bit more efficient than the expression derived by Jenatton et al. (2011), since we can take advantage of data reduction at each of the GroupNorms stages, but is otherwise very similar.

Chapter 5

CONCLUSION

Spike sorting is a challenging problem that has resisted many attempts at simple solutions. In part this is due to the heterogeneity of data that one may encounter in practice: chronic vs. acute recordings, tetrode vs. silicon probe, and differences among species and brain areas all present different sets of challenges that defy a one-size-fits-all solution. Instead, a modular approach that can be tailored to the specific problem at hand seems more appropriate.

In this document, I have described two new modules—spike detection via sparse deconvolution and clustering using a mixture of drifting t-distributions—and this chapter discusses their contributions, how they fit into the overall problem of spike sorting, and potential avenues for future work.

Spike detection via sparse deconvolution offers improved spike detection by posing it as a deconvolution problem in a spike subspace. This enables us to separate moderately-overlapping spikes without incurring the main disadvantages of template matching approaches, namely (1) the necessity of fitting source templates to each putative neuron, and (2) inextricably coupling the problems of spike detection and clustering. Instead, it represents each detected spike as a point in a low-dimensional feature space, a format that is amenable to subsequent analysis using a wide variety of techniques.

As such, this new approach is a drop-in replacement for traditional spike detection and dimensionality reduction. It retains the practical advantages of being a single-pass operation that dramatically reduces data size while preserving enough information to perform clustering and evaluate cluster overlap. Without any changes to the rest of the spike-sorting toolchain, it offers improved performance on overlapping spikes without impacting performance on non-overlapping spikes (Figure 2.4). Spike removal (Figure 2.5) is also easily incorporated into an existing LFP analysis toolchain.

One natural direction for future work is extending this approach to high-density silicon probes, in which a spike should only be detected on a local subset of channels (in contrast, a spike is typically detected on all channels of a tetrode). Although these spikes will still be represented in a feature space that spans all the probe's channels,

a sparse representation of may be more appropriate for subsequent analysis (Kadir, Goodman, and Harris, 2014).

Another direction for future work is to consider applications to other detection problems. In addition to spike sorting in extracellular recordings, this deconvolution approach may also be useful for spike detection in intra- or juxta-cellular recordings, either because we wish to perform spike removal for analysis of subthreshold membrane voltage, or because these spikes may be difficult to detect (as in intracellular recordings from insect neurons). Alternatively, by using a different set of basis waveforms—one tailored to LFP events such as ripples or sharp waves—we can use this approach to detect other types of neural events and represent them as points in a feature space. Note that this need not be a prelude to clustering; this low-dimensional feature space representation also enables downstream analysis in which this characterization is correlated with other neural or behavioral phenomena.

Finally, it may be useful to develop theoretical guarantees on detection with this approach. One approach is to recognize that the sparse deconvolution problem (2.1) with $l_{2,1}$ regularizer (2.5) is equivalent to maximum-likelihood estimation[1] with a matched subspace detector (Scharf and Friedlander, 1994), which can be used to derive false positive rates and other performance metrics. Another approach is for the case of single-cell spike detection (the intra- or juxta-cellular recordings discussed above), where the cell refractory period allows us to use a different measure of dictionary coherence (Papyan, Sulam, and Elad, 2017) that may enable us to make stronger guarantees on sparse recovery with particular algorithms.

The second spike-sorting module described in this document is clustering using a mixture of drifting t-distributions (MoDT) model. This generative model captures two important features of experimental data—cluster drift and heavy tails—and its fitting procedure is efficient and naturally robust to outliers. This model can then be used wherever a simpler model, such as a mixture-of-Gaussians, has traditionally been used. When used for spike sorting, the MoDT model can (1) increase unit yield by separating clusters that would appear to overlap under a stationary model and (2) decrease user workload by reducing the incidence of clusters that are spuriously split due to drift. As a unit isolation metric, the MoDT model provides more accurate estimates of misclassification error than previous approaches. Importantly, this use case can be applied to data that have been spike-sorted using other techniques,

[1] In this MLE framework, the log regularizer 2.6 is equivalent to a Lomax (a shifted version of the Pareto Type II distribution) prior on the signal magnitude.

including non-parametric techniques that otherwise do not produce estimates of unit quality. Section 3.5 also describes a method for resolving highly-overlapping spikes without recourse to the raw data, although this method may be applied to any generative model, not just the MoDT model.

The main advantage of using the MoDT model for spike sorting, however, is that it enables longer recordings to be clustered as a single segment. Although linking these segments is still necessary for continuous recordings over several days or weeks, the use of longer segments dramatically simplifies the process. Longer segments ensure that all units will fire enough spikes to be clustered, reduces the number of segments that need to be linked, and enables the use of overlapping segments. For example, a week-long recording can be broken into 21 ten-hour segments with an overlap of two hours. These two-hour overlapping segments are long enough to ensure that all units will fire a sufficient number of spikes that we can establish cluster correspondences between these segments based on the spike assignments of the overlapping data.

One direction for future work is to incorporate a more sophisticated model for cluster drift. The current model allows each cluster to drift independently, but a major source of cluster drift is rigid motion of the recording electrodes along the insertion axis. This produces correlated changes in cluster location across all of the observed clusters, and accounting for this phenomenon may improve tracking of neurons with a low firing rate.

Another direction for future work is the analysis of spikes recorded from high-density probes. The challenges here are (1) ensuring that there is enough data to fit the heteroskedastic cluster covariance matrices and (2) keeping the computational complexity in check. Since these are the same problems encountered in scaling up a mixture-of-Gaussians model, many approaches have already been developed to mitigate these concerns (Section 3.2.5).

In conclusion, these contributions to spike detection and clustering make concrete steps to improving certain aspects of the spike sorting problem. Each is based in a theoretical framework that enables future work to build upon and extend these results. Finally, these methods are not restricted to spike sorting but may be applied more broadly to problems of event detection and non-stationary clustering in general.

BIBLIOGRAPHY

Anderson, B. and J. B. Moore (1979). *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall.

Auslender, A. and M. Teboulle (2006). "Interior gradient and proximal methods for convex and conic optimization". *SIAM Journal on Optimization* 16.3, pp. 697–725.

Bankman, I. N., K. O. Johnson, and W. Schneider (1993). "Optimal detection, classification, and superposition resolution in neural waveform recordings". *IEEE transactions on biomedical engineering* 40.8, pp. 836–841.

Bar-Hillel, A., A. Spiro, and E. Stark (Oct. 2006). "Spike sorting: Bayesian clustering of non-stationary data". *J. Neurosci. Methods* 157.2, pp. 303–316. DOI: 10.1016/j.jneumeth.2006.04.023.

Bayram, I. (2015). "Penalty Functions Derived From Monotone Mappings." *IEEE Signal Process. Lett.* 22.3, pp. 264–268.

Beck, A. and M. Teboulle (2009). "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". *SIAM journal on imaging sciences* 2.1, pp. 183–202.

Becker, S. R., E. J. Candès, and M. C. Grant (2011). "Templates for convex cone problems with applications to sparse signal recovery". *Mathematical programming computation* 3.3, p. 165.

Boyd, S., N. Parikh, E. Chu, B. Peleato, and J. Eckstein (July 2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". *Found. Trends Mach. Learn.* 3.1, pp. 1–122. DOI: 10.1561/2200000016.

Bristow, H., A. Eriksson, and S. Lucey (2013). "Fast convolutional sparse coding". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 391–398.

Calabrese, A. and L. Paninski (Mar. 2011). "Kalman filter mixture model for spike sorting of non-stationary data." *J. Neurosci. Methods* 196.1, pp. 159–169. DOI: 10.1016/j.jneumeth.2010.12.002.

Candes, E. J. (2008). "The restricted isometry property and its implications for compressed sensing". *Comptes rendus mathematique* 346.9-10, pp. 589–592.

Candes, E. J., M. B. Wakin, and S. P. Boyd (2008). "Enhancing sparsity by reweighted l_1 minimization". *Journal of Fourier analysis and applications* 14.5-6, pp. 877–905.

Carlson, D. E., V. Rao, J. Vogelstein, and L. Carin (2013). "Real-Time Inference for a Gamma Process Model of Neural Spiking". *Adv. Neural Inf. Process. Syst.* Pp. 2805–2813.

Chen, S. S., D. L. Donoho, and M. A. Saunders (2001). "Atomic decomposition by basis pursuit". *SIAM review* 43.1, pp. 129–159.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". *J. R. Stat. Soc.* 39, pp. 1–38. DOI: 10.2307/2984875.

Dhawale, A. K., R. Poddar, S. B. Wolff, V. A. Normand, E. Kopelowitz, and B. P. Ölveczky (2017). "Automated long-term recording and analysis of neural activity in behaving animals". *Elife* 6, e27702. DOI: 10.7554/eLife.27702.

Donoho, D. L., M. Elad, and V. N. Temlyakov (2006). "Stable recovery of sparse overcomplete representations in the presence of noise". *IEEE Transactions on information theory* 52.1, pp. 6–18.

Fee, M. S., P. P. Mitra, and D. Kleinfeld (Dec. 1996). "Variability of extracellular spike waveforms of cortical neurons". *J. Neurophysiol.* 76.6, pp. 3823–3833. DOI: 10.1152/jn.00865.2015.

Feldman, D., M. Faulkner, and A. Krause (2011). "Scalable Training of Mixture Models via Coresets". *Adv. Neural Inf. Process. Syst.* Pp. 2142–2150.

Foster, D. J. and M. A. Wilson (2006). "Reverse replay of behavioural sequences in hippocampal place cells during the awake state". *Nature* 440.7084, p. 680.

Fournier, J., C. M. Mueller, M. Shein-Idelson, M. Hemberger, and G. Laurent (Aug. 2016). "Consensus-Based Sorting of Neuronal Spike Waveforms". *PLoS ONE* 11.8, e0160494. DOI: 10.1371/journal.pone.0160494.

Franke, F., M. Natora, C. Boucsein, M. H. J. Munk, and K. Obermayer (2010). "An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes". *J. Comput. Neurosci.* 29.1-2, pp. 127–148.

Harris, K. D., R. Q. Quiroga, J. Freeman, and S. L. Smith (Aug. 2016). "Improving data quality in neuronal population recordings." *Nature Neurosci.* 19.9, pp. 1165–1174. DOI: 10.1038/nn.4365.

Heide, F., W. Heidrich, and G. Wetzstein (2015). "Fast and flexible convolutional sparse coding". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5135–5143.

Hill, D. N., S. B. Mehta, and D. Kleinfeld (June 2011). "Quality metrics to accompany spike sorting of extracellular signals." *J. Neurosci.* 31.24, pp. 8699–8705.

Horowitz, M. B., I. Papusha, and J. W. Burdick (2014). "Domain decomposition for stochastic optimal control". *53rd IEEE Conference on Decision and Control*. IEEE, pp. 1866–1873.

Jacob, L., G. Obozinski, and J.-P. Vert (2009). "Group lasso with overlap and graph lasso". *Proceedings of the 26th annual international conference on machine learning*. ACM, pp. 433–440.

Jenatton, R., J. Mairal, G. Obozinski, and F. Bach (2011). "Proximal methods for hierarchical sparse coding". *Journal of Machine Learning Research* 12.Jul, pp. 2297–2334.

Kadir, S. N., D. F. M. Goodman, and K. D. Harris (Nov. 2014). "High-dimensional cluster analysis with the masked EM algorithm." *Neural Comput.* 26.11, pp. 2379–2394. DOI: 10.1162/NECO_a_00661.

Lan, G., Z. Lu, and R. D. Monteiro (2011). "Primal-dual first-order methods with $O(1/\epsilon)$ iteration-complexity for cone programming". *Mathematical Programming* 126.1, pp. 1–29.

Lange, K. L., R. J. Little, and J. M. Taylor (1989). "Robust statistical modeling using the t distribution". *Journal of the American Statistical Association* 84.408, pp. 881–896.

Lee, S. and G. J. McLachlan (Mar. 2014). "Finite mixtures of multivariate skew t-distributions: some recent and new results". *Stat. Comput.* 24.2, pp. 181–202.

Lewicki, M. S. (1998). "A review of methods for spike sorting: the detection and classification of neural action potentials". *Network: Comput. Neural Syst.* 9, R53–R78.

Li, Z., S. Ding, and Y. Li (2015). "Dictionary learning with log-regularizer for sparse representation". *Digital Signal Processing (DSP), 2015 IEEE International Conference on*. IEEE, pp. 609–613.

Magdon-Ismail, M. and J. T. Purnell (Sept. 2010). "Approximating the Covariance Matrix of GMMs with Low-Rank Perturbations". *Intelligent Data Engineering and Automated Learning – IDEAL 2010*. Springer, pp. 300–307.

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

McLachlan, G. and D. Peel (2000). *Finite Mixture Models*. John Wiley & Sons.

Meng, X., J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. B. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar (Apr. 2016). "MLlib: Machine learning in apache spark". *J. Mach. Learn. Res.* 17.34.

Mukhopadhyay, S. and G. C. Ray (1998). "A new interpretation of nonlinear energy operator and its efficacy in spike detection". *IEEE Trans. Biomed. Eng.* 45.2, pp. 180–187. DOI: 10.1109/10.661266.

Neal, R. M. and G. E. Hinton (1998). "A view of the EM algorithm that justifies incremental, sparse, and other variants". *Learning in Graphical Models*. Dordrecht: Springer Netherlands, pp. 355–368.

Needell, D. and R. Vershynin (2009). "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit". *Foundations of computational mathematics* 9.3, pp. 317–334.

Nenadic, Z. and J. W. Burdick (2005). "Spike detection using the continuous wavelet transform". *IEEE Transactions on Biomedical Engineering* 52.1, pp. 74–87.

Nesterov, Y. (2013). "Gradient methods for minimizing composite functions". *Mathematical Programming* 140.1, pp. 125–161.

Pachitariu, M., N. A. Steinmetz, S. N. Kadir, M. Carandini, and K. D. Harris (2016). "Fast and accurate spike sorting of high-channel count probes with KiloSort". *Advances in Neural Information Processing Systems*, pp. 4448–4456.

Papyan, V., J. Sulam, and M. Elad (2017). "Working locally thinking globally: Theoretical guarantees for convolutional sparse coding". *IEEE Transactions on Signal Processing* 65.21, pp. 5687–5701.

Parikh, N. and S. Boyd (2014). "Proximal algorithms". *Foundations and Trends® in Optimization* 1.3, pp. 127–239.

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

Peel, D. and G. J. McLachlan (Oct. 2000). "Robust mixture modelling using the t distribution". *Stat. Comput.* 10.4, pp. 339–348.

Pnevmatikakis, E. A., D. Soudry, Y. Gao, T. A. Machado, J. Merel, D. Pfau, T. Reardon, Y. Mu, C. Lacefield, W. Yang, M. Ahrens, R. Bruno, T. M. Jessell, D. S. Peterka, R. Yuste, and L. Paninski (2016). "Simultaneous denoising, deconvolution, and demixing of calcium imaging data". *Neuron* 89.2, pp. 285–299.

Pouzat, C., M. Delescluse, P. Viot, and J. Diebolt (June 2004). "Improved Spike-Sorting By Modeling Firing Statistics and Burst-Dependent Spike Amplitude Attenuation: A Markov Chain Monte Carlo Approach". *J. Neurophysiol.* 91.6, pp. 2910–2928. DOI: 10.1152/jn.00227.2003.

Prentice, J. S., J. Homann, K. D. Simmons, G. Tkačik, V. Balasubramanian, and P. C. Nelson (2011). "Fast, scalable, Bayesian spike identification for multi-electrode arrays." *PLoS ONE* 6.7, e19884. DOI: 10.1371/journal.pone.0019884.

Rao, N. S., R. D. Nowak, C. R. Cox, and T. T. Rogers (2016). "Classification With the Sparse Group Lasso." *IEEE Trans. Signal Processing* 64.2, pp. 448–463.

Rauch, H. E., F. Tung, and C. T. Striebel (1965). "Maximum likelihood estimates of linear dynamic systems". *AIAA journal* 3.8, pp. 1445–1450.

Rossant, C., S. N. Kadir, D. F. M. Goodman, J. Schulman, M. L. D. Hunter, A. B. Saleem, A. Grosmark, M. Belluscio, G. H. Denfield, A. S. Ecker, A. S. Tolias, S. Solomon, G. Buzsáki, M. Carandini, and K. D. Harris (Mar. 2016). "Spike sorting for large, dense electrode arrays". *Nature Neurosci.* 19.4, pp. 634–641. DOI: 10.1038/nn.4268.

Rousseeuw, P. J. and K. van Driessen (Aug. 1999). "A fast algorithm for the minimum covariance determinant estimator". *Technometrics* 41.3, pp. 212–223. DOI: 10.1080/00401706.1999.10485670.

Scharf, L. L. and B. Friedlander (1994). "Matched subspace detectors". *IEEE Transactions on signal processing* 42.8, pp. 2146–2157.

Schmitzer-Torbert, N., J. Jackson, D. A. Henze, K. D. Harris, and A. D. Redish (2005). "Quantitative measures of cluster quality for use in extracellular recordings." *Neuroscience* 131.1, pp. 1–11. DOI: 10.1016/j.neuroscience.2004.09.066.

Shalchyan, V. and D. Farina (Feb. 2014). "A non-parametric Bayesian approach for clustering and tracking non-stationarities of neural spikes". *J. Neurosci. Methods* 223, pp. 85–91. DOI: 10.1016/j.jneumeth.2013.12.005.

Shan, K. Q., E. V. Lubenov, M. Papadopoulou, and A. G. Siapas (2016). "Spatial tuning and brain state account for dorsal hippocampal CA1 activity in a non-spatial learning task". *Elife* 5, e14321. DOI: 10.7554/eLife.14321.001.

Shan, K. Q., E. V. Lubenov, and A. G. Siapas (2017). "Model-based spike sorting with a mixture of drifting t-distributions". *Journal of neuroscience methods* 288, pp. 82–98. DOI: 10.1016/j.jneumeth.2017.06.017.

Shoham, S., M. R. Fellows, and R. A. Normann (Aug. 2003). "Robust, automatic spike sorting using mixtures of multivariate t-distributions." *J. Neurosci. Methods* 127.2, pp. 111–122.

Siapas, A. G. and M. A. Wilson (1998). "Coordinated interactions between hippocampal ripples and cortical spindles during slow-wave sleep". *Neuron* 21.5, pp. 1123–1128.

Simon, N., J. Friedman, T. Hastie, and R. Tibshirani (2013). "A sparse-group lasso". *Journal of Computational and Graphical Statistics* 22.2, pp. 231–245.

Snider, R. K. and A. B. Bonds (Oct. 1998). "Classification of non-stationary neural signals". *J. Neurosci. Methods* 84.1-2, pp. 155–166.

Swaminathan, A. and R. M. Murray (2014). "Identification of Markov Chains From Distributional Measurements and Applications to Systems Biology". *IFAC Proceedings Volumes* 47.3, pp. 4400–4405.

Takehara, K., S. Kawahara, and Y. Kirino (2003). "Time-dependent reorganization of the brain components underlying memory retention in trace eyeblink conditioning". *Journal of Neuroscience* 23.30, pp. 9897–9905.

Taylor, H. L., S. C. Banks, and J. F. McCoy (1979). "Deconvolution with the l_1 norm". *Geophysics* 44.1, pp. 39–52.

Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso". *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288.

Tolias, A. S., A. S. Ecker, A. G. Siapas, A. Hoenselaar, G. A. Keliris, and N. K. Logothetis (Dec. 2007). "Recording chronically from the same neurons in awake, behaving primates." *J. Neurophysiol.* 98.6, pp. 3780–3790. DOI: 10.1152/jn.00260.2007.

Udell, M., C. Horn, R. Zadeh, and S. Boyd (2016). "Generalized Low Rank Models". *Found. Trends Mach. Learn.* 9.1, pp. 1–118. DOI: 10.1561/2200000055.

Vogelstein, J. T., A. M. Packer, T. A. Machado, T. Sippy, B. Babadi, R. Yuste, and L. Paninski (2010). "Fast nonnegative deconvolution for spike train inference from population calcium imaging". *Journal of neurophysiology* 104.6, pp. 3691–3704.

Wang, Y., Q. Yao, J. T. Kwok, and L. M. Ni (2018). "Scalable Online Convolutional Sparse Coding". *IEEE Transactions on Image Processing*.

Wierzynski, C. M., E. V. Lubenov, M. Gu, and A. G. Siapas (2009). "State-dependent spike-timing relationships between hippocampal and prefrontal circuits during sleep". *Neuron* 61.4, pp. 587–596.

Wohlberg, B. (2016). "Efficient algorithms for convolutional sparse representations". *IEEE Transactions on Image Processing* 25.1, pp. 301–315.

Wolf, M. T. and J. W. Burdick (Nov. 2009). "A Bayesian clustering method for tracking neural signals over successive intervals." *IEEE Trans. Biomed. Eng.* 56.11, pp. 2649–2659. DOI: 10.1109/TBME.2009.2027604.

Yang, X. and S. A. Shamma (1988). "A totally automated system for the detection and classification of neural spikes". *IEEE Transactions on Biomedical Engineering* 35.10, pp. 806–816.

Yger, P., G. L. Spampinato, E. Esposito, B. Lefebvre, S. Deny, C. Gardella, M. Stimberg, F. Jetter, G. Zeck, S. Picaud, J. Deubel, and O. Marre (2018). "A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo". *eLife* 7, e34518. DOI: 10.7554/eLife.34518.

Yuan, L., J. Liu, and J. Ye (2011). "Efficient methods for overlapping group lasso". *Advances in Neural Information Processing Systems*, pp. 352–360.

Yuan, M. and Y. Lin (2006). "Model selection and estimation in regression with grouped variables". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.1, pp. 49–67.

Zhao, P., G. Rocha, and B. Yu (2009). "The composite absolute penalties family for grouped and hierarchical variable selection". *The Annals of Statistics* 37.6A, pp. 3468–3497.