Short-and-Sparse Deconvolution – A Geometric Approach

Yenson Lau[*,†], Qing Qu[*,♯], Han-Wen Kuo[†], Pengcheng Zhou[◇], Yuqian Zhang[♭], and John Wright[†,‡]

[†]Department of Electrical Engineering and Data Science Institute, Columbia University
[♯]Center for Data Science, New York University
[◇]Department of Statistics and Center for Theoretical Neurosicence, Columbia University
[♭]Department of Computer Science, Cornell University
[‡]Department of Applied Physics and Applied Mathematics, Columbia University

July 27, 2020

Abstract

Short-and-sparse deconvolution (SaSD) is the problem of extracting localized, recurring motifs in signals with spatial or temporal structure. Variants of this problem arise in applications such as image deblurring, microscopy, neural spike sorting, and more. The problem is challenging in both theory and practice, as natural optimization formulations are nonconvex. Moreover, practical deconvolution problems involve smooth motifs (kernels) whose spectra decay rapidly, resulting in poor conditioning and numerical challenges. This paper is motivated by recent theoretical advances [ZLK$^+$17, KZLW19], which characterize the optimization landscape of a particular nonconvex formulation of SaSD. This is used to derive a *provable* algorithm which exactly solves certain non-practical instances of the SaSD problem. We leverage the key ideas from this theory (sphere constraints, data-driven initialization) to develop a *practical* algorithm, which performs well on data arising from a range of application areas. We highlight key additional challenges posed by the ill-conditioning of real SaSD problems, and suggest heuristics (acceleration, continuation, reweighting) to mitigate them. Experiments demonstrate both the performance and generality of the proposed method.

Index terms— sparse blind deconvolution, convolutional dictionary learning, computational imaging, nonconvex optimization, alternating descent methods.

[*]These authors contributed equally to this work.

1 Introduction

Signals in medical/scientific/natural imaging can often be modeled as superpositions of basic, recurring motifs (see Figure 1 for an illustration). For example, in calcium imaging [SGHK03, GK12], the excitation of neurons produces short pulses of fluorescence, repeating at distinct firing times. In the material and biological sciences, repeated motifs often encode crucial information about the subject of interest; e.g., in nanomaterials these motifs correspond to defects in the crystal lattice due to doping [CSL$^+$18]. In all of these applications, the motifs of interest are *short*, and they are *sparsely* distributed within the sample of interest. Signals with this short-and-sparse structure also arise in natural image processing: when a blurry image is taken due to the resolution limit or malfunction of imaging procedure, it can be modeled as a short blur pattern applied to a visually plausible sharp image [CW98, RBZ06a, LWDF11a].

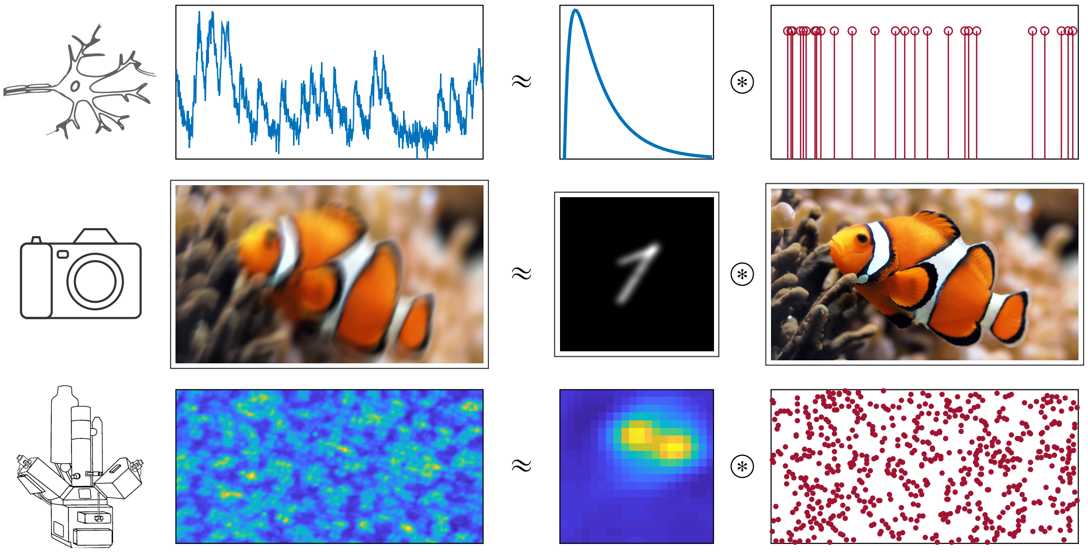


Figure 1: Natural signals with short-and-sparse structure. In calcium imaging (top), each neuronal spike induces a fluoresence pattern measuring a transient increase in calcium concentration. In photography (middle), photos with sharp edges (sparse in the gradient domain) are often obfuscated by blurring due to shaking the camera. In scanning tunneling microscopy (bottom), dopants embedded in some base material produce individual electronic signatures. For each of these cases, the observed signal can be modeled as a convolution between a *short* kernel and a *sparse* activation map.

Mathematically, an observed signal $\boldsymbol{y}$ with this *short-and-sparse* (SaS) structure can be modeled as a *convolution*[1] of a *short* signal $\boldsymbol{a}_0 \in \mathbb{R}^{n_0}$ and a much longer *sparse* signal $\boldsymbol{x}_0 \in \mathbb{R}^m$ $(m \gg n_0)$:

$$\boldsymbol{y} \quad = \quad \boldsymbol{a}_0 \;\circledast\; \boldsymbol{x}_0. \tag{1.1}$$

In all of the above applications, the signals $\boldsymbol{a}_0$ and $\boldsymbol{x}_0$ are not known ahead of time. The *short-and-sparse deconvolution* (SaSD) problem asks us to recover these two signals from the observation $\boldsymbol{y}$. This is a challenging *inverse* problem: natural optimization formulations are *nonconvex* and have many equivalent solutions. The kernel $\boldsymbol{a}_0$ is often smooth, and hence attenuates high frequencies. Although study of the SaSD problem stretches back several decades and across several disciplines [Hay94, LB95, KH96], the need for efficient, reliable, and general purpose optimization methods remains.

One major challenge associated with developing methods for SaSD arises from our relatively limited understanding of the global geometric structure of nonconvex optimization problems. Our goal is to recover this ground truth $(\boldsymbol{a}_0, \boldsymbol{x}_0)$ (perhaps up to some trivial ambiguities), which typically requires us to obtain a globally optimal solution to a nonconvex optimization problem. This is impossible in general. Fortunately, recent theoretical evidence [ZKW18, KZLW19] guarantees that the SaSD problem can solved efficiently under certain

[1] For simplicity we follow the convention of [KZLW19] and use cyclic convolution throughout this paper, unless otherwise specified. The choice is superficial; any algorithms and results discussed here should also apply to linear convolution with minor modifications.

idealized assumptions. Using an appropriate selection of optimization domain and a specific initialization scheme, these results yield provable methods that solve certain instances of SaSD in polynomial time.

Unfortunately, practical SaSD problems raise additional challenges beyond the assumptions in theory, causing the provable methods [ZKW18, KZLW19] to fail on real problem instances. While the emphasis of [ZLK+17, KZLW19] is on theoretical guarantees, here we focus on practical computation. We show how to combine ideas from this theory with heuristics that better address the properties of practical deconvolution problems, to build a novel method that performs well on data arising in a range of application areas. Many of our design choices are natural and have a strong precedent in the literature. We will show how these natural choices help to cope with the (complicated!) geometry of practically occurring deconvolution problems. A critical issue in moving from theory to practice is the poor conditioning of naturally-occurring deconvolution problems: we show how to address this with a combination of ideas from sparse optimization, including momentum, continuation, and reweighting. The end result is a general purpose method, which we demonstrate on data for spike recovery [FZP17] and neuronal localization [PSG+16] from calcium imaging data, as well as fluorescence microscopy [RBZ06a].

Organization of the paper. The remainder of the paper is organized as follows. Section 2 introduces key aspects of SaSD, and Section 3 shows how they play out in a theoretical analysis of SaSD, culminating in a provable algorithm grounded in geometric intuition. In Section 4, we discuss how to combine this intuition with additional heuristics to create practical methods. Section 5 revisits and demonstrates these ideas in a simulated setting. Section 6 illustrates the performance of our method on data drawn from a number of applications. Finally, Section 7 reviews the literature, and poses interesting future directions.

Reproducible research. The code for implementations of our algorithms can be found online:

`https://github.com/qingqu06/sparse_deconvolution`.

For more details of our work on SaSD, we refer interested readers to our project website

`https://deconvlab.github.io/`.

2 Two Key Intuitions for SaS Deconvolution

We begin by describing two basic intuitions for SaS deconvolution, which play an important role in the geometry of optimization and the design of efficient methods.

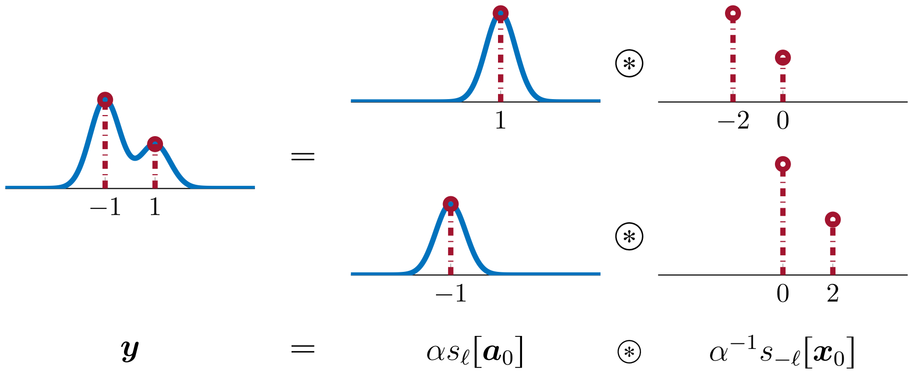


Figure 2: Scaling-shift symmetry. The SaS convolution model exhibits a scaled shift symmetry: $\alpha s_\ell[\boldsymbol{a}_0]$ and $\alpha^{-1} s_{-\ell}[\boldsymbol{x}_0]$ have the same convolution as $\boldsymbol{a}_0$ and $\boldsymbol{x}_0$. Therefore, the ground truth $(\boldsymbol{a}_0, \boldsymbol{x}_0)$ can only by identified up to some scale and shift ambiguity.

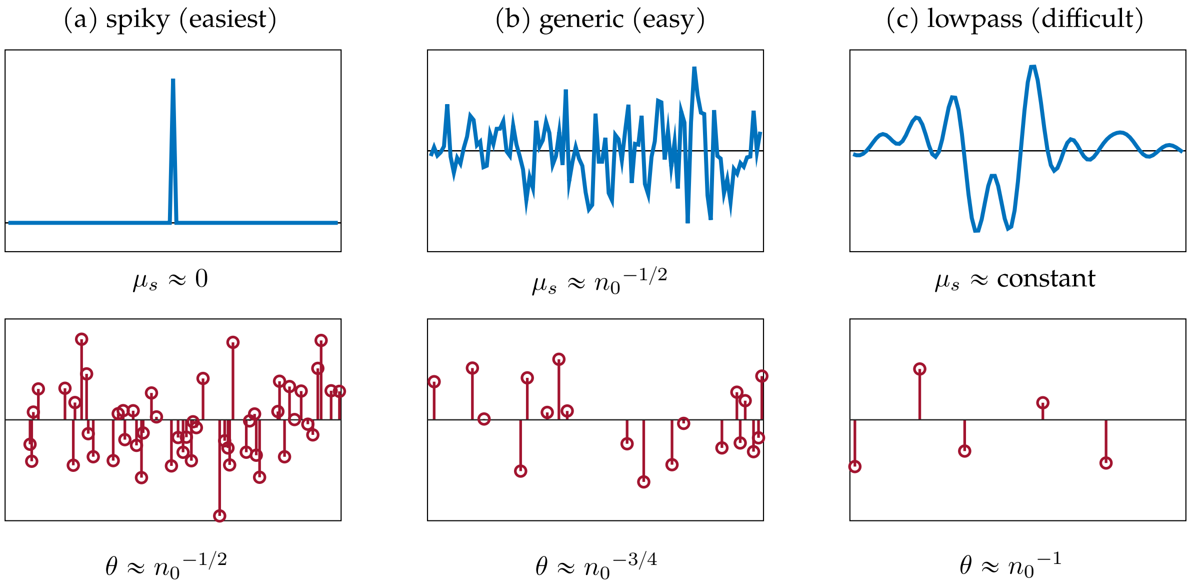


Figure 3: Sparsity-coherence tradeoff [KZLW19]: examples with varying coherence parameter $\mu_s(\boldsymbol{a}_0)$ and sparsity rate θ (i.e., probability a given entry is nonzero). Smaller shift-coherence $\mu_s(\boldsymbol{a}_0)$ allows SaSD to be solved with higher θ, and vice versa. In order of increasing difficulty: (a) when $\boldsymbol{a}_0$ is a Dirac delta function, $\mu_s(\boldsymbol{a}_0) = 0$; (b) when $\boldsymbol{a}_0$ is sampled from a uniform distribution on the sphere $\mathbb{S}^{n_0-1}$, its shift-coherence is roughly $\mu_s(\boldsymbol{a}_0) \approx {n_0}^{-1/2}$; (c) when $\boldsymbol{a}_0$ is low-pass, $\mu_s(a_0) \rightarrow$ const. as n_0 grows.

Symmetry structure. The SaS model exhibits a basic *scaled shift symmetry*: for any nonzero scalar α and cyclic shift $s_\ell[\cdot]$

$$\boldsymbol{y} \;=\; \boldsymbol{a}_0 \circledast \boldsymbol{x}_0 \;=\; (\pm\alpha s_\ell[\boldsymbol{a}_0]) \circledast (\pm\alpha^{-1} s_{-\ell}[\boldsymbol{x}_0]).$$

In other words, shifting $\boldsymbol{a}_0$ to the right by ℓ samples and shifting $\boldsymbol{x}_0$ to the left by the same amount leaves the convolution $\boldsymbol{a}_0 \circledast \boldsymbol{x}_0$ unchanged (see Figure 2). We can therefore only expect to recover the ground truth $(\boldsymbol{a}_0, \boldsymbol{x}_0)$ up to some scaling and some shift. As a result, natural optimization formulations for SaSD exhibit multiple global minimizers, corresponding to these scaled shifts of the ground truth. Due to the existence of multiple discrete global minimizers, natural formulations are *nonconvex*. Fortunately, this symmetry structure often creates leads to benign objective landscapes for optimization; two such examples for SaSD are [ZKW18, KZLW19].

Sparsity-coherence tradeoff. Clearly, not all SaSD problems are equally easy to solve. Problems with denser $\boldsymbol{x}_0$ are more challenging. Moreover, there is a basic tradeoff between the sparsity of the spike train $\boldsymbol{x}_0$ and the properties of the kernel $\boldsymbol{a}_0$. If $\boldsymbol{a}_0$ is smooth (e.g., Gaussian), then each occurrence of $\boldsymbol{a}_0$ would, on average, need to be relatively far apart to be distinguishable; i.e. $\boldsymbol{x}_0$ would have to be sparser. Conversely, denser instances of $\boldsymbol{x}_0$ should be allowable if $\boldsymbol{a}_0$ is "spikier".[2] One way of formalizing this tradeoff is through the *shift-coherence* of the kernel $\boldsymbol{a}_0$, which measures the "similarity" between $\boldsymbol{a}_0$ and its cyclic-shifts:

$$\mu_s(\boldsymbol{a}_0) \doteq \max_{\ell \neq 0} \left| \left\langle \frac{\boldsymbol{a}_0}{\|\boldsymbol{a}_0\|_2}, \frac{\mathrm{s}_\ell[\boldsymbol{a}_0]}{\|\boldsymbol{a}_0\|_2} \right\rangle \right| \in [0,1]. \tag{2.1}$$

As $\mu_s(\boldsymbol{a}_0)$ increases, the shifts of $\boldsymbol{a}_0$ become more correlated and hence closer together on the sphere. [KZLW19] uses this quantity to study the behavior of a particular nonconvex formulation of SaSD. For generic choices of $\boldsymbol{x}_0$, such as $\boldsymbol{x}_0 \sim \mathcal{BG}(\theta)$ drawn from a Bernoulli-Gaussian distribution, the *sparsity-coherence tradeoff* of [KZLW19] guarantees recoverability when the sparsity rate θ is sufficiently small relative to $\mu_s(\boldsymbol{a}_0)$. Intuitively speaking, this implies that SaSD problems with smaller $\mu_s(\boldsymbol{a}_0)$ tend to be "easier" to solve (Figure 3).

[2] Similar tradeoffs occur in non-blind deconvolution where $\boldsymbol{a}_0$ is known (e.g. [CFG14]) and in other inverse problems.

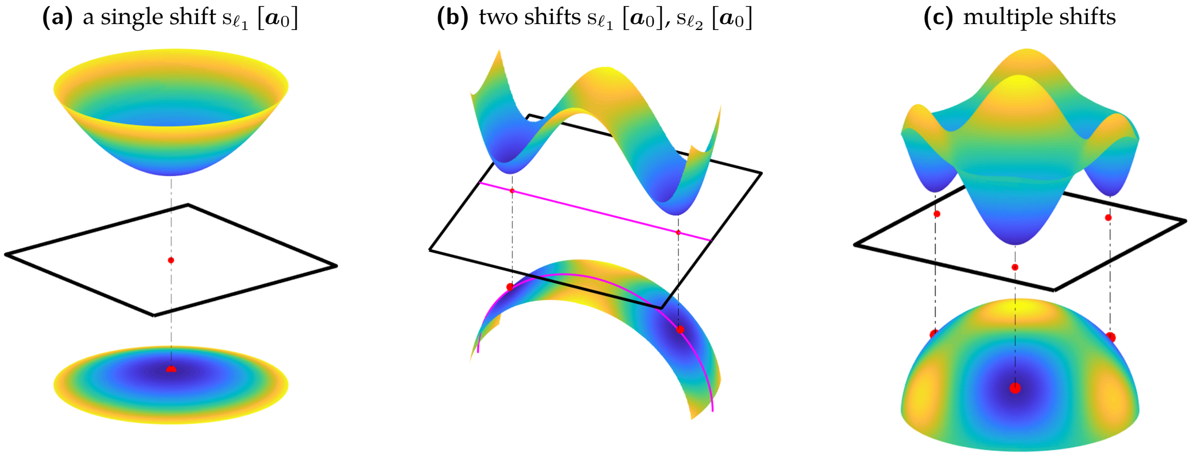

(a) a single shift $\mathrm{s}_{\ell_1}[\boldsymbol{a}_0]$
(b) two shifts $\mathrm{s}_{\ell_1}[\boldsymbol{a}_0]$, $\mathrm{s}_{\ell_2}[\boldsymbol{a}_0]$
(c) multiple shifts

Figure 4: Geometry of Approximate Bilinear Lasso loss $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$ near superpositions of shifts of $\boldsymbol{a}_0$ [KZLW19]. **Top:** function values of $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$ visualized as height. **Bottom:** heat maps of $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$ on the sphere $\mathbb{S}^{n-1}$. (**a**) the region near a single shift is strongly convex; (**b**) the region between two shifts contains a saddle-point, with negative curvature pointing towards each shift and positive curvature pointing away; (**c**) region near the span of several shifts of $\boldsymbol{a}_0$.

In the next section, we will use the idealized formulation of [KZLW19] to illustrate how these basic properties of the SaSD problem shape the landscape of optimization. In later sections, we will borrow these ideas to develop practical, general purpose methods. The major challenge in moving from theory to practice is in coping with highly coherent $\boldsymbol{a}_0$: in most practical applications, $\boldsymbol{a}_0$ is smooth and hence $\mu_s(\boldsymbol{a}_0)$ is large.

3 Problem Formulation and Nonconvex Geometry

In this section, we summarize some recent algorithmic theory characterizing the optimization landscape of an idealized nonconvex formulation for SaSD [ZLK+17, KZLW19], with the goal of applying the geometric intuition from this theory towards designing practical optimization methods.

3.1 The Bilinear Lasso and its marginalization

A natural idea for solving SaSD is to minimize a reconstruction loss $\psi(\boldsymbol{a}, \boldsymbol{x})$ between $\boldsymbol{a} \circledast \boldsymbol{x}$ and $\boldsymbol{y}$, plus a sparsity-promoting regularizer $g(\boldsymbol{x})$ on $\boldsymbol{x}$. This can be achieved, for instance, by minimizing the squared reconstruction error in combination with an ℓ_1-penalty on $\boldsymbol{x}$,

$$\boxed{\min_{\boldsymbol{a},\boldsymbol{x}} \quad \Psi_{\mathrm{BL}}(\boldsymbol{a}, \boldsymbol{x}) \quad \doteq \quad \tfrac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{a} \circledast \boldsymbol{x}\right\|_2^2 \; + \; \lambda \left\|\boldsymbol{x}\right\|_1, \qquad \text{s.t.} \quad \boldsymbol{a} \in \mathbb{S}^{n-1}.} \tag{3.1}$$

This *Bilinear Lasso* problem (BL) resembles the Lasso estimator in statistics [Tib96], and is a nonconvex optimization problem. The sparsity of the solution for $\boldsymbol{x}$ is controlled by the regularization penalty λ: a larger λ leads to sparser $\boldsymbol{x}$, and vice versa[3]. We constrain $\boldsymbol{a}$ onto the sphere $\mathbb{S}^{n-1}$, which reduces the scaling ambiguity into a sign ambiguity. We also increase the dimension of $\boldsymbol{a}$ to $n = 3n_0 - 2$; this creates an objective landscape that allows various descent methods to recover a full shift of $\boldsymbol{a}_0$ and avoid any shift-truncation effects, upon the application of a simple data initialization scheme.

In this paper, we will also frequently refer to the *marginalized* Bilinear Lasso cost

$$\varphi_{\mathrm{BL}}(\boldsymbol{a}) \;\doteq\; \min_{\boldsymbol{x}} \; \Psi_{\mathrm{BL}}(\boldsymbol{a}, \boldsymbol{x}) \;=\; \min_{\boldsymbol{x}} \; \tfrac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{a} \circledast \boldsymbol{x}\right\|_2^2 \; + \; \lambda \left\|\boldsymbol{x}\right\|_1. \tag{3.2}$$

It is clear that minimizing $\Psi_{\mathrm{BL}}(\boldsymbol{a}, \boldsymbol{x})$ is equivalent to minimizing $\varphi_{\mathrm{BL}}(\boldsymbol{a})$ over $\boldsymbol{a} \in \mathbb{S}^{n-1}$.

[3] [KZLW19] suggests a good choice $\lambda \in \mathcal{O}(1/\sqrt{\theta n})$, where $\theta \in (0, 1)$ denotes the sparsity level.

3.2 Structured nonconvexity and geometric properties

To understand the nonconvex optimization landscape of the Bilinear Lasso, it is natural to study the marginalized objective in Equation (3.2). The benefit of this approach is twofold: (i) for a fixed $\boldsymbol{a}$, the Lasso problem in Equation (3.2) is convex w.r.t. $\boldsymbol{x}$, and (ii) the *short* kernel $\boldsymbol{a}$ lives on a low dimensional manifold — the space $\boldsymbol{a} \in \mathbb{S}^{n-1}$ is where measure concentrates when $\boldsymbol{x}_0$ is generic random and has high dimension $(m \gg n_0)$. Unfortunately, $\varphi_{\mathrm{BL}}(\boldsymbol{a})$ remains challenging for analysis; a major culprit is that the Lasso estimator in Equation (3.2) does not usually admit closed-form solutions.

Approximate Bilinear Lasso. When $\boldsymbol{a}$ is incoherent $(\mu_s(\boldsymbol{a}) \approx 0)$, however, we approximately have $\|\boldsymbol{a} \circledast \boldsymbol{x}\|_2^2 \approx \|\boldsymbol{x}\|_2^2$. Carrying this approximation through to Equation (3.1) yields an *Approximate Bilinear Lasso* (ABL) objective[4] $\varphi_{\mathrm{ABL}}(\boldsymbol{a}, \boldsymbol{x}) = \min_{\boldsymbol{x}} \Psi_{\mathrm{ABL}}(\boldsymbol{a}, \boldsymbol{x})$, which satisfies $\varphi_{\mathrm{ABL}}(\boldsymbol{a}) \approx \varphi_{\mathrm{BL}}(\boldsymbol{a})$ whenever $\mu_s(\boldsymbol{a}) \approx 0$ [KZLW19]. For the purposes of our discussion, this objective serves as a valid simplification of the Bilinear Lasso when the true kernel is itself incoherent $(\mu_s(\boldsymbol{a}_0) \approx 0)$. Although such incoherence assumptions are stringent and impractical, $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$ admits a simple analytical form and is more amenable to analysis as a result.

Geometry of φ_{ABL} in the span of a few shifts. Under the assumptions that $\boldsymbol{a}_0$ is incoherent and $\boldsymbol{x}_0$ is generic, $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$ enjoys a number of nice properties on the sphere $\mathbb{S}^{n-1}$. In particular, Kuo et al. [KZLW19] provides a geometrical characterization of the optimization landscape $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$ near the span of several shifts[5] of $\boldsymbol{a}_0$:

1. *Near a single shift of $\boldsymbol{a}_0$.* Within a local neighborhood of each shift $s_\ell[\boldsymbol{a}_0]$, the optimization landscape of $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$ exhibits *strong convexity* (Figure 4a), with a *unique* minimizer corresponding to a shift $s_\ell[\boldsymbol{a}_0]$.

2. *In the vicinity of two shifts.* Near the span of two shifts,

$$\mathcal{S}_{\{\ell_1,\ell_2\}} \;=\; \left\{\, \alpha_1 s_{\ell_1}[\boldsymbol{a}_0] + \alpha_2 s_{\ell_2}[\boldsymbol{a}_0] \;:\; \alpha_1,\ \alpha_2 \in \mathbb{R} \,\right\} \bigcap \mathbb{S}^{n-1},$$

   the only local minimizers are approximately $s_{\ell_1}[\boldsymbol{a}_0]$ and $s_{\ell_2}[\boldsymbol{a}_0]$. A saddle point $\boldsymbol{a}_s$ exists at the symmetric superposition of the shifts (i.e. $\boldsymbol{a}_s = \alpha_1 s_{\ell_1}[\boldsymbol{a}_0] + \alpha_2 s_{\ell_2}[\boldsymbol{a}_0]$ with $\alpha_1 \approx \alpha_2$), but can be escaped by taking advantage of the large negative curvature present[6] (Figure 4b).

3. *In the vicinity of multiple shifts.* The geometric properties for two shifts carry over to those of multiple shifts of $\boldsymbol{a}_0$. Any local minimizers over

$$\mathcal{S}_{\mathcal{I}} \;\doteq\; \left\{\, \textstyle\sum_{\ell\in\mathcal{I}} \alpha_\ell s_\ell\,[\boldsymbol{a}_0] \;:\; \alpha_\ell \in \mathbb{R} \,\right\} \bigcap \mathbb{S}^{n-1} \tag{3.3}$$

   are again close to signed shifts (Figure 4c). Any saddle-points present sit at symmetric superpositions of two or more shifts, and exhibit strong negative curvature in directions towards the participating shifts. Additionally, the function value of $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$ increases when moving away from $\mathcal{S}_{\mathcal{I}}$.

[KZLW19] proves that these geometric properties of φ_{ABL} hold for sufficiently small[7] $|\mathcal{I}|$ whenever the sparsity-coherence tradeoff $n_0\theta \lessapprox \mu_s^{-1/2}(\boldsymbol{a}_0)$ is satisfied. This bound is stringent, however, and shows that the ABL formulation is unsuited for practical applications where $\mu_s(\boldsymbol{a}_0)$ often approaches one as n_0 grows.

The benign optimization landscape of $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$ provides strong implications for optimization. Indeed, if we could initialize $\boldsymbol{a}$ near $\mathcal{S}_{\mathcal{I}}$, iterates of many local descent methods such as [Gol80, CGT00, BAC18, NP06] can exploit gradient and negative curvature to remain near $\mathcal{S}_{\mathcal{I}}$, and eventually converge to the target solution

[4] As our focus here is on solving the Bilinear Lasso, we intentionally omit the concrete form of $\Psi_{\mathrm{ABL}}(\boldsymbol{a})$ and $\varphi_{\mathrm{ABL}}(\boldsymbol{a})$. Readers may refer to Section 2 of [KZLW19] for more details.

[5] When optimizing over $\mathbb{S}^{n-1}$, $n = 3n_0 - 2$, we denote ℓ-th (full) shift with the abuse of notation $s_\ell[\boldsymbol{a}_0] = [\boldsymbol{0}_\ell; \boldsymbol{a}_0; \boldsymbol{0}_{n-\ell-n_0}] \in \mathbb{S}^{n-1}$, for $\ell \in \{0, \dots, n - n_0\}$. Each shift is a length-m cyclic shift of $\boldsymbol{a}_0$, truncated to a length-n window without removing any entries from $\boldsymbol{a}_0$.

[6] Here, negative curvature means that the Hessian exhibits negative eigenvalues, such that the function can be decreased by following the negative eigenvector direction.

[7] It is sufficient for $|\mathcal{I}| = \mathcal{O}(\theta n_0)$, where θ is the probability that any entry of $\boldsymbol{x}_0$ is nonzero [KZLW19].

– a signed-shift of $\boldsymbol{a}_0$. Finding a good initialization is also deceptively simple: since $\boldsymbol{x}_0$ is sparse, *any length-n_0 truncation of the observation $\boldsymbol{y}$ is itself approximately a superposition of a few shifts of $\boldsymbol{a}_0$*,

$$\boldsymbol{y} \;=\; \sum_{\ell \in \operatorname{supp}(\boldsymbol{x}_0)} (x_0)_\ell \cdot \mathrm{s}_\ell \left[\boldsymbol{a}_0\right]. \tag{3.4}$$

Therefore, if we simply chose n_0 consecutive entries of $\boldsymbol{y}$, (e.g. $[y_i, y_{i+1}, \dots, y_{i+n_0-1}],\ i \in [m - n_0]$) randomly from the observation $\boldsymbol{y}$ and initialize $\boldsymbol{a}_0$ by setting

$$\boldsymbol{a}^{(0)} \;=\; \mathcal{P}_{\mathbb{S}^{n-1}} \left(\left[\mathbf{0}_{n_0-1}\ ;\ y_i, y_{i+1}, \dots, y_{i+n_0-1}\ ;\ \mathbf{0}_{n_0-1} \right] \right), \tag{3.5}$$

then $\boldsymbol{a}^{(0)} \in \mathbb{R}^n$ is close to a subsphere $\mathcal{S}_{\mathcal{I}}$ spanned by roughly $\mathcal{O}(n_0\theta)$ shifts of $\boldsymbol{a}_0$. Moreover, any truncation effects are absorbed by the zero-padding in Equation (3.5). In [KZLW19], this initialization scheme is improved and made rigorous, and interested readers may refer to Appendix B.3 for details.

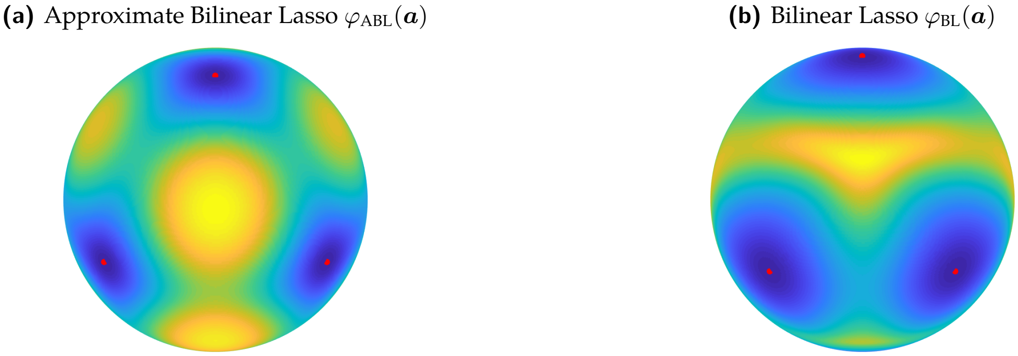


Figure 5: Approximate Bilinear Lasso vs. Bilinear Lasso: Given an incoherent truth kernel $\boldsymbol{a}_0 \sim \mathcal{U}\left(\mathbb{S}^{n_0-1}\right)$, we plot the heat maps of objective landscapes of (**a**) the Approximate Bilinear Lasso and (**b**) Bilinear Lasso losses, restricted to the subsphere spanned by $\boldsymbol{a}_0,\ \mathrm{s}_1\left[\boldsymbol{a}_0\right]$, and $\mathrm{s}_2\left[\boldsymbol{a}_0\right]$, shown as red dots on the heat map. The curvature properties of both objective landscapes are empirically similar at key locations, e.g., near and between shifts.

Optimization over the sphere. For both the Bilinear Lasso and ABL, a unit-norm constraint on $\boldsymbol{a}$ is enforced to break the scaling symmetry between $\boldsymbol{a}_0$ and $\boldsymbol{x}_0$. Choosing the ℓ_2-norm, however, has surprisingly strong implications for optimization. The ABL objective, for example, is piecewise concave whenever $\boldsymbol{a}$ is sufficiently far away from any shift of $\boldsymbol{a}_0$, but the sphere induces positive curvature near individual shifts to create strong convexity. These two properties combine to ensure recoverability of $\boldsymbol{a}_0$. In contrast, enforcing ℓ_1-norm constraints often leads to spurious minimizers for deconvolution problems [LWDF11b, BVG13, ZLK+17].

Implications for the Bilinear Lasso. The ABL is an example of a formulation for SaSD posessing a (regionally) benign optimization landscape, which guarantees that efficient recovery is possible when $\boldsymbol{a}_0$ is incoherent. Applications of sparse deconvolution, however, are often motivated by sharpening or resolution tasks [HBZ09, CFG14, CE16] where the motif $\boldsymbol{a}_0$ is smooth and coherent ($\mu_s(\boldsymbol{a}_0)$ is large). The ABL objective is a poor approximation of the Bilinear Lasso in such cases and therefore fails to yield practical algorithms.

In such cases, the Bilinear Lasso should be optimized directly, and Figure 5 shows that its loss surface does indeed share similar symmetry breaking properties with the ABL objective. In the next section, we apply the geometric intuition gained from the ABL formulation, in combination with a number of computational heuristics, to create an optimization method for SaSD that performs well in general problem instances.

4 Designing Practical Nonconvex Optimization Algorithms

Several algorithms for SaSD type problems have been developed for specific applications, such as image deblurring [LWDF11b, BDH+13, CE16], neuroscience [RPQ15, FZP17, SFB18], and image super-resolution

[BK02, SGG+09, YWHM10]. In this section, however, we will instead leverage the intuition from Section 3 and build optimization methods for the Bilinear Lasso

$$\min_{\boldsymbol{a},\,\boldsymbol{x}} \;\; \Psi_{\mathrm{BL}}(\boldsymbol{a},\boldsymbol{x}) \;\; \doteq \;\; \underbrace{\tfrac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{a}\circledast\boldsymbol{x}\right\|_2^2}_{\text{smooth }\psi(\boldsymbol{a},\boldsymbol{x})} + \underbrace{\lambda\cdot\left\|\boldsymbol{x}\right\|_1}_{\text{nonsmooth }g(\boldsymbol{x})}\;, \quad \text{s.t.} \quad \boldsymbol{a}\in\mathbb{S}^{n-1}, \tag{4.1}$$

that perform well in general settings for SaSD, as the Bilinear Lasso more accurately accounts for interactions between $\boldsymbol{a}\circledast\boldsymbol{x}$ when $\boldsymbol{a}_0$ is shift-coherent. In such situations, optmization of Ψ_{BL} will also suffer from slow convergence and poor resolution of $\boldsymbol{x}_0$, which we will address in this section with a number of heuristics. This leads to an efficient and practical algorithms for solving sparse deconvolution problems.

4.1 Solving the Bilinear Lasso via alternating descent method

Efficient global optimization of the nonconvex objective in Equation (4.1) is a nontrivial task, largely due to the existence of spurious local minima and saddle points. In the following, we introduce a simple first-order method dealing with these issues. As suggested by our discussion of the geometry of the Dropped Quadratic in Section 3, we avoid such spurious minimizers using a data-driven initialization scheme introduced in Section 3.2. On the other hand, our study in Section 3 implies that all saddle points exhibit large negative curvature and can hence be effectively escaped by first-order methods[8] [LPP+17, JGN+17, GBW18].

Starting from the data-driven initialization, we optimize the Bilinear Lasso using a first-order *alternating descent method* (ADM). The basic idea of our ADM algorithm is to alternate between taking first-order descent steps on $\Psi(\boldsymbol{a},\boldsymbol{x})$ w.r.t. one variable while the other is fixed:

Fix $\boldsymbol{a}$ and take a descent step on $\boldsymbol{x}$. At each iteration k, with fixed $\boldsymbol{a}^{(k)}$, ADM first descends the objective $\Psi_{\mathrm{BL}}(\boldsymbol{a},\boldsymbol{x})$ by taking a *proximal gradient* step w.r.t. $\boldsymbol{x}$ with an appropriate stepsize t_k

$$\boldsymbol{x}^{(k+1)} \;\leftarrow\; \operatorname{prox}_g^{\lambda t_k}\left(\boldsymbol{x}^{(k)} - t_k\cdot\nabla_{\boldsymbol{x}}\,\psi\left(\boldsymbol{a}^{(k)},\boldsymbol{x}^{(k)}\right)\right), \tag{4.2}$$

where $\operatorname{prox}_g(\cdot)$ denotes the proximal operator of $g(\cdot)$ [Nes13a]. Since the subproblem of minimizing $\Psi_{\mathrm{BL}}(\boldsymbol{a},\boldsymbol{x})$ only w.r.t. $\boldsymbol{x}$ is the Lasso problem, the proximal step taken in Equation (4.2) here is classical[9] [BT09, PB+14].

Fix $\boldsymbol{x}$ and take a descent step on $\boldsymbol{a}$. Next, we fix the iterate $\boldsymbol{x}^{(k+1)}$ and we take a *Riemannian gradient* step [AMS09] w.r.t. $\boldsymbol{a}$ over the sphere $\mathbb{S}^{n-1}$, with stepsize $\tau_k > 0$,

$$\boldsymbol{a}^{(k+1)} \;\leftarrow\; \mathcal{P}_{\mathbb{S}^{n-1}}\left(\boldsymbol{a}^{(k)} - \tau_k\cdot\operatorname{grad}_{\boldsymbol{a}}\psi\left(\boldsymbol{a}^{(k)},\boldsymbol{x}^{(k+1)}\right)\right), \tag{4.3}$$

where $\operatorname{grad}_{\boldsymbol{a}}\psi(\boldsymbol{a},\boldsymbol{x})$ denotes the Riemannian gradient of $\psi(\boldsymbol{a},\boldsymbol{x})$ w.r.t. $\boldsymbol{a}$, and $\mathcal{P}_{\mathbb{S}^{n-1}}(\cdot)$ is a projection operator onto the sphere $\mathbb{S}^{n-1}$. The Riemannian gradient $\operatorname{grad}_{\boldsymbol{a}}\psi(\boldsymbol{a},\boldsymbol{x})$ can be interpreted as the standard gradient projected to the (Euclidean) tangent space[10] of $\mathbb{S}^{n-1}$ at point $\boldsymbol{a}$, and the projection operator $\mathcal{P}_{\mathbb{S}^{n-1}}(\cdot)$ ensures that our iterate stays on the sphere[11].

ADM simply alternates between steps of Equation (4.2) and Equation (4.3) until convergence, and can seamlessly incorporate other acceleration techniques that we will discuss in the later part of this section. We refer readers to Appendix B.1 for more implementation details.

The geometric intuition gained in Section 3 is based on the marginalized objective $\varphi_{\mathrm{BL}}(\boldsymbol{a})$ over the sphere $\mathbb{S}^{n-1}$, whereas here we simply a descent step on $\Psi_{\mathrm{BL}}(\boldsymbol{a},\boldsymbol{x})$ w.r.t. $\boldsymbol{x}$ rather than minimize $\boldsymbol{x}$ explicitly to reduce

[8] In [ZLK+17] and [KZLW19], they employed second-order trust-region [CGT00, BAC18] and curvilinear search [Gol80, GMWZ17] methods for solving SaSD. Although second-order methods can also escape strict saddle points by directly exploiting the Hessian, they are much more expensive computationally and hence not practical for large datasets.

[9] The Equation (4.2) can also be rewritten and interpreted as $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - t_k\mathcal{G}_{t_k}\left(\boldsymbol{x}^{(k)}\right)$ with the *composite gradient mapping* $\mathcal{G}_{t_k}$ [Nes13a]. $\mathcal{G}_{t_k}$ behaves like the "gradient" on the smooth Moreau envelope of $\Psi_{\mathrm{BL}}(\boldsymbol{a},\boldsymbol{x})$, as a function of $\boldsymbol{x}$.

[10] The tangent space is a $n-1$ dimensional Euclidean linear space, containing all the tangent vectors at $\boldsymbol{a}\in\mathbb{S}^{n-1}$. We refer the readers to [AMS09, Section 3] for more concrete definitions.

[11] The Riemannian gradient step is a specific *manifold retraction* operator on the sphere, which takes a point from the tangent space at some point $\boldsymbol{a}$ and pushes it to a new point on the manifold. We refer interested readers to Section 3 of [AMS09] for more details.

computational complexity per iteration. Nonetheless, the sequence of gradients $\nabla_{\boldsymbol{a}} \Psi_{\mathrm{BL}}(\boldsymbol{a}^{(k)}, \boldsymbol{x}^{(k)})$ on $\boldsymbol{a}$ approximates $\nabla \varphi_{\mathrm{BL}}(\boldsymbol{a}^{(k)})$ as $k \to \infty$, since ADM is guaranteed to converge to some stationary point [BST14, PS16]. Therefore, ADM on $\Psi_{\mathrm{BL}}(\boldsymbol{a}, \boldsymbol{x})$ eventually becomes equivalent to Riemannian gradient descent on $\varphi_{\mathrm{BL}}(\boldsymbol{a})$.

4.2 Heuristics for improving the geometry of Bilinear Lasso

Although the Bilinear Lasso is able to account for the interactions between $\boldsymbol{a}_0$ and $\boldsymbol{x}_0$ under high coherence, the smooth term $\|\boldsymbol{a} \circledast \boldsymbol{x} - \boldsymbol{y}\|_2^2$ nonetheless becomes ill-conditioned as $\mu(\boldsymbol{a}_0)$ increases, leading to slow convergence for practical problem instances. Here we will discuss a number of heuristics which will help to obtain faster algorithmic convergence and produce better solutions in such settings.

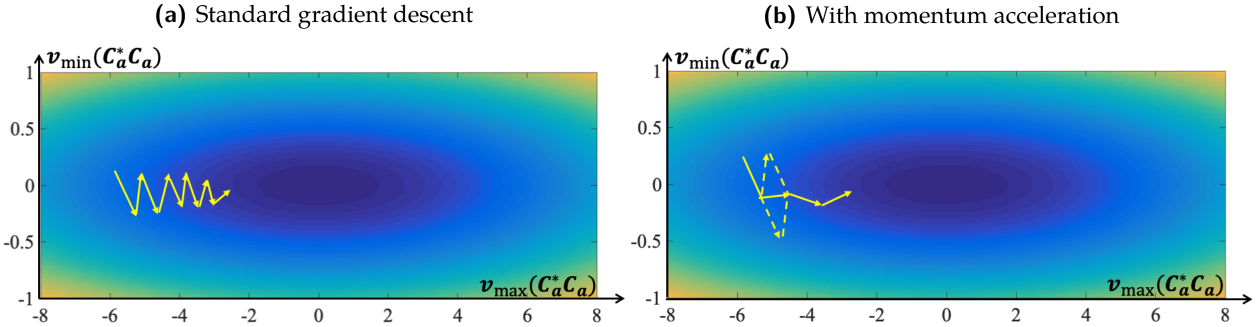


Figure 6: Momentum acceleration. The left figure shows the behavior of standard gradient descent which oscillates on functions of ill-conditioned Hessian; the right figure shows that by incorporating the previous steps the momentum acceleration alleviates the oscillation effects and achieves faster convergence.

4.2.1 Accelerating first-order descent under high coherence

When $\mu_s(\boldsymbol{a}_0)$ is large, the Hessian of Ψ_{BL} becomes ill-conditioned as $\boldsymbol{a}$ converges to single shifts. the objective landscape contains "narrow valleys" in which first-order methods tend to exhibit severe oscillations (Figure 6a) [Nes13b]. For a nonconvex problem such as the Bilinear Lasso, iterates of first-order methods could encounter many narrow and flat valleys along the descent trajectory, resulting in slow convergence.

One remedy here is to add *momentum* [Pol64, BT09] to standard first-order iterations. For example, when updating $\boldsymbol{x}$, we could modify the iterate in Equation (4.2) by

$$\boldsymbol{w}^{(k)} = \boldsymbol{x}^{(k)} + \beta \cdot \underbrace{\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right)}_{\text{inertial term}}, \tag{4.4}$$

$$\boldsymbol{x}^{(k+1)} = \operatorname{prox}_{t_k g}\left(\boldsymbol{w}^{(k)} - t_k \nabla_{\boldsymbol{x}} \psi\left(\boldsymbol{a}^{(k)}, \boldsymbol{w}^{(k)}\right)\right). \tag{4.5}$$

Here, the *inertial term* incorporates the momentum from previous iterations, and $\beta \in (0,1)$ controls the inertia[12]. In a similar fashion, we can modify the iterate [AMS09] for updating[13] $\boldsymbol{a}$ in Equation (4.3). We term the new algorithm *inertial alternating descent method* (iADM), and we refer readers to Appendix B.1.2 for more details.

As illustrated in Figure 6b, the additional inertial term improves convergence by substantially reducing oscillation effects for ill-conditioned problems. The acceleration of momentum methods for convex problems are well-known in practice[14]. Recently, momentum methods has also been proven to improve convergence for nonconvex and nonsmooth problems [PS16, JNJ18].

[12] Setting $\beta = 0$ here removes momentum and reverts to standard proximal gradient descent.

[13] It modifies iPALM [PS16] to perform updates on $\boldsymbol{a}$ via retraction on the sphere.

[14] In the setting of strongly convex and smooth function $f(\boldsymbol{z})$, the momentum method improves the iteration complexity from $\mathcal{O}(\kappa \log(1/\varepsilon))$ to $\mathcal{O}(\sqrt{\kappa} \log(1/\varepsilon))$ with κ being the condition number, while leaving the computational complexity approximately unchanged [B+15].

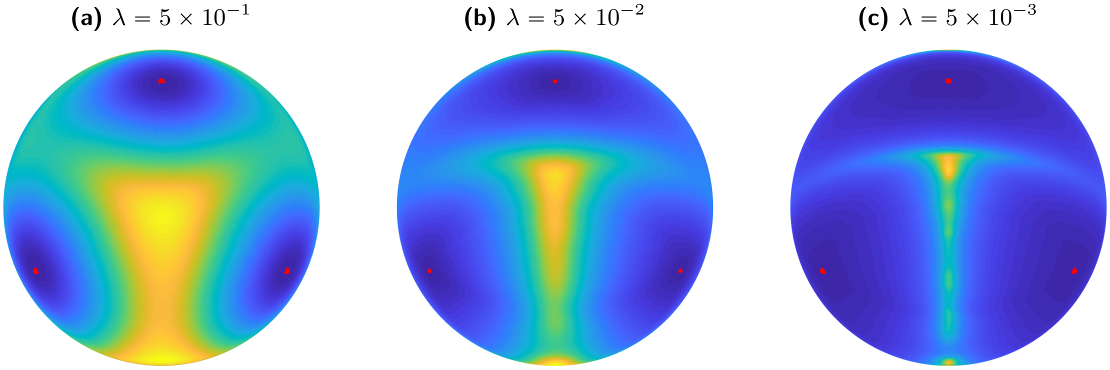

(a) $\lambda = 5 \times 10^{-1}$ **(b)** $\lambda = 5 \times 10^{-2}$ **(c)** $\lambda = 5 \times 10^{-3}$

Figure 7: Low-dimensional functional landscape of Bilinear Lasso with varying λ. Each subfigure shows the objective $\varphi_{\mathrm{BL}}(\boldsymbol{a})$, with $\boldsymbol{a}$ restricted to the subsphere $\mathcal{S}_{\{0,1,2\}}$ defined in Equation (3.3), with varying choices of $\lambda > 0$. The kernel $\boldsymbol{a}_0$ is incoherent and drawn uniformly from the sphere. The red dots denote the location of the kernel and its shifts.

4.2.2 A practical method for SaSD based on homotopy continuation

It is also possible to improve optimization by modifying the objective Ψ_{BL} directly through the sparsity penalty λ. Variations of this idea appear in both [ZLK+17] and [KZLW19], and can also help to mitigate the effects of large shift-coherence in practical problems.

When solving (3.1) in the noise-free case, it is clear that larger choices of λ encourage sparser solutions for $\boldsymbol{x}$. Conversely, smaller choices of λ place local minimizers of the marginal objective $\varphi_{\mathrm{BL}}(\boldsymbol{a}) \doteq \min_{\boldsymbol{x}} \Psi_{\mathrm{BL}}(\boldsymbol{a}, \boldsymbol{x})$ closer to signed-shifts of $\boldsymbol{a}_0$ by emphasizing reconstruction quality. When $\mu(\boldsymbol{a}_0)$ is large, however, φ_{BL} becomes ill-conditioned as $\lambda \to 0$ due to the poor spectral conditioning of $\boldsymbol{a}_0$, leading to severe flatness near local minimizers (Figure 7) and the creation spurious local minimizers when noise is present. At the expense of precision, larger values of λ limit $\boldsymbol{x}$ to a small set of support patterns and simplify the landscape ofφ_{BL}. It is therefore important both for fast convergence and accurate recovery for λ to be chosen appropriately.

When problem parameters – such as the severity of noise, or p_0 and θ – are not known a priori, a *homotopy continuation method* [HYZ08, WNF09, XZ13] can be used to obtain a *range* of solutions for SaSD. Using the initialization (3.5), a rough estimate $(\widehat{\boldsymbol{a}}^{(1)}, \widehat{\boldsymbol{x}}^{(1)})$ is first obtained by solving (3.1) with iADM using a large choice for $\lambda^{(1)}$; this estimate is refined by gradually decreasing $\lambda^{(n)}$ to produce the *solution path* $\left\{(\widehat{\boldsymbol{a}}^{(n)}, \widehat{\boldsymbol{x}}^{(n)}; \lambda^{(n)})\right\}$. By ensuring that $\boldsymbol{x}$ remains sparse along the solution path, homotopy provides the objective Ψ_{BL} with (restricted) strong convexity w.r.t. both $\boldsymbol{a}$ and $\boldsymbol{x}$ throughout optimization [ANW10]. As a result, homotopy achieves linear convergence for SaSD where sublinear convergence is expected otherwise (Figures 13 and 14).

Algorithm for SaSD. We summarize our discussion by presenting a practical algorithm for solving SaSD (Algorithm 1), which initializes $\boldsymbol{a}$ using Equation (3.5) and subsequently find a local minimizer of the Bilinear Lasso using homotopy continuation, combined with the accelerated first-order iADM method, with an appropriate choice of λ. However, we note should be possible to substitute iADM with any first or second-order descent method (e.g. the Riemannian trust-region method [ABG07, CSL+18]). We compare some of these different choices in Section 5.2.

For Algorithm 1, we usually set $\beta = 0.9$ to incorporate sufficient momentum for iADM; setting β too large, however, can cause iADM to diverge. The stepsizes t_k and τ_k in iADM are obtained by backtracking (linesearch) [NW06, PS16]. We often set the initial penalty $\lambda_0 = \left\| \boldsymbol{C}^*_{\iota_{n\to m}\widehat{\boldsymbol{a}}^{(0)}} \boldsymbol{y} \right\|_\infty$ large enough to ensure sparse $\boldsymbol{x}$, and choose $\lambda_\star$ based on problem dimension and noise level (often $\lambda_\star = 0.1/\sqrt{n}$ is good choice). Typically, a good choice is to set the decaying parameter $\eta = 0.9$ and the precision factor $\delta = 0.1$. We refer readers to Appendices for more implementation details.

4.3 Extension for convolutional dictionary learning

The optimization methods we introduced for SaSD here can be naturally extended to tackle sparse blind deconvolution problems with multiple unknown kernels/motifs (a.k.a. convolutional dictionary learning

Algorithm 1 Solving SaSD with homotopy continuation

```
Input: Measurement $\boldsymbol{y} \in \mathbb{R}^m$; momentum parameter $\beta \in [0,1)$; initial and final sparse penalties $\lambda_0$, $\lambda_\star$ ($\lambda_0 > \lambda_\star$); decay penalty parameter $\eta \in (0,1)$; precision factor $\delta \in (0,1)$ and tolerance $\varepsilon_\star$.
Output: final solution $(\boldsymbol{a}_\star, \boldsymbol{x}_\star)$.
  Set iteration number $K \leftarrow \lfloor \log(\lambda_\star/\lambda_0) \,/\, \log \eta \rfloor$.
  Initialize $\widehat{\boldsymbol{a}}^{(0)} \in \mathbb{R}^n$ using Equation (3.5), $\widehat{\boldsymbol{x}}^{(0)} = \mathbf{0}_m$, and $\lambda^{(0)} = \lambda_0$, $\varepsilon^{(0)} = \delta\lambda^{(0)}$;
  for $k = 1, \dots, K$ do
    Solve
      $$\min_{\boldsymbol{a} \in \mathbb{S}^{n-1}, \boldsymbol{x}} \Psi_{\lambda^{(k-1)}}(\boldsymbol{a}, \boldsymbol{x}) \doteq \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{a} \circledast \boldsymbol{x}\right\|_2^2 + \lambda^{(k-1)}\left\|\boldsymbol{x}\right\|_1$$
    to precision $\varepsilon^{(k-1)} = \delta\lambda^{(k-1)}$ via iADM, using $\left(\widehat{\boldsymbol{a}}^{(k-1)}, \widehat{\boldsymbol{x}}^{(k-1)}\right)$ as warm start
      $$\left(\widehat{\boldsymbol{a}}^{(k)}, \widehat{\boldsymbol{x}}^{(k)}\right) \leftarrow \text{iADM}\left(\, \widehat{\boldsymbol{a}}^{(k-1)}, \widehat{\boldsymbol{x}}^{(k-1)}; \boldsymbol{y}, \lambda^{(k-1)}, \beta \,\right).$$
    Update $\lambda^{(k)} \leftarrow \eta\lambda^{(k-1)}$.
  end for
  Final round: starting from $\left(\widehat{\boldsymbol{a}}^{(K)}, \widehat{\boldsymbol{x}}^{(K)}\right)$, optimize $\Psi_{\lambda_\star}(\boldsymbol{a}, \boldsymbol{x})$ with penalty $\lambda_\star$ to precision $\varepsilon_\star$ via
      $$\left(\widehat{\boldsymbol{a}}_\star, \widehat{\boldsymbol{x}}_\star\right) \leftarrow \text{iADM}\left(\, \widehat{\boldsymbol{a}}^{(K)}, \widehat{\boldsymbol{x}}^{(K)}; \boldsymbol{y}, \lambda_\star, \beta \,\right).$$
```

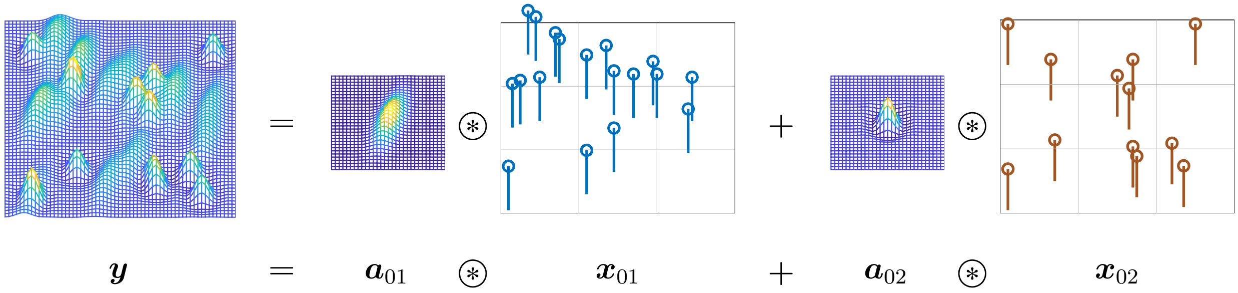


Figure 8: Convolutional dictionary learning. Simultaneous recovery for multiple unknown kernels $\{\boldsymbol{a}_{0k}\}_{k=1}^N$ and sparse activation maps $\{\boldsymbol{x}_{0k}\}_{k=1}^N$ from $\boldsymbol{y} = \sum_{k=1}^N \boldsymbol{a}_{0k} \circledast \boldsymbol{x}_{0k}$.

[CF17, GCW18]), which have broad applications in microscopy data analysis [YHV17, ZCB$^+$14, CSL$^+$18] and neural spike sorting [ETS11, RPQ15, SFB18]. As illustrated in Figure 8, the new observation $\boldsymbol{y}$ in this task is the sum of N convolutions between short kernels $\{\boldsymbol{a}_{0k}\}_{k=1}^N$ and sparse maps $\{\boldsymbol{x}_{0k}\}_{k=1}^N$,

$$\boldsymbol{y} = \sum_{k=1}^N \boldsymbol{a}_{0k} \circledast \boldsymbol{x}_{0k}, \qquad \boldsymbol{a}_{0k} \in \mathbb{R}^{n_0}, \quad \boldsymbol{x}_{0k} \in \mathbb{R}^m, \quad (1 \leqslant k \leqslant N).$$

The natural extension of SaSD, then, is to recover $\{\boldsymbol{a}_{0k}\}_{k=1}^N$ and $\{\boldsymbol{x}_{0k}\}_{k=1}^N$ up to signed, shift, and permutation ambiguities, leading to the SaS convolutional dictionary learning (SaS-CDL) problem. The SaSD problem can be seen as a special case of SaS-CDL with $N = 1$. Based on the Bilinear Lasso formulation in Equation (4.1) for solving SaSD, we constrain all kernels $\boldsymbol{a}_{0k}$ over the sphere, and consider the following nonconvex objective:

$$\min_{\{\boldsymbol{a}_k\}_{k=1}^N,\, \{\boldsymbol{x}_k\}_{k=1}^N} \frac{1}{2}\left\|\boldsymbol{y} - \sum_{k=1}^N \boldsymbol{a}_k \circledast \boldsymbol{x}_k\right\|_2^2 + \lambda \sum_{k=1}^N \left\|\boldsymbol{x}_k\right\|_1, \quad \text{s.t.} \quad \boldsymbol{a}_k \in \mathbb{S}^{n-1} \quad (1 \leqslant k \leqslant N). \tag{4.6}$$

When the kernels $\{\boldsymbol{a}_{0k}\}_{k=1}^N$ are incoherent enough to each other, we anticipate that all local minima are near signed shifts of the ground truth. Similar to the idea of solving the Bilinear Lasso in Equation (4.1), we optimize Equation (4.6) via ADM and its variants, by taking alternating descent steps on $\{\boldsymbol{a}_k\}_{k=1}^N$ and $\{\boldsymbol{x}_k\}_{k=1}^N$ with one fixed. We refer readers to Appendix B and Appendix C for more technical details.

4.4 Additional modifications for practical settings

Here briefly summarize some prevalent issues that appear in with real datasets and how the our SaS model and associated deconvolution method can be adjusted to deal with these additional challenges.

- **Linear vs. cyclic convolution.** In this work, we follow the convention of [KZLW19] and mainly discuss SaSD in the context of cyclic convolution. The linear convolution, however, is a better model for many practical SaSD tasks (e.g. involving natural images or time series). Despite this, there is no loss of generality as any statements about cyclic convolution can easily be carried over to linear convolution; by zero-padding $\boldsymbol{x}$ appropriately, one can always rewrite a linear convolution as a cyclic convolution. This is also convenient practically as convolution operations should be implemented using Fast Fourier transform techniques (which map directly to cyclic convolution) to reduce computational complexity for each iteration.

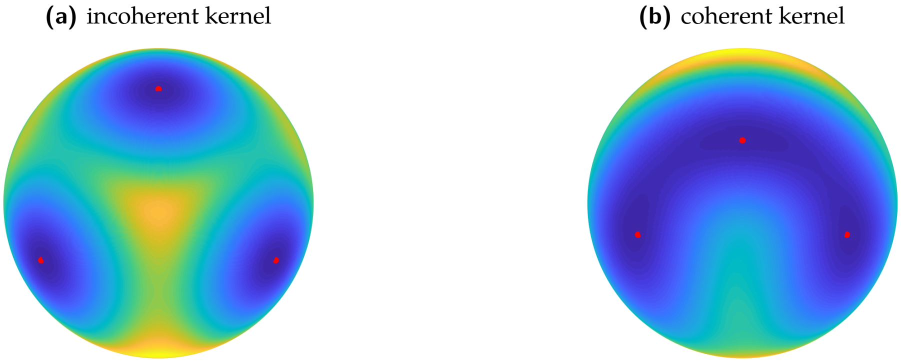
- **Resolution of $\boldsymbol{x}_0$ under noise.** We introduce a reweighting technique [CWB08] to deal with noisy datasets. The method adaptively sets large penalty on small entries of $\boldsymbol{x}$ to suppress noisy small entries, and set small penalty on large entries to promote sparse solutions of $\boldsymbol{x}$. We refer readers to Appendix B.3 for more algorithmic details.

- **Dealing with extra data structure.** In many problems such as calcium imaging [PSG+16] and spike sorting [SFB18], the sparse spike train $\boldsymbol{x}_0$ is usually nonnegative. As we shall see in Section 5, by enforcing nonnegative constraint on $\boldsymbol{x}$ for ADM, it often enables recovery of denser $\boldsymbol{x}_0$. Additionally, measurement in practice often contains unknown low frequency DC component $\boldsymbol{b}$, such that $\boldsymbol{y} = \boldsymbol{a}_0 \circledast \boldsymbol{x}_0 + \boldsymbol{b}$. We add an extra minimization in ADM to deal with b. We refer readers to Appendix B.3 for more technical details.

5 Synthetic Experiments

In this section, we experimentally demonstrate several core ideas presented in this work on both incoherent and coherent kernels. Incoherent kernels are randomly drawn by $\boldsymbol{a}_0 \sim \mathrm{Uniform}(\mathbb{S}^{n_0-1})$, which leads to $\mu_s(\boldsymbol{a}_0) \in \mathcal{O}\left(\sqrt{\frac{\log n_0}{n_0}}\right)$ diminishing w.r.t. dimension n_0. Coherent kernels are descretized from the Gaussian window function $\boldsymbol{a}_0 = \boldsymbol{g}_{n_0,0.5}$, where $\boldsymbol{g}_{n_0,\sigma} \doteq \mathcal{P}_{\mathbb{S}^{n_0-1}}\left(\left[\exp\left(-\frac{(2i-n_0-1)^2}{\sigma^2(n_0-1)^2}\right)\right]_{i=1}^{n_0}\right)$; in this case $\mu_s(\boldsymbol{a}_0) \to 1$ as n_0 grows. This allows us to illustrate some of the difficulties of optimization encountered by the Bilinear Lasso under high coherence, as well as the effectiveness of heuristics proposed in Section 4 for alleviating these difficulties.

5.1 Recovery of true kernel under coherence



(a) incoherent kernel

(b) coherent kernel

Figure 9: Incoherent vs. coherent kernels. The subfigures from left to right present the optimization landscape of $\varphi_{\mathrm{BL}}(\boldsymbol{a})$ w.r.t. $\boldsymbol{a} \in \mathbb{S}^{n-1}$ defined in (3.2), restricted to a subspace spanned by three shifts of $\boldsymbol{a}_0$. The left figure shows the landscape of incoherent kernel, and the right one presents that of the coherent kernel. The red dots denote the location of the shifts of ground truth $\boldsymbol{a}_0$.

Low-dimensional plots of function landscapes. As $\mu_s(\boldsymbol{a}_0)$ increases, the shifts of $\boldsymbol{a}_0$ lie closer together on the sphere. We show how this affects the optimization landscape of the Bilinear Lasso $\varphi_{\text{BL}}(\boldsymbol{a})$ over $\boldsymbol{a} \in \mathbb{S}^{n-1}$ by plotting the objective restricted in the subsphere spanned by three shifts[15] of $\boldsymbol{a}_0 \in \mathbb{S}^{n_0-1}$ with $n_0 = 20$, $m = 2 \times 10^3$, $\theta = {n_0}^{-3/4}$, and $\lambda = 0.5$. From Figure 9, we see that φ_{BL} exhibits clear symmetry breaking structure between the shifts of $\boldsymbol{a}_0$ in the incoherent case. As $\mu_s(\boldsymbol{a}_0)$ increases, however, adjacent shifts of $\boldsymbol{a}_0$ lie close together and symmetry breaking becomes more difficult. Practically speaking, recovering a precise shift of $\boldsymbol{a}_0$ becomes less important when recovering smooth, highly coherent kernels. Nonetheless Figure 9 suggests that the target minimizers of φ_{BL} become non-discretized in these cases.

Recovery performance. Next, we corroborate our observation of sparsity-coherence tradeoff by comparing recovery performance for incoherent vs. coherent kernels. We fix $m = 100n_0$, and plot the probability for successful recovery, which occurs if

$$\min_{\ell \in [2n_0]} \left\{1 - \left|\left\langle \boldsymbol{a}_0, \boldsymbol{\iota}^*_{n_0 \to n} s_\ell\left[\boldsymbol{a}_\star\right]\right\rangle\right|\right\} \leqslant 10^{-2},$$

w.r.t. dimension n_0 and sparsity level θ. For each (n_0, θ), we randomly generate ten independent instances of the data $\boldsymbol{y} = \boldsymbol{a}_0 \circledast \boldsymbol{x}_0$. Here $\boldsymbol{a}_\star$ denotes the optimal solution produced by minimizing Ψ_{BL} with $\lambda = 10^{-2}/\sqrt{\theta n_0}$.

From Figure 10, we see that successful recovery is likely when sparsity θ is sufficiently small compared to n_0 in general. Furthermore, recovering $\boldsymbol{a}_0$ in the coherent setting is noticably more difficult than the incoherent setting, and typically requires lower sparsity rates θ. Finally, enforcing extra structure such as nonnegativity in appropriate settings enables recovery with denser of $\boldsymbol{x}_0$ (Figures 10a and 10c).

5.2 Demonstration of data-driven initialization and homotopy acceleration

Our next experiments study the effectiveness of the data-driven (DD) initialization from Equation (3.5) and the heuristics introduced in Section 4, namely momentum acceleration and homotopy. Throughout this subsection, we set the kernel length $n_0 = 100$ and the number of samples $m = 10^4$. We generate the data $\boldsymbol{y} = \boldsymbol{a}_0 \circledast \boldsymbol{x}_0 + b\mathbf{1}_m$ with both coherent and incoherent $\boldsymbol{a}_0$, $\boldsymbol{x}_0 \sim \mathcal{BR}(\theta)$ with sparsity level $\theta = {n_0}^{-3/4}$, and b is a constant unkown bias. No noise is added. We stop each algorithm either when the preset maximum iteration is reached, or when differences between two consecutive iterates (in ℓ_2 norm) is smaller than threshold 10^{-6}.

Effectiveness of data-driven initialization. We compare the ADM and iADM methods using the data-driven initialization Equation (3.5) vs. uniform random initializations for $\boldsymbol{a}$. From Figures 11 and 12, we see that both methods converge faster to solutions of higher quality with data-driven initialization, as a result of $\boldsymbol{a}^{(0)}$ being initialized near the superposition of a few shifts of $\boldsymbol{a}_0$.

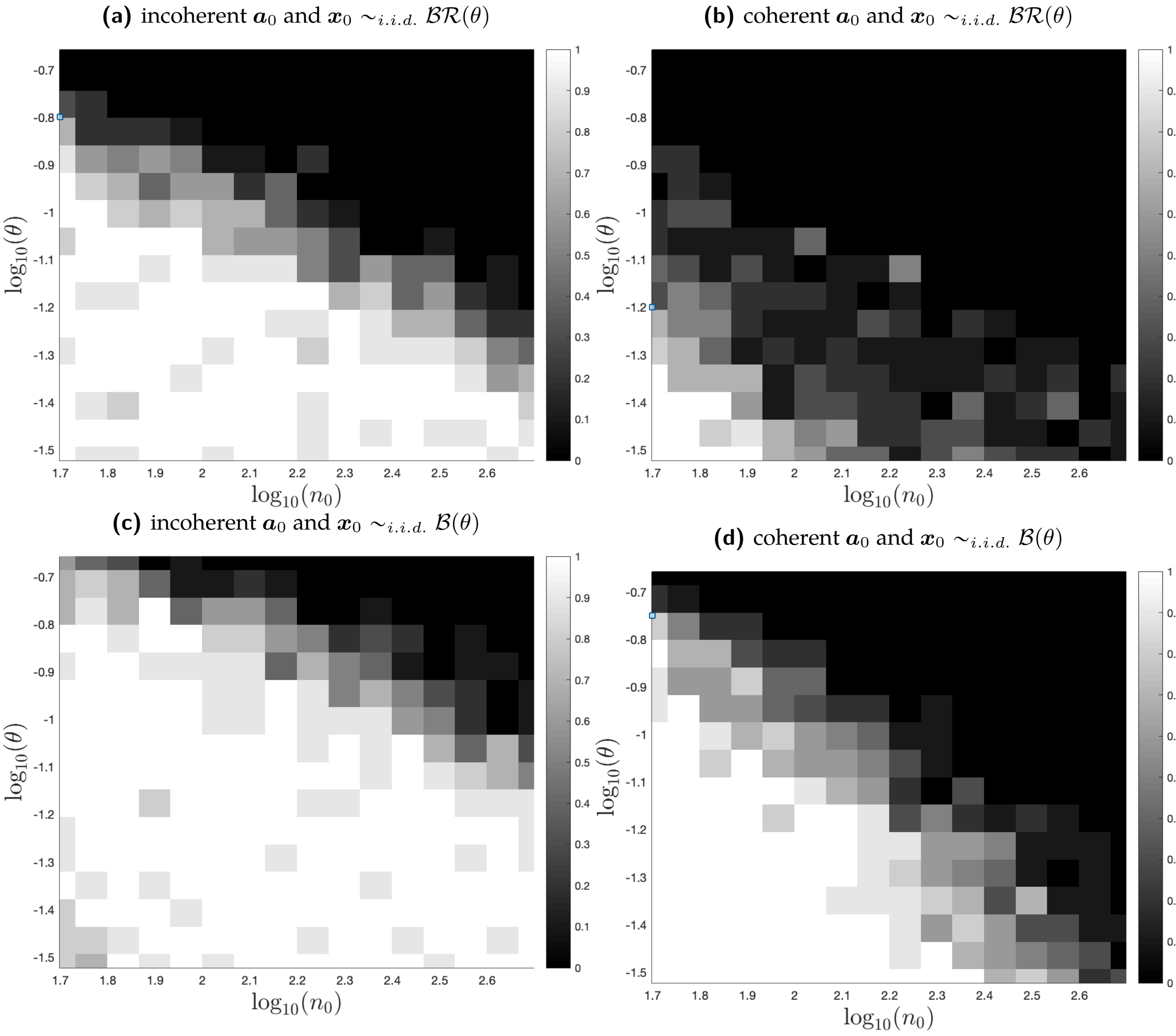
Convergence with acceleration and homotopy. Next we compare the convergence speeds of the ADM, with and without momentum (iADM) and homotopy continuation. We use Equation (3.5) to initialize $\boldsymbol{a}$, and $\boldsymbol{x}$ is initialized as zero. From Figures 13 and 14, we see applying acceleration and homotopy leads to in significant improvements over vanilla ADM in terms of convergence rate, especially when $\boldsymbol{a}_0$ is coherent.

5.3 Comparison with existing methods

Finally, we compare iADM, and iADM with homotopy, against a number of existing methods for minimizing φ_{BL}. The first is *alternating minimization* [KZLW19], which at each iteration k minimizes $\boldsymbol{a}^{(k)}$ with $\boldsymbol{x}^{(k)}$ fixed using accelerated (Riemannian) gradient descent with backtracking, and vice versa. The next method is the popular *alternating direction method of multipliers* (ADMM) [BPC+11]. Finally, we compare against iPALM [PS16] with backtracking, using the unit ball constraint on $\boldsymbol{a}_0$ instead of the unit sphere.

For each method, we deconvolve signals with $n_0 = 50, m = 100$, and $\theta = n_0^{-3/4}$ for both coherent and incoherent $\boldsymbol{a}_0$. For both iADM, iADM with homotopy, and iPALM we set $\alpha = 0.3$. For homotopy, we set $\lambda^{(1)} = \max_\ell |\langle s_\ell[\boldsymbol{a}^{(0)}], \boldsymbol{y}\rangle|$, $\lambda^\star = \frac{0.3}{\sqrt{n_0}\lambda}$, and $\delta = 0.5$. Furthermore we set $\eta = 0.5$ or $\eta = 0.8$ and for ADMM,

[15] For incoherent kernel, we generate the kernel $\boldsymbol{a}_0$ with the last two entries zero, and consider the subspace spanned by $\boldsymbol{a}_0, \mathrm{s}_1\left[\boldsymbol{a}_0\right], \mathrm{s}_2\left[\boldsymbol{a}_0\right]$. For the coherent kernel, we consider the subspace spanned by $\boldsymbol{a}_0, \mathrm{s}_{\lceil n_0/3\rceil}\left[\boldsymbol{a}_0\right], \mathrm{s}_{\lceil 2n_0/3\rceil}\left[\boldsymbol{a}_0\right]$.



(a) incoherent $\boldsymbol{a}_0$ and $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{BR}(\theta)$

(b) coherent $\boldsymbol{a}_0$ and $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{BR}(\theta)$

(c) incoherent $\boldsymbol{a}_0$ and $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{B}(\theta)$

(d) coherent $\boldsymbol{a}_0$ and $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{B}(\theta)$

Figure 10: phase transitions for solving SaS-BD: (a) shows the case when $\boldsymbol{a}_0$ is incoherent, and $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{BR}(\theta)$; (b) shows the case when $\boldsymbol{a}_0$ is coherent, and $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{BR}(\theta)$; (c) shows the case when $\boldsymbol{a}_0$ is incoherent, and $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{B}(\theta)$; (d) shows the case when $\boldsymbol{a}_0$ is coherent, and $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{B}(\theta)$. For signal $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{B}(\theta)$, positivity constraint is enforced. For each subfigure, brighter means higher probability of successful recovery, while darker means higher probability of failure.

we set the slack parameter to $\rho = 0.7$ or $\rho = 0.5$ for incoherent and coherent $\boldsymbol{a}_0$ respectively. From Figure 15, we can see that ADMM performs better than iADM in the incoherent case, but becomes less reliable in the coherent case. In both cases, iADM with homotopy is the best performer. Finally, we observe roughly equal performance between iPALM and iADM.

6 Experiments for Real Applications

In this section, we demonstrate experimentally the effectiveness of the proposed methods for both SaSD and SaS-CDL on a wide variety of applications in computational imaging and neuroscience. Our goal here is not necessarily to outperform state of the art methods, which are often tailored to specific applications. Rather, we hope to provide evidence that the intuition and heuristics highlighted in Sections 3 and 4 are widely applicable, and can serve as a robust starting point for tackling SaS problems broadly in areas of imaging science.

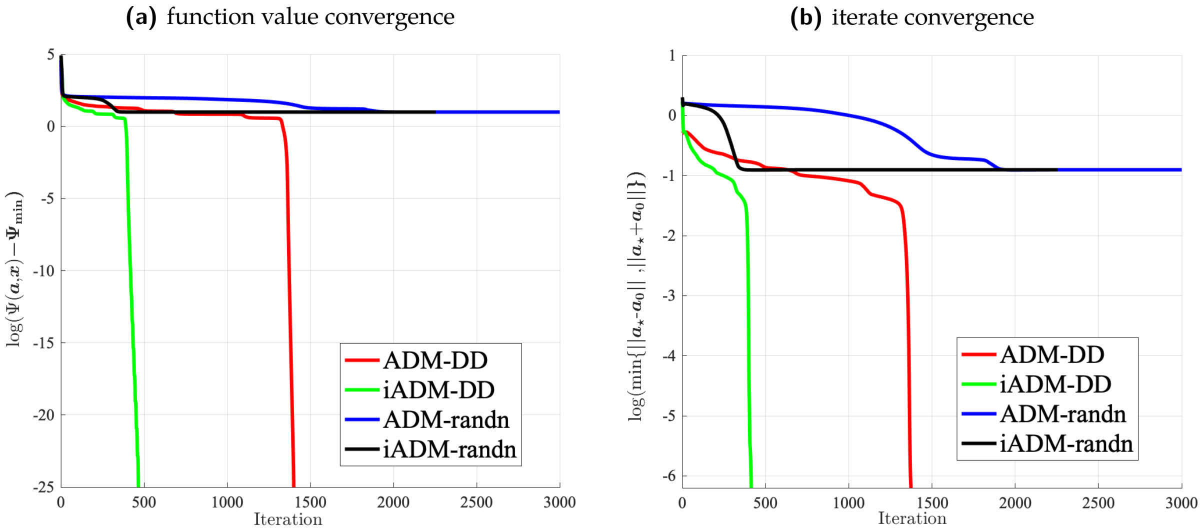


Figure 11: Comparison of initialization methods for solving SaS-BD on incoherent random kernel $\boldsymbol{a}_0$: (a) shows the function value $\Psi_{\mathrm{BL}}(\boldsymbol{a},\boldsymbol{x})$ convergence; (b) shows the iterate convergence on $\boldsymbol{a}$, where $\boldsymbol{a}_\star$ denotes a shift correction of each iterate $\boldsymbol{a}$. Here, ADM-DD and iADM-DD denote the ADM and iADM methods using data-driven initialization, and ADM-randn and iADM-randn denote the ADM and iADM methods using initializations drawn uniformly random from the sphere $\mathbb{S}^{n_0-1}$.

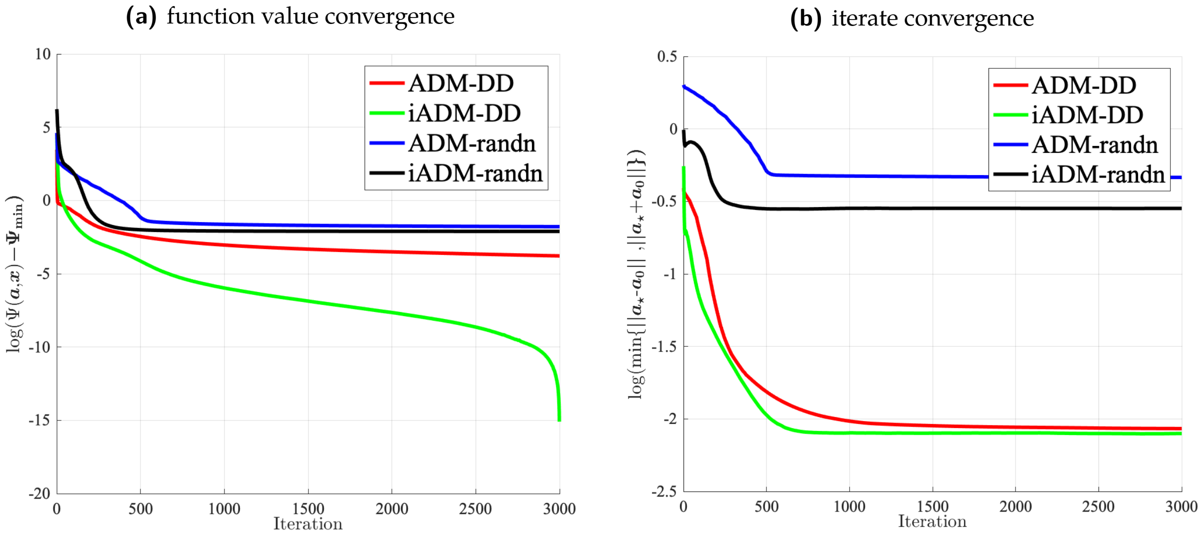
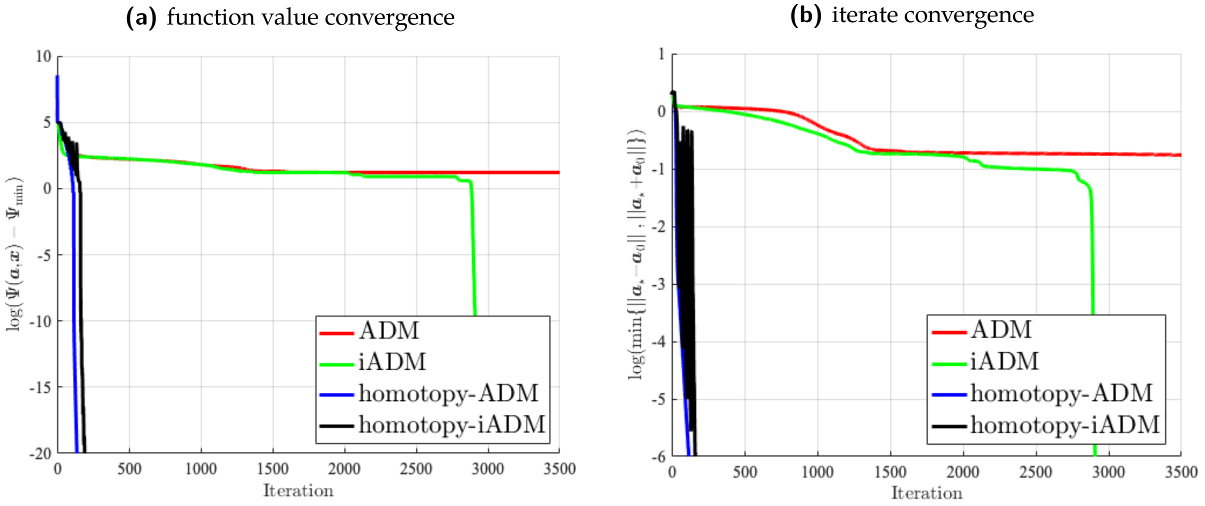


Figure 12: Comparison of initialization methods for solving SaS-BD on coherent smooth Gaussian kernel $\boldsymbol{a}_0$: (a) shows the function value $\Psi_{\mathrm{BL}}(\boldsymbol{a},\boldsymbol{x})$ convergence; (b) shows the iterate convergence on $\boldsymbol{a}$, where $\boldsymbol{a}_\star$ denotes a shift correction of each iterate $\boldsymbol{a}$. Here, ADM-DD and iADM-DD denote the ADM and iADM methods using data-driven initialization, and ADM-randn and iADM-randn denote the ADM and iADM methods using initializations drawn uniformly random from the sphere $\mathbb{S}^{n_0-1}$.

6.1 Sparse deconvolution of time sequences in neuroscience

6.1.1 Sparse deconvolution of calcium imaging

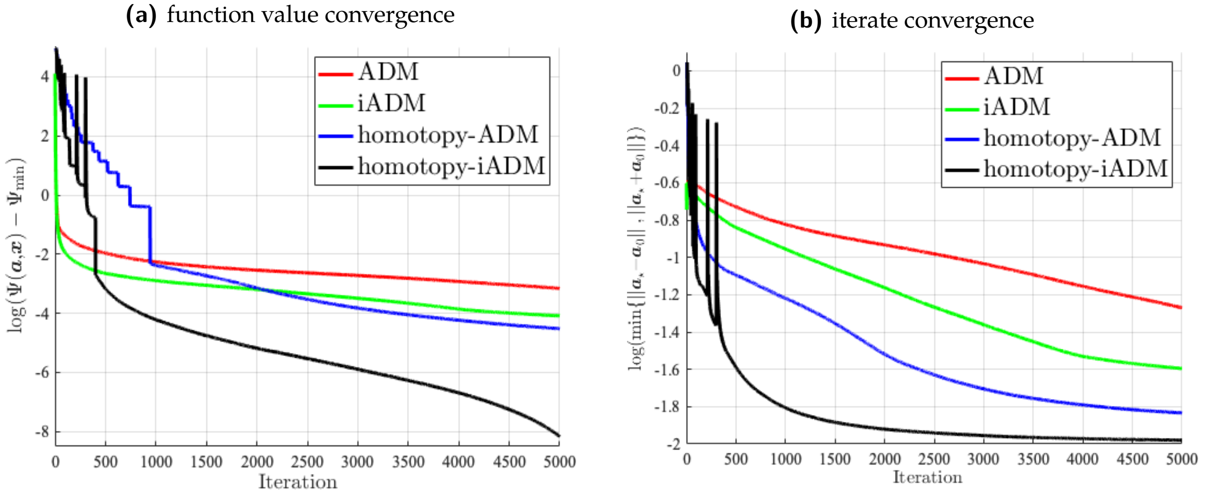
It is well known that neurons process and transmit information via discrete spiking activity. Whenever a neuron fires, it produces a transient change in chemical concentrations in the immediate environment. Transients in calcium (Ca^{2+}) concentration, for example, can be measured using calcium fluoresence imaging. The resulting fluoresence signal can be modeled as the convolution between the short transient response $\boldsymbol{a}_0$ and the



(a) function value convergence

(b) iterate convergence

Figure 13: Comparison of algorithm convergence for solving SaS-BD on incoherent random kernel $\boldsymbol{a}_0$: (a) shows the function value $\Psi_{\mathrm{BL}}(\boldsymbol{a}, \boldsymbol{x})$ convergence; (b) shows the iterate convergence on $\boldsymbol{a}$, where $\boldsymbol{a}_\star$ denotes a shift correction of each iterate $\boldsymbol{a}$. The algorithms we compared here are ADM, iADM, and its homotopy accelerations.



(a) function value convergence

(b) iterate convergence

Figure 14: Comparison of algorithm convergence for solving SaS-BD on coherent smooth Gaussian kernel $\boldsymbol{a}_0$: (a) shows the function value $\Psi_{\mathrm{BL}}(\boldsymbol{a}, \boldsymbol{x})$ convergence; (b) shows the iterate convergence on $\boldsymbol{a}$, where $\boldsymbol{a}_\star$ denotes a shift correction of each iterate $\boldsymbol{a}$. The algorithms we compared here are ADM, iADM, and its homotopy accelerations.

spike train in the form of nonnegative, sparse map $\boldsymbol{x}_0$,

$$\underbrace{\boldsymbol{y}}_{\text{raw fluorescence trace}} = \underbrace{\boldsymbol{a}_0}_{\text{transient response}} \circledast \underbrace{\boldsymbol{x}_0}_{\text{action potentials}} + \underbrace{b\mathbf{1}_m}_{\text{bias}} + \underbrace{\boldsymbol{n}}_{\text{noise}}, \qquad \boldsymbol{x}_0 \geqslant \mathbf{0}. \tag{6.1}$$

The task of recovering the spike train $\boldsymbol{x}_0$ from such SaS signals are frequently of interest in the neuroscience, and can naturally be cast as a SaSD problem. An advantage of this approach is its ability to estimate transient response (which is rarely known a priori) simultaneously. This is important when neurons exhibit dense bursts of spiking activity, which is an especially challenging setting for deconvolution tasks.

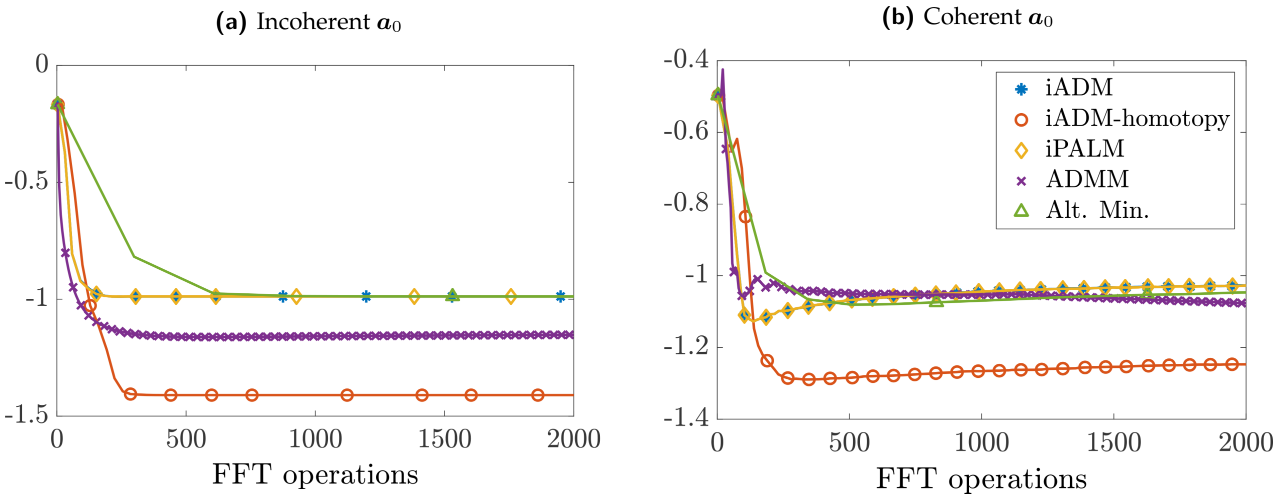


Figure 15: Algorithmic comparison. (a) Convergence of various methods minimizing Ψ_{BL} with incoherent $\boldsymbol{a}_0$ over FFT operations used (for computing convolutions). The y-axis denotes the log of the angle between $\boldsymbol{a}^{(k)}$ and the nearest shift of $\boldsymbol{a}_0$, and each marker denotes five iterations. (b) Convergence for coherent $\boldsymbol{a}_0$.

Simulated data. Recent work [VPM$^+$10, PSG$^+$16, FZP17] suggests that the calcium dynamics $\boldsymbol{y}$ can be well approximated by using a autoregressive (AR) process of order r,

$$y(t) = \sum_{i=1}^{r} \gamma_i y(t-i) + x_0(t) + b + n_s(t),$$

where $x_0(t)$ is the number of spikes that the neuron fired at t-th timestep, $n_s(t)$ is noise, and $\{\gamma_i\}_{i=1}^{r}$ are autoregressive parameters. [PSG$^+$16, FZP17] showed that the AR(r) model is equivalent to Equation (6.1) with a parameterized kernel $\boldsymbol{a}_0$. The order r is chosen to be a small positive integer, usually $r = 1$ or $r = 2$. When $r = 1$, the AR(1) kernel is a one-sided exponential function

$$a_0(t) = \exp\left(-t/\tau\right), \qquad t \geqslant 0, \tag{6.2}$$

for some $\tau > 0$. The AR(1) model serves as a good approximation of the calcium dynamics when the temporal resolution of imaging sensors is low. In contrast, the AR(2) model serves as a more accurate model for high temporal resolution calcium dynamics, with

$$a_0(t) = \exp\left(-t/\tau_1\right) - \exp\left(-t/\tau_2\right), \qquad t \geqslant 0, \tag{6.3}$$

where τ_1 and τ_2 are some parameters with $\tau_1 > \tau_2 > 0$. As illustrated in Figure 16, for high temporal resolution calcium dynamics, the AR(2) model tends to be a better model which captures the short rise-time of calcium transients by the difference of two exponential functions.

Here we demonstrate the effectiveness of the proposed methods on synthetic data for both AR(1) and AR(2) models. We generate a sequence of simulated calcium dynamics $\boldsymbol{y}$ with length $T = 100(s)$ and sampling rate $f = 100Hz$ (i.e. $m = 10^4$ samples in total). We generate the kernel $\boldsymbol{a}_0 \in \mathbb{R}^{n_0}$ with length $T = 1(s)$ (i.e. $n_0 = 100$): for the AR(1) model, we set $\tau = 0.25$ in Equation (6.2); for AR(2) model, we set $\tau_1 = 0.2$ and $\tau_2 = 0.03$ in Equation (6.3). Each kernel is normalized so they lie on the sphere. The sparse spike train $\boldsymbol{x}_0$ is generated from Bernoulli distribution $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{B}(\theta)$ with sparsity rate $\theta = {n_0}^{-4/5}$. We set the bias $b = 1$ and noise $\boldsymbol{n} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2 \boldsymbol{I}\right)$, $\sigma = 5 \times 10^{-2}$ in Equation (6.1).

We test and compare the proposed iADM and its reweighted variant (see Appendix B.2) for deconvolving the data, with $\lambda = 10^{-1}$. Reweighting is especially effective under noise contamination (Section 4.4), as demonstrated by Figure 16 where it provides more accurate predictions of the unknown neuron kernels for both AR(1) and AR(2) models. From Figures 17 and 18, we can clearly see that deconvolution is more difficult under the AR(2) model. In such cases reweighting can significantly improve resolution of spiking activity, allowing accurate estimation of firing times even in under dense bursts.

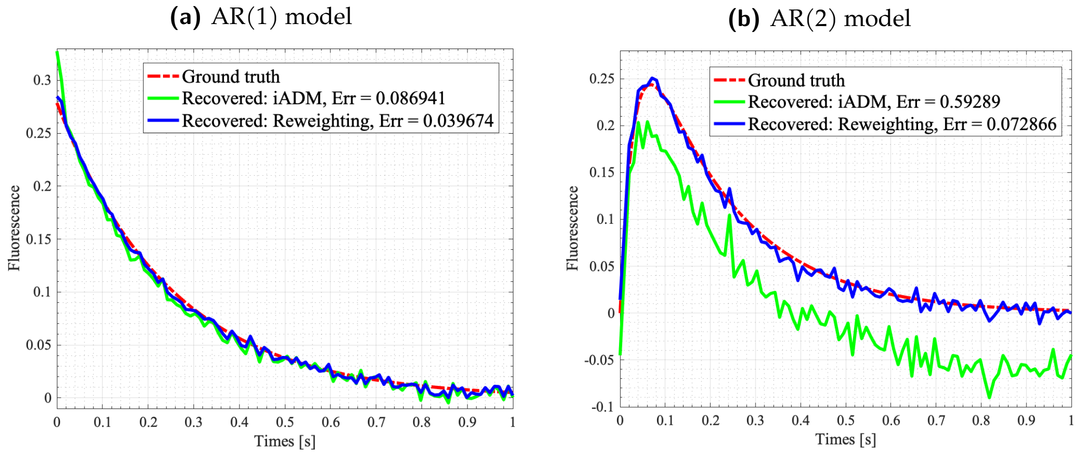
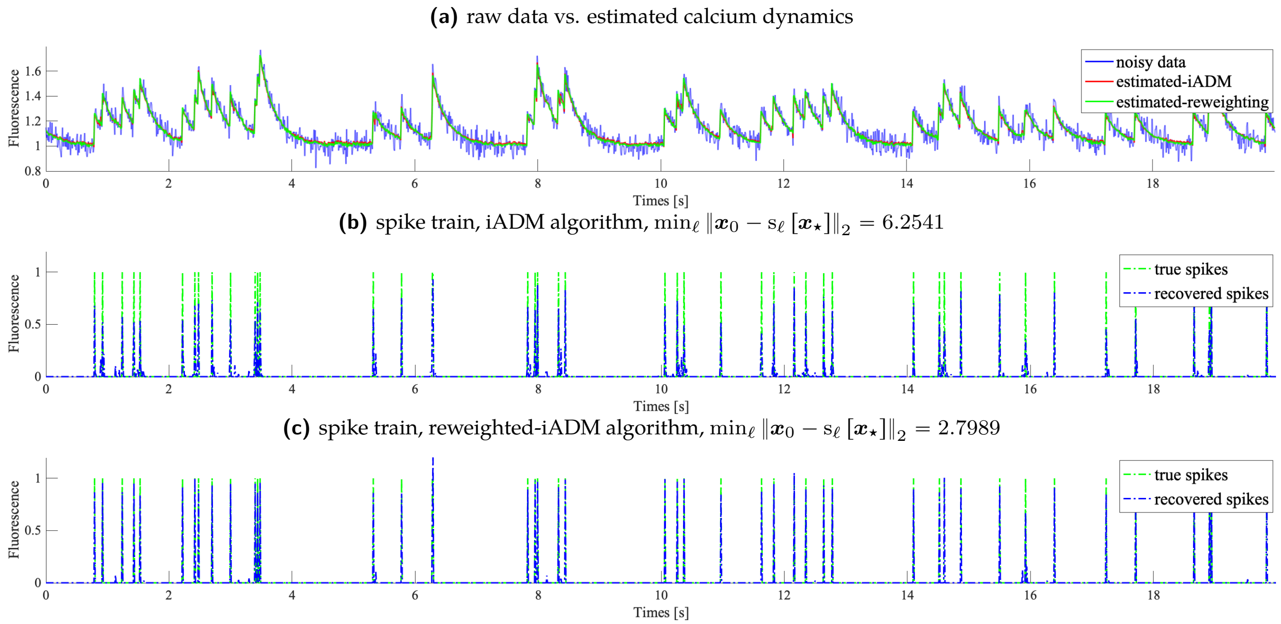


Figure 16: Recovery of transient response $\boldsymbol{a}_0$ for calcium imaging. The left figure denotes kernel $\boldsymbol{a}_0$ for the AR(1) model, and the right figure shows the kernel $\boldsymbol{a}_0$ for AR(2) model.



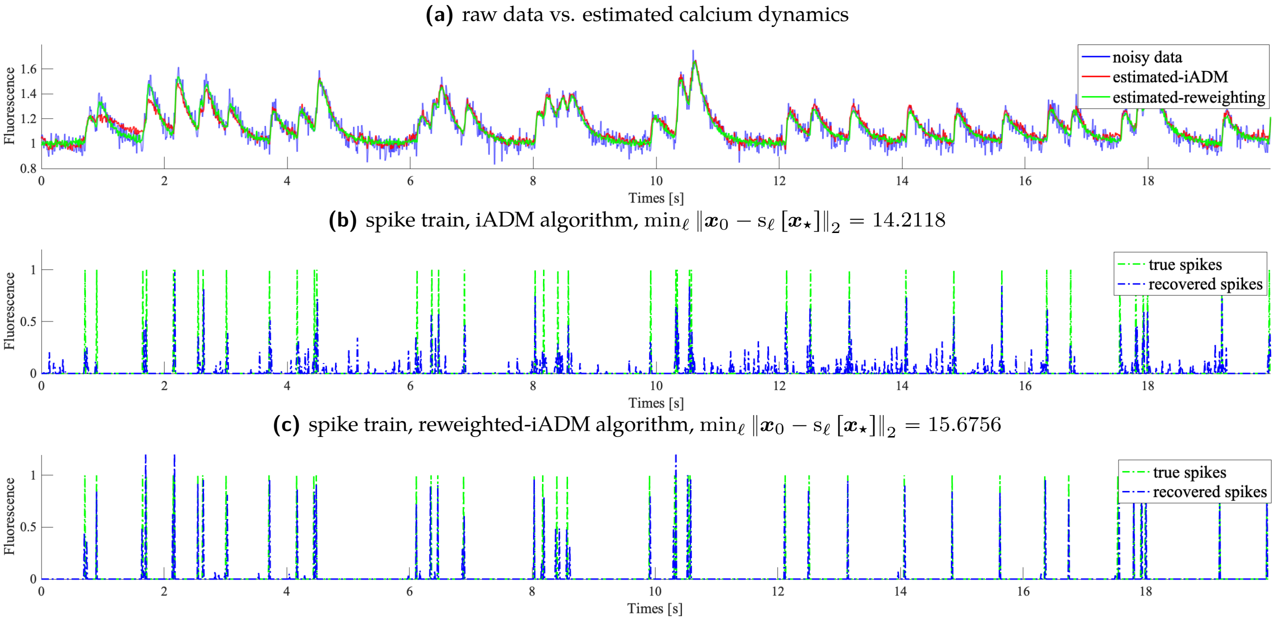
(b) spike train, iADM algorithm, $\min_\ell \left\| \boldsymbol{x}_0 - \mathrm{s}_\ell\left[\boldsymbol{x}_\star\right] \right\|_2 = 6.2541$

(c) spike train, reweighted-iADM algorithm, $\min_\ell \left\| \boldsymbol{x}_0 - \mathrm{s}_\ell\left[\boldsymbol{x}_\star\right] \right\|_2 = 2.7989$

Figure 17: Estimation of spike train $\boldsymbol{x}_0$ for AR(1) model. The first figure shows the estimation of calcium dynamics, the second figure shows the estimation of the spiking train $\boldsymbol{x}_0$ by iADM algorithm, and the third figure demonstrates the reweighting variant of iADM. $\min_\ell \left\| \boldsymbol{x}_0 - \mathrm{s}_\ell\left[\boldsymbol{x}_\star\right] \right\|_2$ denotes the distance between the target $\boldsymbol{x}_0$ and estimated solution $\boldsymbol{x}_\star$. As we observe, the proposed methods can accurately predict the spiking locations even when spikes overlap.

Real calcium imaging dataset. Finally, we demonstrate the effectiveness of proposed methods on the real calcium imaging dataset[16]. The data has been resampled to sampling rate $f = 100Hz$, and linear drifting trends are removed from calcium traces using robust regression [TBF+16]. Since these measurements are contaminated by large system noise, as is often the case in realistic settings, we choose a large sparsity penalty $\lambda = 6 \times 10^{-1}$ for Equation (3.1). Figure 19 shows the recovered kernel by the proposed iADM and its reweighting variant. Figure 20 shows the estimated spike train. By comparison, the reweighting method appears to produce better estimation of spiking activity.

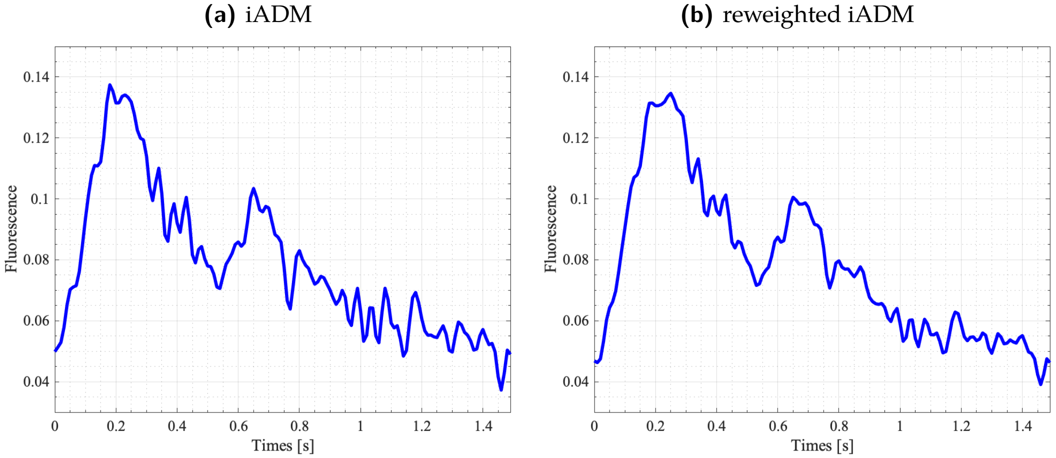
[16] The data is obtain from the spikefinder website, `http://spikefinder.codeneuro.org/`.


(a) raw data vs. estimated calcium dynamics

(b) spike train, iADM algorithm, $\min_\ell \left\| \boldsymbol{x}_0 - \mathrm{s}_\ell \left[\boldsymbol{x}_\star \right] \right\|_2 = 14.2118$

(c) spike train, reweighted-iADM algorithm, $\min_\ell \left\| \boldsymbol{x}_0 - \mathrm{s}_\ell \left[\boldsymbol{x}_\star \right] \right\|_2 = 15.6756$

Figure 18: Estimation of spike train $\boldsymbol{x}_0$ for AR(2) model. The first figure shows the estimation of calcium dynamics, the second figure shows the estimation of the spiking train $\boldsymbol{x}_0$ by iADM algorithm, and the third figure demonstrates the reweighting variant of iADM. We use $\min_\ell \left\| \boldsymbol{x}_0 - \mathrm{s}_\ell \left[\boldsymbol{x}_\star \right] \right\|_2$ to denote the distance between the target $\boldsymbol{x}_0$ and estimated solution $\boldsymbol{x}_\star$. In comparison with the original iADM algorithm, the reweighting method is very effective in suppressing noise.


(a) iADM

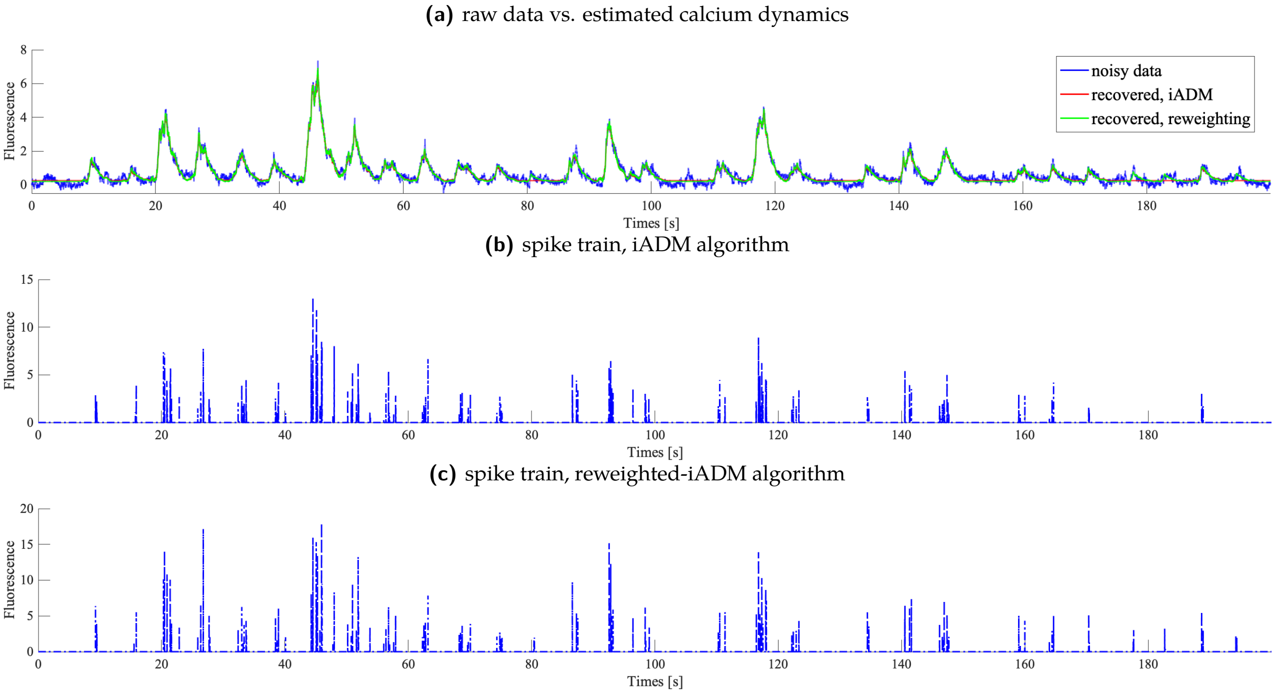
(b) reweighted iADM

Figure 19: Recovery of transient response $\boldsymbol{a}_0$ for real dataset. Left figure shows the recovered kernel by the iADM algorithm, right figure shows the recovered kernel by its reweighting variant.

6.1.2 Spike sorting by convolutional dictionary learning

Electrophysiological activity recorded by electrodes usually record superpositions of waveforms generated from multiple neurons simultaneously [RPQ15]. The goal of spike sorting is to estimate the spiking times from the measurement and decompose the spiking activities of the specific neurons. We refer interested readers to [Lew98, RPQ15] for a more detailed overview of this problem. Traditional spike sorting approaches [QNBS04, CMB$^+$17, YSE$^+$18, CRQ18] are often time consuming, lack standardization, and involving manual intervention, which makes it difficult to maintain data provenance and assess the quality of scientific results.

In the following, we introduce a fully automated approach based on SaS-CDL and nonconvex optimization; this is similar to the approach taken by [SFB18]. Mathematically, the measured waveform can be modeled as a superposition of convolutions of individual neuron waveform templates and their corresponding spike


(a) raw data vs. estimated calcium dynamics

(b) spike train, iADM algorithm

(c) spike train, reweighted-iADM algorithm

Figure 20: Estimation of spike train $\boldsymbol{x}_0$ for real calcium imaging dataset. The first figure shows the estimation of calcium dynamics, the second figure shows the estimation of the spiking train $\boldsymbol{x}_0$ by iADM algorithm, and the third figure demonstrates the reweighting variant of iADM.

trains,

$$\underbrace{\boldsymbol{y}}_{\text{voltage signal}} = \sum_{k=1}^{N} \underbrace{\boldsymbol{a}_{0k}}_{\text{waveform template}} \circledast \underbrace{\boldsymbol{x}_{0k}}_{\text{sparse spike train}} + \underbrace{b\mathbf{1}_m}_{\text{bias}} + \underbrace{\boldsymbol{n}}_{\text{noise}},$$

where each waveform templates $\{\boldsymbol{a}_{0k}\}_{k=1}^{N} \in \mathbb{R}^{n_0}$ correspond to different neurons, and therefore exhibit different kernel shapes. Given the signal $\boldsymbol{y}$, the task of spike sorting is to recover all $\{\boldsymbol{a}_{0k}\}_{k=1}^{N}$ and $\{\boldsymbol{x}_{0k}\}_{k=1}^{N}$; this is a classic example of the SaS-CDL problem as discussed in Section 4.3.

The difficulty of spike sorting (or SaS-CDL) is not only captured by the shift-coherence of the individual waveforms $\boldsymbol{a}_{0k}$ individually, but also by the shift-coherence between different waveforms from $\{\boldsymbol{a}_{0k}\}_{k=1}^{N}$. The problem increases with the cross-correlation of differing kernels. Let $\boldsymbol{A}_0 = \begin{bmatrix}\boldsymbol{a}_{01} & \cdots & \boldsymbol{a}_{0N}\end{bmatrix}$. Quantitatively, we can define *mutual incoherence* of $\boldsymbol{A}_0$ by

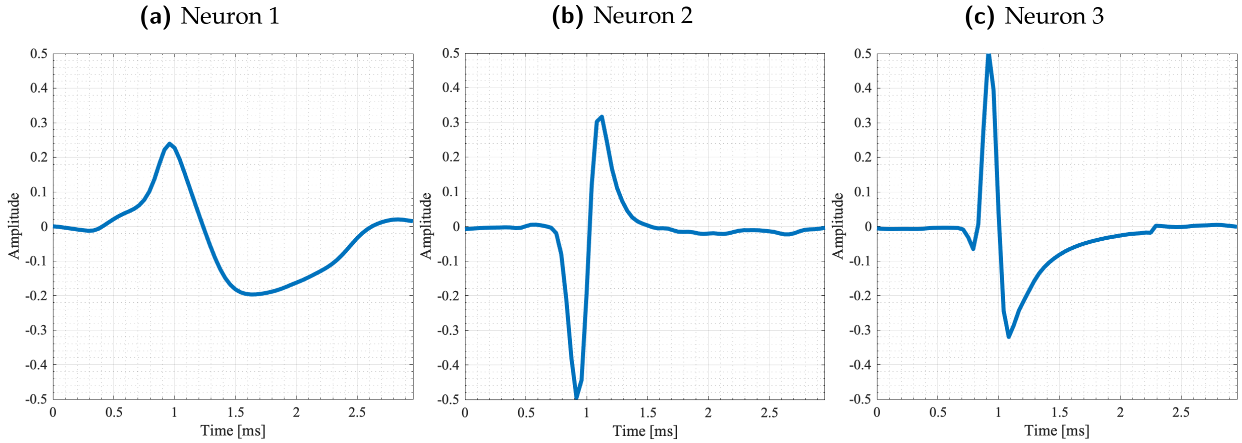
$$\mu_m\left(\boldsymbol{A}_0\right) = \max_{1 \leqslant i < j \leqslant N} \left\| \boldsymbol{C}_{\boldsymbol{a}_{0i}}^{*} \boldsymbol{a}_{0j} \right\|_{\infty},$$

which is essentially the largest shift-correlation between all kernels. The SaS-CDL problem becomes easy when $\mu_m\left(\boldsymbol{A}_0\right)$ is small, and vice versa. In the following, we demonstrate the effectiveness of the proposed methods for spike sorting on one easy dataset (with small $\mu_m\left(\boldsymbol{A}_0\right)$) and one difficult dataset (with large $\mu_m\left(\boldsymbol{A}_0\right)$).

We demonstrate the proposed reweighting variant of iADM algorithm on a classical spike-sorting dataset[17]. The signal is sampled at a frequency of $f = 24kHz$, and each time sequence records spiking activities of 3 different types of neurons. The waveform templates are constructed using a database of 594 different average spike shapes compiled from recordings in the neocortex and basal ganglia. A more detailed description of dataset can be found in Section 4 of [QNBS04]. We test the proposed method on two signal sequences of length $m = 10^5$, each measures the spiking activities of three different types of neurons with length $n_0 = 72$: one signal sequence is easy to deconvolve with low mutual coherence $\mu_m\left(\boldsymbol{A}_0\right)$, and another is relatively more

[17] It can be downloaded online at `https://vis.caltech.edu/~rodri/Wave_clus/Wave_clus_home.htm`.

difficult with larger $\mu_m(\boldsymbol{A}_0)$. The data is contaminated by random noise, with noise level 0.05 (i.e., the standard deviation relative to the amplitude of the spike classes). The recovered waveform and sparse spike train for the "easy" case are shown in Figure 21 and Figure 22, respectively. And the results for the "difficult" case are shown in Figure 21 and Figure 22. As we observe, the proposed method successfully recovers the waveform templates and spiking locations for each type of neuron. As the latter "difficult" signal sequence contains neuron waveform of similar shapes, we observe slightly more false alarms in spike detection.


(a) Neuron 1

(b) Neuron 2

(c) Neuron 3

Figure 21: Recovered neuron waveform template of "easy" dataset. The data contains three neurons of *distinct* waveforms, with noise level 0.05. Each subfigure corresponds to the recovered waveform template of one specific type of neuron.

6.2 Microscopy imaging and data analysis

Finally, we apply our proposed method towards applications in microscopy, and demonstrate its effectiveness in image super-resolution and decomposition problem settings.

6.2.1 Sparse blind deconvolution for super-resolution fluorescence microscopy

Fluorescence microscopy is a widely used imaging method in biomedical research [Hel07, FST08], and has enabled numerous breakthroughs in neuroscience [GK12], biology and biochemstry [LC11, NN14, BBM+16]. The spatial resolution of fluorescence microscopy is however limited by diffraction: the wavelength of the light (i.e., several hundred nanometers) is often larger than the typical molecular length scales in cells, preventing a detailed characterization of most subcellular structures.

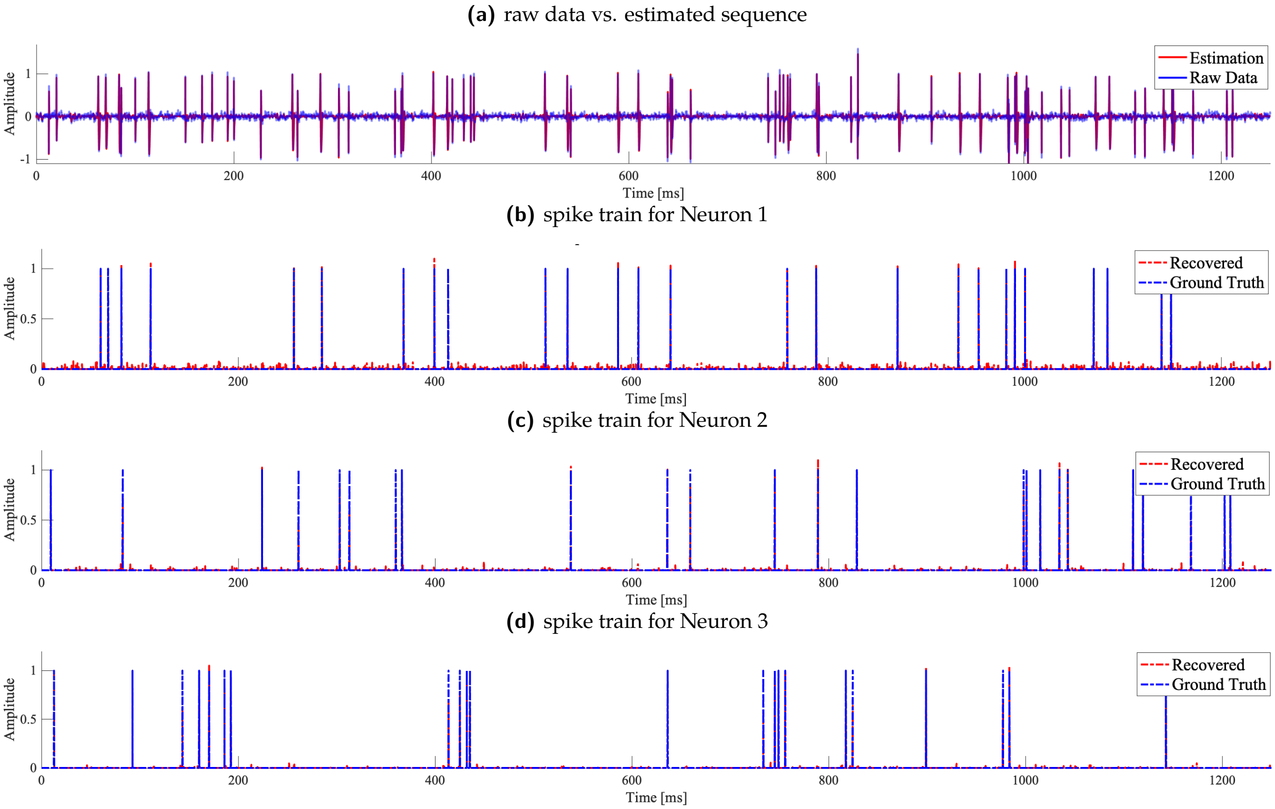
A computational imaging technique recently developed to overcome this resolution limit is *stochastic optical reconstruction microscopy*[18] (STORM) [RBZ06b, HWBZ08, HBZ10]. Instead of activating all the fluorophores at the same time, STORM randomly activates subsets of photoswitchable fluorescent probes to seperate the fluorophores present into multiple frames of sparsely activated molecules (see Figure 26 and Figure 27). From the purspective of the sparsity-coherence tradeoff, this effectively reduces the sparsity of $\boldsymbol{x}_0$, making deconvolution easier to solve. Therefore, if the location of these molecules can be precisely determined computationally for each frame, synthesizing all deconvolved frames produces a super-resolution microscopy image with near nanoscale resolutions.

For each frame, the localization task can be formulated as a sparse deconvolution problem, i.e.,

$$\underbrace{\boldsymbol{Y}}_{\text{frame}} = \underbrace{\boldsymbol{A}_0}_{\text{point spread function}} \boxed{*} \underbrace{\boldsymbol{X}_0}_{\text{sparse point sources}} + \underbrace{\boldsymbol{N}}_{\text{noise}},$$

where we want to recover $\boldsymbol{X}_0$ given $\boldsymbol{Y}$. The classical approaches solve the problem by fitting the blurred spots with Gaussian point-spread functions (PSFs) using either maximum-likelihood or Bayesian estimation

[18] Similar methods with different names have been developed at the same time by using different fluorophores and microscopes, such as photoactivated localization microscopy (PALM) [BPS+06], and fluorescence photoactivation localization microscopy (fPALM) [HGM06].

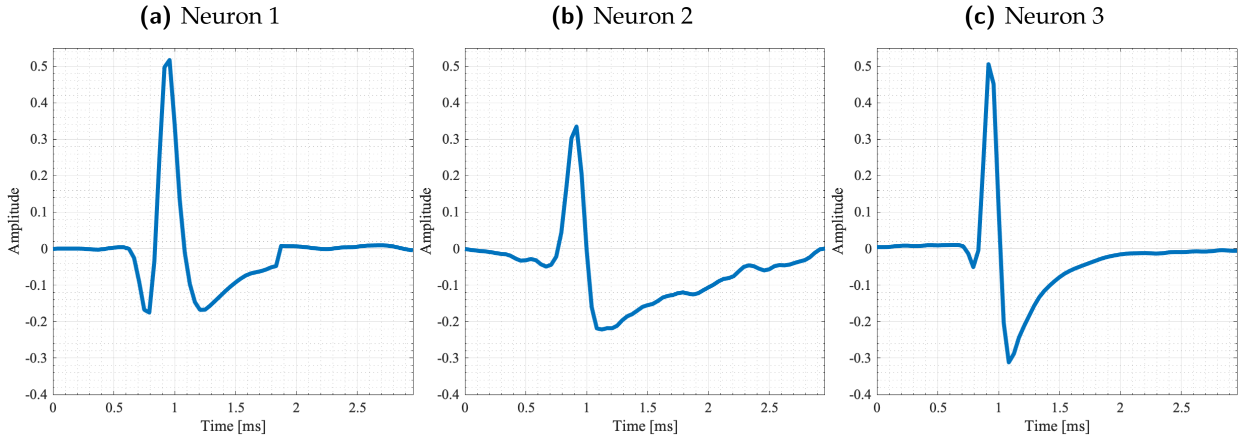

(a) raw data vs. estimated sequence

(b) spike train for Neuron 1

(c) spike train for Neuron 2

(d) spike train for Neuron 3

Figure 22: Detected spike train of "easy" dataset. The data contains three neurons of *distinct* waveforms, with noise level 0.05. The first subfigure shows the estimation of the raw data sequence. The second to fourth subfigures show the predicted spike train for each neuron, respectively.


(a) Neuron 1

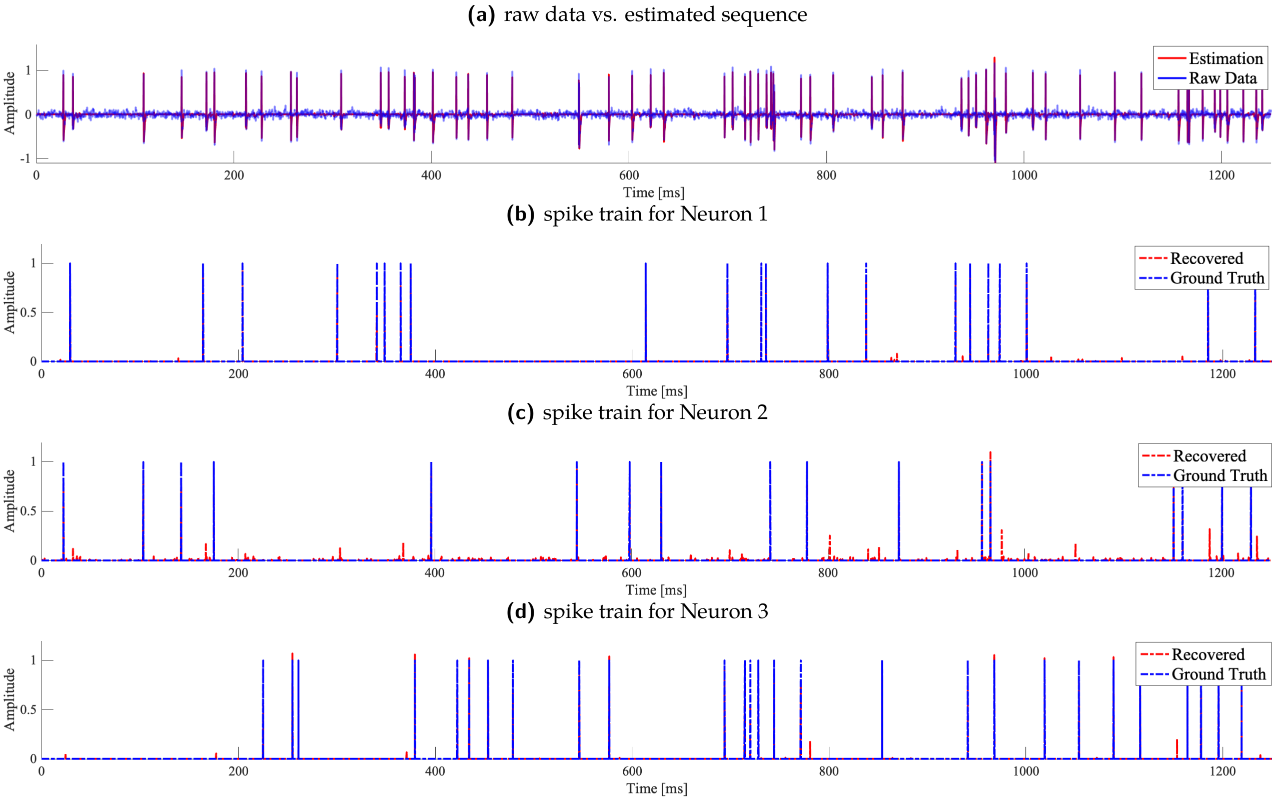
(b) Neuron 2

(c) Neuron 3

Figure 23: Recovered neuron waveform template of "difficult" dataset. The data contains three neurons of *similar* waveforms, with noise level 0.05. Each subfigure corresponds to the recovered waveform template of one specific type of neuron.

techniques [QLL^{+}10, HUK11, ZZEH12]. These approaches suffer from several limitations: (i) estimation is computationally expensive and poor in quality when dense clusters of fluorophores are activated; (ii) for 3D imaging, the PSF exhibits aberration across the focus plane [SN06], making it almost impossible to directly estimate it from the data.

To deal with these challenges, we solve the single-molecule localization problem using our proposed method for SaSD to jointly estimate the PSF $\boldsymbol{A}_0$ and the point source map $\boldsymbol{X}_0$. Our frames come from the


(a) raw data vs. estimated sequence

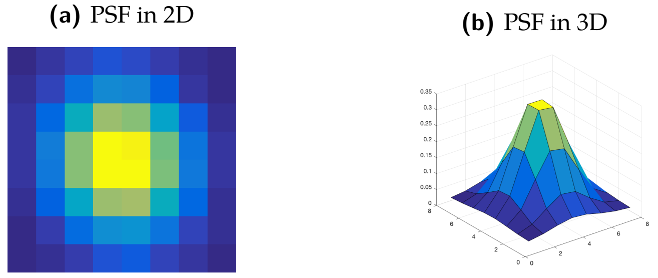
(b) spike train for Neuron 1

(c) spike train for Neuron 2

(d) spike train for Neuron 3

Figure 24: Detected spike train of "difficult" dataset. The data contains three neurons of *similar* waveforms, with noise level 0.05. The first subfigure shows the estimation of the raw data sequence. The second to fourth subfigures show the predicted spike train for each neuron, respectively.

single-molecule localization microscopy (SMLM) benchmarking dataset[19]. We apply the reweighted iADM algorithm on the 2D real video sequence "Tubulin", which contains 500 high density frames. The fluorescence wavelength is 690 nanometer (nm), the imaging frequency is $f = 25Hz$, and each frame is of size 128×128. The single-molecule localization problem is solved on the same 128×128 pixel grid[20], where each pixel is of 100 nm resolution. Figure 25 shows the recovered PSF, Figure 26 presents the recovered activation map for each individual time frame, and Figure 27 presents the aggregated super-resolution image. These results show that our approach can automatically predict the PSF and the activation map for each video frame, producing higher resolution microscopy images without manual intervention.

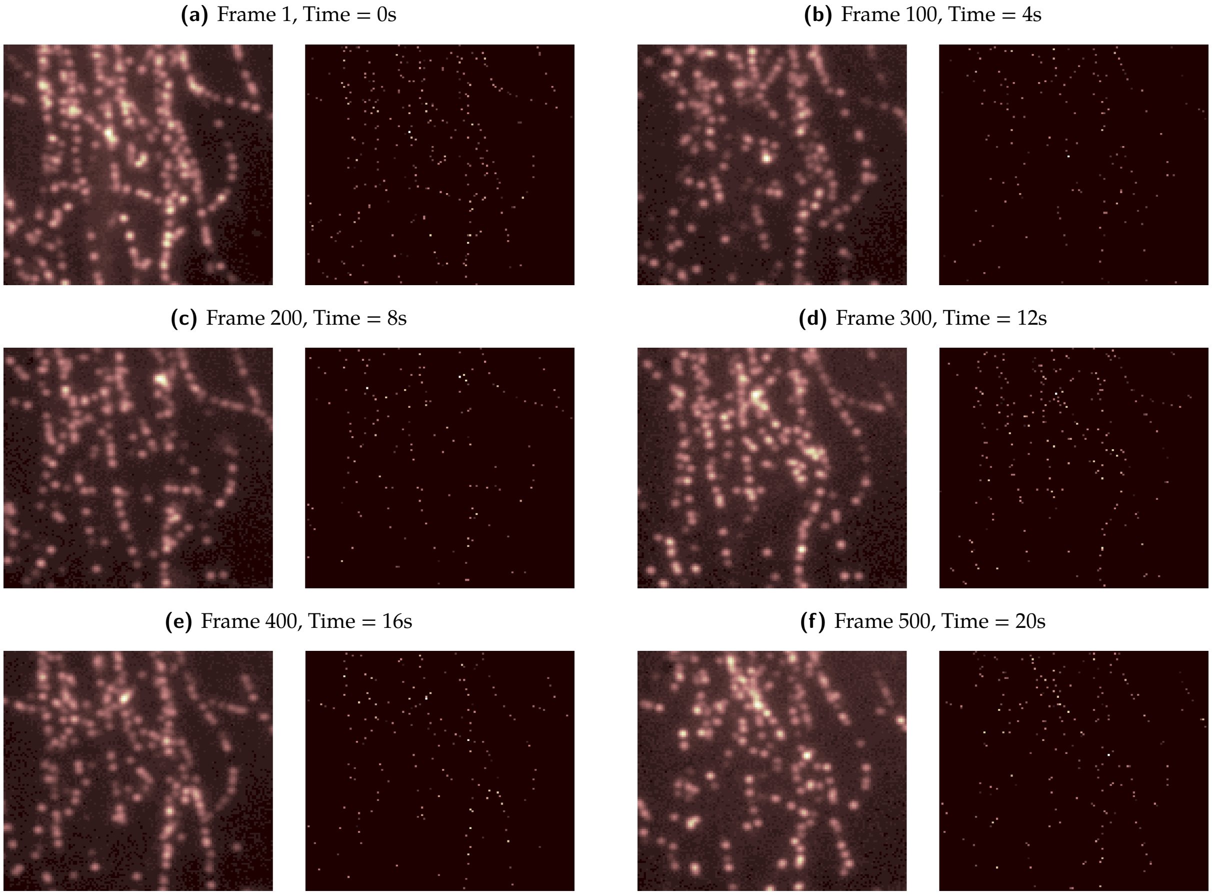

(a) PSF in 2D

(b) PSF in 3D

Figure 25: Estimated PSF for STORM imaging. The left hand side shows the estimated 8×8 PSF in 2D, the right hand side visualizes the PSF in 3D.

[19] All the data can be downloaded at http://bigwww.epfl.ch/smlm/datasets/index.html.

[20] Usually, the localization problem is solved on a finer grid (e.g., grid with $4-10$ times better resolution) so that the resulting resolution can reach $20 - 30$ nm. We will discuss potential methods to deal with this finer-grid SaSD problem in Section 7.2 as future work.


(a) Frame 1, Time = 0s

(b) Frame 100, Time = 4s

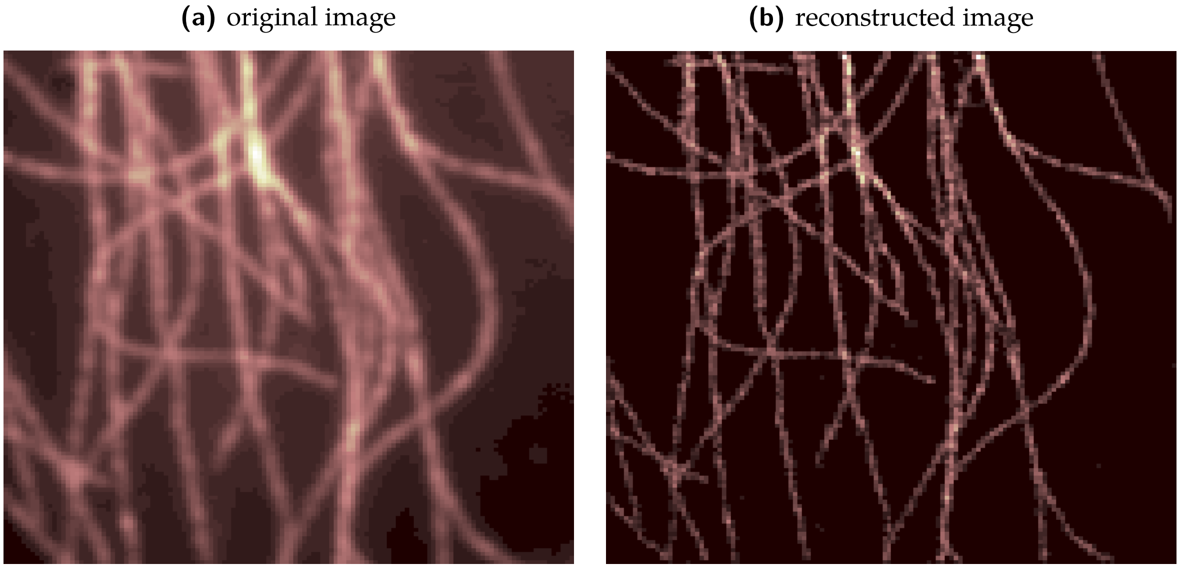
(c) Frame 200, Time = 8s

(d) Frame 300, Time = 12s

(e) Frame 400, Time = 16s

(f) Frame 500, Time = 20s

Figure 26: Predicted activation map for each individual frame. For each subfigure, the left hand side shows the original video frame, and the right hand side presents the predicted activation map using our SaSD solver.


(a) original image

(b) reconstructed image

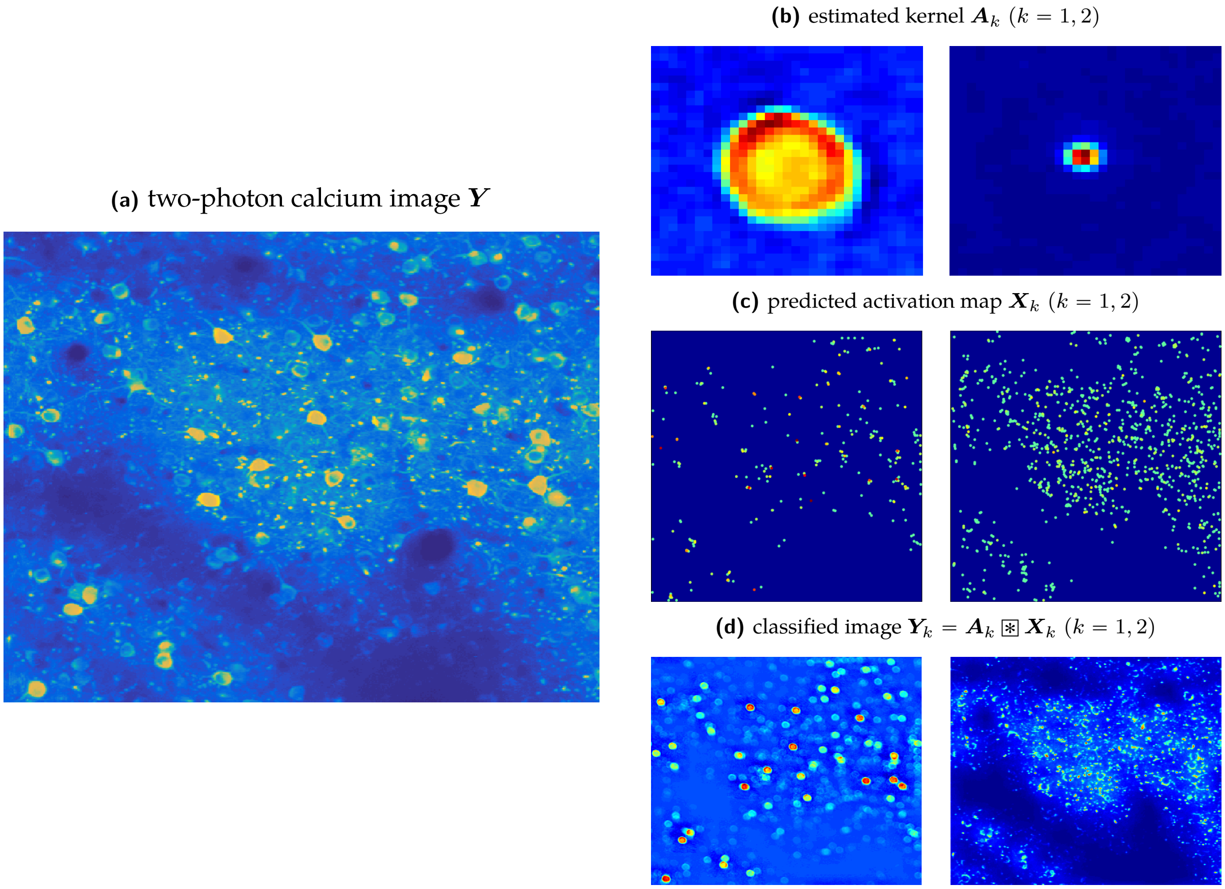
Figure 27: Aggregated result for STORM imaging. The left hand side shows the original microscopy image, and the right hand side presents the super-resolution image obtained by our method. The pixel resolution is 100 nm.

6.2.2 Convolutional dictionary learning for microscopy data analytics

Recent advances in imaging and computational techniques have resulted in the ability to obtain microscopic data in unprecedented detail and volume. SaSD and its extensions are found to be well-suited for extracting motifs and location information from such datasets from neuroscience, material science and beyond, as we have seen from Section 6.1.2. In certain settings for microscopy, the observed image can also be decomposed as

$$\underbrace{\boldsymbol{Y}}_{\text{microscopy image}} = \sum_{k=1}^{K} \underbrace{\boldsymbol{A}_{0k}}_{\text{motif } k} \boxed{*} \underbrace{\boldsymbol{X}_{0k}}_{\text{activation map}} + \underbrace{\boldsymbol{N}}_{\text{noise}}.$$

and useful information can be obtained by solving the resulting 2D SaS-CDL problem [PSG+16, CSL+18]. In this section, we demonstrate our proposed method for SaS-CDL on two different imaging modalities.


(a) two-photon calcium image $\boldsymbol{Y}$

(b) estimated kernel $\boldsymbol{A}_k$ $(k = 1, 2)$

(c) predicted activation map $\boldsymbol{X}_k$ $(k = 1, 2)$

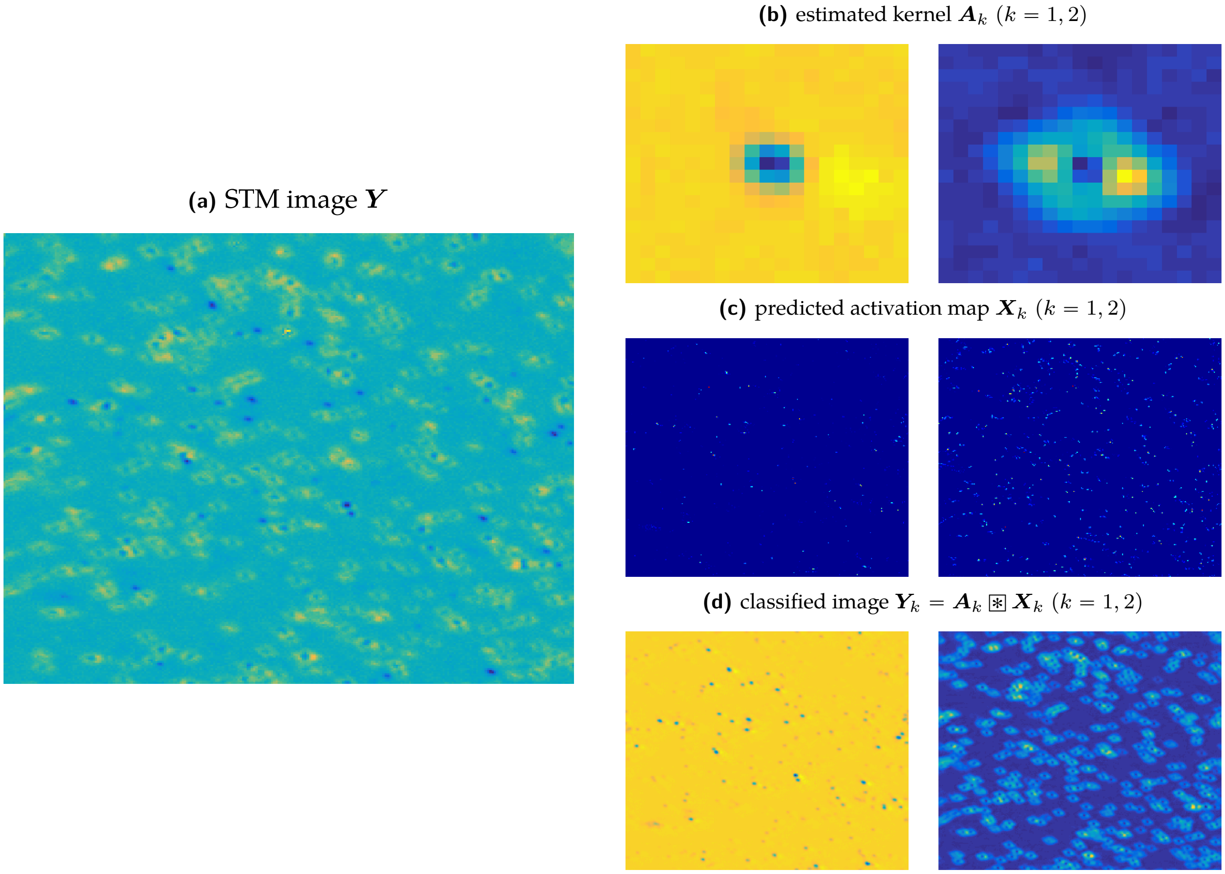
(d) classified image $\boldsymbol{Y}_k = \boldsymbol{A}_k \boxed{*} \boldsymbol{X}_k$ $(k = 1, 2)$

Figure 28: Localization and classification for calcium microscopy images. (a) shows the original image; (b) shows the estimated kernel shape for the neuron (left) and dendrite (right); (c) presents the predicted activation map for the neuron (left) and dendrite (right); (d) presents the reconstructed image $\boldsymbol{Y}_k = \boldsymbol{A}_k \boxed{*} \boldsymbol{X}_k$ $(k = 1, 2)$ for the neuron (left) and dendrite (right).

Neuronal localization for 2D calcium imaging. Tracking the spike locations of neurons in 2D calcium imaging video sequences is a challenging task due to the presence of (non)rigid motion, overlapping sources, and irregular background noise [PSG+16, GFK+17, GFG+19]. Here we show how the SaS-CDL problem can serve as a basis for distinguishing between overlapping sources. Figure 28a shows a single 512×512 frame from the two-photon fluorescence calcium microscopy dataset obtained by Allen Institute for Brain Science[21]. The

[21] The data can be found at http://observatory.brain-map.org/visualcoding/search/overview.

frame shows the cross sections of two types of neuronal components, the somata and the denrdrites, whose fluorophores that are activated at the given time frame. It is clear that these two components are primarily distinguished by their size. We decompose the frame into the somatic and dendritic components by solving SaS-CDL with the proposed method, giving us a rough estimate of the "average" somatic or dendritic motif (Figure 28b), as well as the location of each component (Figure 28c). This allows the image to be decomposed into images consisting of somata or dendrites exclusively (Figure 28d). Therefore SaS-CDL can either be the basis for a preprocessing step to remove undesired components, such as the dendrites, from a microscopy image. Furthermore, this deconvolution technique allows the individual activation map to be tracked for each video frame, opening a new way for nonrigid motion to be corrected across frames by synthesizing all activation maps. We left this as a promising future research direction.


(a) STM image $\boldsymbol{Y}$

(b) estimated kernel $\boldsymbol{A}_k$ $(k = 1, 2)$

(c) predicted activation map $\boldsymbol{X}_k$ $(k = 1, 2)$

(d) classified image $\boldsymbol{Y}_k = \boldsymbol{A}_k \boxast \boldsymbol{X}_k$ $(k = 1, 2)$

Figure 29: Defect detection for STM images. (a) shows the original STM image; (b) shows the estimated kernel shape for the defects; (c) presents the predicted activation map for the defects; (d) presents the reconstructed image $\boldsymbol{Y}_k = \boldsymbol{A}_k \boxast \boldsymbol{X}_k$ $(k = 1, 2)$ for the defect.

Defect detection in scan tunneling microscopy (STM) image. Modern high-resolution microscopes, such as the scanning tunneling electron microscope, are commonly used to study specimens that have dense and aperiodic spatial structure [CLE93, RCG+07, RSP+09]. Extracting meaningful information from images obtained from such microscopes remains a formidable challenge [KBF+03]. For instance, Figure 29a presents a STM NaFeAs sample image (with size 128×128) of a Co-doped iron arsenide crystal lattice. A method for automatically acquiring the signatures of the defects (motifs) and their locations is highly desirable [CSL+18]. Here we apply our proposed method to solve SaS-CDL and extract both the defect signatures (Figure 29b) and their locations (see Figure 29c), as well as decomposing the image into contributions based on the individual defects (Figure 29d).

7 Conclusion and Discussion

7.1 Relationship to the literature and conclusion

Nonconvex optimization. Unlike convex optimization problems, nonconvex functions usually have numerous spurious local minima, and one may also encounter "flat" saddle points that are very difficult to escape [SQW15]. In theory, even finding a local minimum of a general nonconvex function is NP-hard [MK87] – never mind the global minimum. However, recent advancements in nonconvex optimization [SQW15, GHJY15] showed that typical nonconvex problems in practice are often more structured, so that they often have much more benign geometric landscapes than the worst case: (i) all saddle points can be efficiently escaped by using negative curvature information; (ii) the equivalent "good" solutions (created by the intrinsic symmetry) are often the global optimizers of the nonconvex objective. This type of benign geometric structure has been discovered for many nonconvex problems in signal processing and machine learning, such as phase retrieval [CLS15, SQW18, QZEW17], dictionary learning [QSW14, SQW16a, SQW16b], low rank matrix recovery [GLM16, Chi16] (orthogonal) tensor decomposition [GHJY15], and phase synchronization problems [BAC18], etc. Inspired by similar benign geometric structure for a simplified nonconvex Dropped Quadratic formulation, this work provides an efficient and practical nonconvex optimization method for solving blind sparse deconvolution problems.

Blind deconvolution. The blind deconvolution problem is an ill-posed problem in its most general form. Nonetheless, problems in practice often exhibits intrinsic low-dimensional structures, showing promises for efficient optimization. Motivated by a variety of applications, many low-dimensional models for (blind) deconvolution problems have been studied in the literature. [ARR14, Chi16, LS15, LLB16, KK17, AD18, Li18] studied the problem when the unknown signals $\boldsymbol{a}_0$ and $\boldsymbol{x}_0$ either live in known low-dimensional subspaces, or are sparse in some known dictionary. These results assumed that the subspace/dictionary are chosen at random, such that the problem does not exhibit the signed shift ambiguity and can be provably solved via convex relaxation[22]. However, the assumption of random subspace/dictionary model is often unrealistic in practice. Recently, [WC16, LB18, QLZ19] consider sparse blind deconvolution with multiple measurements, where they show the problem can be efficiently solved to global optimality when the kernel is invertible. In contrast, the SaS model studied in this work exhibits much broader applications.

Because of the shift symmetry, the SaS model does not appear to be amenable for convexification, and it exhibits a more complicated nonconvex geometry. To tackle this problem, Wipf et al. [WZ14] imposes ℓ_2 regularization on $\boldsymbol{a}_0$ and provides an empirically reliable algorithm. Zhang et al. [ZLK$^+$17] studies the geometry of a simplified nonconvex objective over the sphere, and proves that in the dilute limit in which $\boldsymbol{x}_0$ is a single spike, all strict local minima are close to signed shift truncations of $\boldsymbol{a}_0$. Zhang et al. [ZKW18] formulated the problem as an ℓ_4 maximization problem over the sphere[23]. They proved that on a restricted region of the sphere every local minimizer is near a truncated signed shift of $\boldsymbol{a}_0$, when $\boldsymbol{a}_0$ is well-conditioned and $\boldsymbol{x}_0$ is sparse. Kuo et al. [KZLW19] studies a Dropped Quadratic simplification of the Bilinear Lasso objective, which provably obtains exact recovery for an incoherent kernel $\boldsymbol{a}_0$ and sparse $\boldsymbol{x}_0$. However, both the ℓ_4 maximization and Dropped Quadratic objectives are still quite far from practical formulations for solving SaSD. In contrast, as demonstrated in this work, optimizing the Bilinear Lasso formulation turns out to be much more effective in practice.

Geometry inspired optimization method for SaSD. Inspired by the benign geometric structure of the nonconvex objective, we proposed efficient nonconvex optimization methods that directly optimizes the Bilinear Lasso. The new approach exploits the geometry by (i) using data driven initializations to avoid spurious local minimizers, (ii) adopting momentum accelerating for coherent kernels, and (iii) adaptively shrinking the penalty parameter λ to achieve faster convergence and higher accuracy solutions. Our vanilla algorithm is a simple alternating descent method, which is inspired by the recent PALM methods [BST14, PS16]. In comparison with classical alternating minimization methods for sparse blind deconvolution [CW00, SM12, ZLK$^+$17], our approach does not require solving expensive Lasso subproblems, and the iterates make fast progress towards the optimal solution. On the other hand, as our method is first-order in nature, it is much more efficient

[22] Some recent work [LLSW18, MWCC17] show this problem can also be provably solved via nonconvex approaches.

[23] A similar objective is considered for the multichannel sparse blind deconvolution problem [LB18].

than the second-order trust-region [CGT00, BAC18] and curvilinear search [Gol80] methods considered in [ZLK$^+$17, KZLW19].

Convolutional dictionary learning. Furthermore, our approach has natural extensions for tackling the SaS-CDL problem when multiple unknown kernels present. By consider a similar nonconvex objective analogous to SaSD, our geometric inspired algorithm empirically solves the SaS-CDL problem to global optimality in a very efficient manner. The new method joins recent algorithmic development for solving CDL [CPR13, HA15, PRSE17, GCW18, LGCWY18, MCCM18, ZSE19]. Again, most[24] of the previous approaches [GCW18] deploy an alternating minimization strategy, which exactly solves the expensive Lasso subproblem for each iteration. In contrast, our method is much more simple, efficient and effective, demonstrated by experiments on real datasets.

7.2 Discussion and future work

Moving forward, we believe this work has opened up several future directions that could be of great empirical and theoretical interests.

Geometric analysis of Bilinear Lasso. The Bilinear Lasso formulation is one of most natural formulations for solving the SaSD problem. In light of our empirical success of solving the Bilinear Lasso, analyzing and understanding its global nonconvex landscapes is of great importance. As discuss in Section 3, the Dropped Quadratic formulation studied in [KZLW19] has commonalities with the Bilinear Lasso: both exhibit local minima at signed shifts, and both exhibit negative curvature in symmetry breaking directions. However, a major difference (and hence, major challenge) is that gradient methods for Bilinear Lasso do not retract to a subspaces – they retract to a more complicated, nonlinear set. As the empirical success we possessed here, better understandings of the geometric structure for the Bilinear Lasso in much needed. A better understanding will also shed light on SaS-CDL with multiple unknown kernels.

Parameterized sparse blind deconvolution. In this work, we studied the blind deconvolution problem with no prior knowledge of the kernel/motif $\boldsymbol{a}_0$. However, in many application, one can often obtain some side information, where the kernel is often determined by only a few parameters associated with the underlying physical processes. For example, in the calcium imaging problem we studied in Section 6, an auto regression (AR) model is often used to characterize the spiking and decaying process of the kernel, which is only determined by one or two parameters [VPM$^+$10, FZP17]. Thus, how to estimate these kernel parameters raises a challenging but interesting question. Our preliminary investigation shows that nonconvex optimization landscapes of this parameterized "semi-blind" sparse deconvolution problem also possess benign geometric properties for certain types of kernels (see Figure 30 for an illustration).

SaSD meets super-resolution. In many imaging applications, it is often desirable to solve blind deconvolution and super-resolution problems simultaneously. In other words, let $\mathcal{D}[\cdot]$ be a downsampling operator, we want to recover the high-resolution kernel $\boldsymbol{a}_0$ and sparse activation map $\boldsymbol{x}_0$ from the low-resolution measurement of the form $\boldsymbol{y} = \mathcal{D}[\boldsymbol{a}_0 \circledast \boldsymbol{x}_0]$. This type of problem appears often due to the resolution/hardware limit of the imaging system, and therefore fine details of both $\boldsymbol{a}_0$ and $\boldsymbol{x}_0$ are missing due to downsampling. For instance, in Section 6 we show that the spatial resolution of fluorescent microscopy is constraint by the diffraction limit of the light [HBZ09]. If we can solve this super-resolution SaSD problem, we can obtain much higher resolution image of living cells *in vivo*. However, our preliminary investigations show that optimizing the natural nonconvex formulation

$$\min_{\boldsymbol{a},\boldsymbol{x}} \quad \frac{1}{2}\left\|\boldsymbol{y} - \mathcal{D}\left[\boldsymbol{a} \circledast \boldsymbol{x}\right]\right\|_2^2 + \lambda\left\|\boldsymbol{x}\right\|_1, \quad \text{s.t.} \quad \boldsymbol{a} \in \mathbb{S}^{n-1}$$

tends to produce downsampled $\boldsymbol{a}_0$ and $\boldsymbol{x}_0$. How to solve this problem is largely open and remains a very interesting question. One possibility is to enforce extra constraints on $\boldsymbol{a}_0$, such as penalizing TV-norm to promote smoothness.

[24] The recent work [MCCM18] resembles some similarities to ours. However, the problem setting and formulation are still quite different.

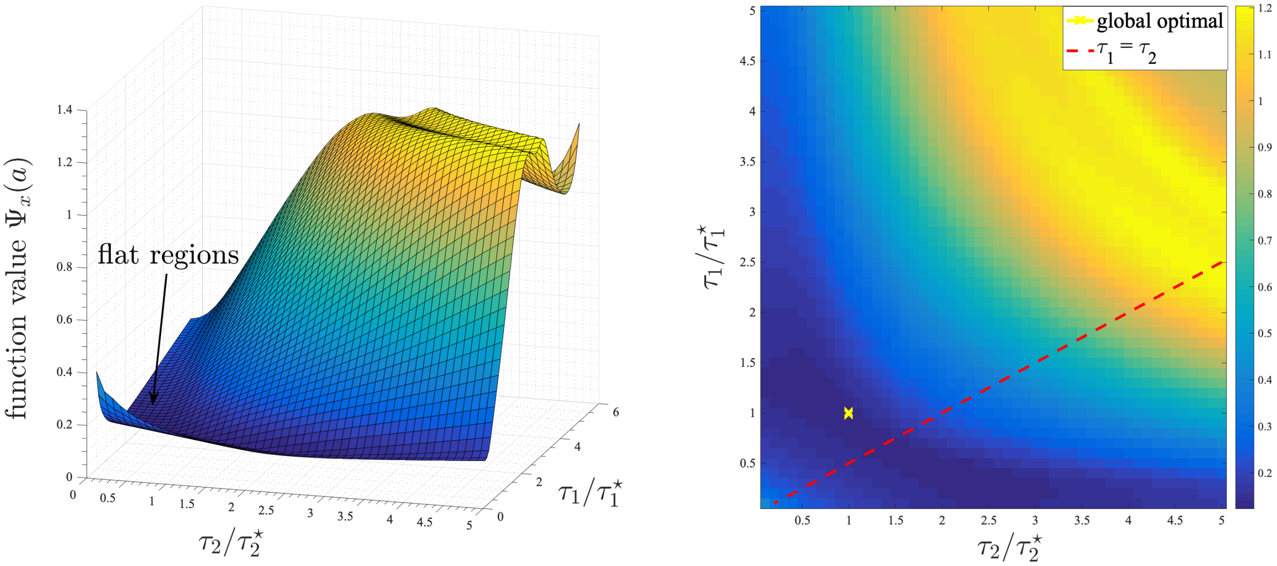


Figure 30: Nonconvex landscape of parameterized SaSD, with AR(2) kernel and two unknown parameters. The kernel $a_0(t) = \exp\left(-t/\tau_1^\star\right) - \exp\left(-t/\tau_2^\star\right)$ is parameterized by two parameters $\tau_1^\star = 0.2$ and $\tau_2^\star = 0.1$. We generate the data $\boldsymbol{y} = \boldsymbol{a}_0(t) \circledast \boldsymbol{x}_0$, where $\boldsymbol{x}_0 \sim_{i.i.d.} \mathcal{B}(\theta)$ with $\theta = 10^{-2}$. We plot the marginalized function landscape of $\Psi_x(a) = \min_{\boldsymbol{x}} \frac{1}{2} \left\| \boldsymbol{y} - \boldsymbol{a}(\tau) \circledast \boldsymbol{x} \right\|_2^2 + \lambda \left\| \boldsymbol{x} \right\|_1$ w.r.t. τ_1 and τ_2, where $\lambda = 10^{-3}$, $n_0 = 150$ and $m = 10^4$. The figures on the left and right hand sides are 3D and 2D plots of the function landscape, respectively. As we can see, the ground truth $(\tau_1^\star, \tau_2^\star)$ is the global minimizer to the nonconvex objective, but the landscape near region of the ground truth is very flat and therefore very difficult to make progress on minimizing the nonconvex objective.

Dealing with structured data. Data in practice often possesses much richer structure than the basic SaS model we studied here. For instance, in calcium imaging, the signal we obtained often has drifting/motion issues across time frames, and it also exhibits low-rank background DC components [PSG+16, GFK+17]. In STORM optical microscopy, there are rich spatial and temporal correlations within and between video frames [SMSE18]. Moreover, in many microscopy imaging data analysis problems, the motif we want to locate often exhibits unknown deformations and random rotations, and its shape is often asymmetric. How to deal with these extra structures raises a variety of challenging problems for future research.

Acknowledgement

This work was funded by NSF 1343282, NSF CCF 1527809, and NSF IIS 1546411. We would like to thank Gongguo Tang, Shuyang Ling, Carlos Fernandez-Granda, Ruoxi Sun, Liam Paniski for fruitful discussions. QQ also acknowledges supports from Microsoft PhD fellowship and the Moore-Sloan Foundation.

References

[ABG07] Pierre-Antoine. Absil, Christopher G. Baker, and Kyle A. Gallivan. Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007.

[AD18] Ali Ahmed and Laurent Demanet. Leveraging diversity and sparsity in blind deconvolution. *IEEE Transactions on Information Theory*, 64(6):3975–4000, 2018.

[AMS09] Pierre-Antoine. Absil, Robert Mahoney, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[ANW10] Alekh Agarwal, Sahand Negahban, and Martin J Wainwright. Fast global convergence rates of gradient methods for high-dimensional statistical recovery. In *Advances in Neural Information Processing Systems*, pages 37–45, 2010.

[ARR14] Ali Ahmed, Benjamin Recht, and Justin Romberg. Blind deconvolution using convex programming. *IEEE Transactions on Information Theory*, 60(3):1711–1732, 2014.

[B^+15] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[BAC18] Nicolas Boumal, Pierre-Antoine Absil, and Coralia Cartis. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33, 2018.

[BBM^+16] Alistair N Boettiger, Bogdan Bintu, Jeffrey R Moffitt, Siyuan Wang, Brian J Beliveau, Geoffrey Fudenberg, Maxim Imakaev, Leonid A Mirny, Chao-ting Wu, and Xiaowei Zhuang. Super-resolution imaging reveals distinct chromatin folding for different epigenetic states. *Nature*, 529(7586):418, 2016.

[BDH^+13] David Briers, Donald D Duncan, Evan R Hirst, Sean J Kirkpatrick, Marcus Larsson, Wiendelt Steenbergen, Tomas Stromberg, and Oliver B Thompson. Laser speckle contrast imaging: theoretical and practical limitations. *Journal of biomedical optics*, 18(6):066018, 2013.

[Bec17] Amir Beck. *First-order methods in optimization*, volume 25. SIAM, 2017.

[BK02] Simon Baker and Takeo Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, 2002.

[BPC^+11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[BPS^+06] Eric Betzig, George H Patterson, Rachid Sougrat, O Wolf Lindwasser, Scott Olenych, Juan S Bonifacino, Michael W Davidson, Jennifer Lippincott-Schwartz, and Harald F Hess. Imaging intracellular fluorescent proteins at nanometer resolution. *Science*, 313(5793):1642–1645, 2006.

[BST14] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[BT09] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[BVG13] Alexis Benichoux, Emmanuel Vincent, and Rémi Gribonval. A fundamental pitfall in blind deconvolution with sparse and shift-invariant priors. In *ICASSP-38th International Conference on Acoustics, Speech, and Signal Processing-2013*, 2013.

[CE16] Patrizio Campisi and Karen Egiazarian. *Blind image deconvolution: theory and applications*. CRC press, 2016.

[CF17] Il Yong Chun and Jeffrey A Fessler. Convolutional dictionary learning: Acceleration and convergence. *IEEE Transactions on Image Processing*, 27(4):1697–1712, 2017.

[CFG14] Emmanuel J Candès and Carlos Fernandez-Granda. Towards a mathematical theory of super-resolution. *Communications on pure and applied Mathematics*, 67(6):906–956, 2014.

[CGT00] Andrew R. Conn, Nicholas I.M. Gould, and Philippe L. Toint. *Trust region methods*, volume 1. SIAM, 2000.

[Chi16] Yuejie Chi. Guaranteed blind sparse spikes deconvolution via lifting and convex optimization. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):782–794, 2016.

[CLE93] MF Crommie, CP Lutz, and DM Eigler. Imaging standing waves in a two-dimensional electron gas. *Nature*, 363(6429):524, 1993.

[CLS15] Emmanuel J. Candès, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval via wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, April 2015.

[CMB$^+$17] Jason E Chung, Jeremy F Magland, Alex H Barnett, Vanessa M Tolosa, Angela C Tooker, Kye Y Lee, Kedar G Shah, Sarah H Felix, Loren M Frank, and Leslie F Greengard. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394, 2017.

[CPR13] Rakesh Chalasani, Jose C Principe, and Naveen Ramakrishnan. A fast proximal method for convolutional sparse coding. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5. IEEE, 2013.

[CRQ18] Fernando Chaure, Hernan Gonzalo Rey, and Rodrigo Quian Quiroga. A novel and fully automatic spike sorting implementation with variable number of features. *Journal of Neurophysiology*, 2018.

[CSL$^+$18] Sky C Cheung, John Y Shin, Yenson Lau, Zhengyu Chen, Ju Sun, Yuqian Zhang, John N Wright, and Abhay N Pasupathy. Dictionary learning in fourier transform scanning tunneling spectroscopy. *arXiv preprint arXiv:1807.10752*, 2018.

[CW98] Tony F Chan and Chiu-Kwong Wong. Total variation blind deconvolution. *IEEE transactions on Image Processing*, 7(3):370–375, 1998.

[CW00] Tony F Chan and Chiu-Kwong Wong. Convergence of the alternating minimization algorithm for blind deconvolution. *Linear Algebra and its Applications*, 316(1-3):259–285, 2000.

[CWB08] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008.

[ETS11] Chaitanya Ekanadham, Daniel Tranchina, and Eero P Simoncelli. A blind sparse deconvolution method for neural spike identification. In *Advances in Neural Information Processing Systems*, pages 1440–1448, 2011.

[FST08] Marta Fernández-Suárez and Alice Y Ting. Fluorescent probes for super-resolution imaging in living cells. *Nature Reviews Molecular cell Biology*, 9(12):929, 2008.

[FZP17] Johannes Friedrich, Pengcheng Zhou, and Liam Paninski. Fast online deconvolution of calcium imaging data. *PLoS Computational Biology*, 13(3):e1005423, 2017.

[GBW18] Dar Gilboa, Sam Buchanan, and John Wright. Efficient dictionary learning with gradient descent. *arXiv preprint arXiv:1809.10313*, 2018.

[GCW18] Cristina Garcia-Cardona and Brendt Wohlberg. Convolutional dictionary learning: A comparative review and new algorithms. *IEEE Transactions on Computational Imaging*, 4(3):366–381, 2018.

[GFG$^+$19] Andrea Giovannucci, Johannes Friedrich, Pat Gunn, Jeremie Kalfon, Brandon L Brown, Sue Ann Koay, Jiannis Taxidis, Farzaneh Najafi, Jeffrey L Gauthier, Pengcheng Zhou, et al. Caiman an open source tool for scalable calcium imaging data analysis. *Elife*, 8:e38173, 2019.

[GFK$^+$17] Andrea Giovannucci, Johannes Friedrich, Matt Kaufman, Anne Churchland, Dmitri Chklovskii, Liam Paninski, and Eftychios A Pnevmatikakis. Onacid: Online analysis of calcium imaging data in real time. In *Advances in Neural Information Processing Systems*, pages 2381–2391, 2017.

[GHJY15] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Proceedings of The 28th Conference on Learning Theory*, pages 797–842, 2015.

[GK12] Christine Grienberger and Arthur Konnerth. Imaging calcium in neurons. *Neuron*, 73(5):862–885, 2012.

[GLM16] Rong Ge, Jason D Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems*, pages 2973–2981, 2016.

[GMWZ17] Donald Goldfarb, Cun Mu, John Wright, and Chaoxu Zhou. Using negative curvature in solving nonlinear programs. *Computational Optimization and Applications*, 68(3):479–502, 2017.

[Gol80] Donald Goldfarb. Curvilinear path steplength algorithms for minimization which use directions of negative curvature. *Mathematical Programming*, 18(1):31–40, 1980.

[HA15] Furong Huang and Animashree Anandkumar. Convolutional dictionary learning through tensor factorization. In *Feature Extraction: Modern Questions and Challenges*, pages 116–129, 2015.

[Hay94] Simon S Haykin. *Blind deconvolution*. Prentice Hall, 1994.

[HBZ09] Bo Huang, Mark Bates, and Xiaowei Zhuang. Super-resolution fluorescence microscopy. *Annual Review of Biochemistry*, 78:993–1016, 2009.

[HBZ10] Bo Huang, Hazen Babcock, and Xiaowei Zhuang. Breaking the diffraction barrier: super-resolution imaging of cells. *Cell*, 143(7):1047–1058, 2010.

[Hel07] Stefan W Hell. Far-field optical nanoscopy. *science*, 316(5828):1153–1158, 2007.

[HGM06] Samuel T Hess, Thanu PK Girirajan, and Michael D Mason. Ultra-high resolution imaging by fluorescence photoactivation localization microscopy. *Biophysical journal*, 91(11):4258–4272, 2006.

[HUK11] Seamus J Holden, Stephan Uphoff, and Achillefs N Kapanidis. Daostorm: an algorithm for high-density super-resolution microscopy. *Nature Methods*, 8(4):279, 2011.

[HWBZ08] Bo Huang, Wenqin Wang, Mark Bates, and Xiaowei Zhuang. Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy. *Science*, 319(5864):810–813, 2008.

[HYZ08] Elaine T Hale, Wotao Yin, and Yin Zhang. Fixed-point continuation for \ell_1-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19(3):1107–1130, 2008.

[JGN$^+$17] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1724–1732, 2017.

[JNJ18] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. In *Conference On Learning Theory*, pages 1042–1085, 2018.

[KBF$^+$03] Steven A Kivelson, Ian P Bindloss, Eduardo Fradkin, Vadim Oganesyan, JM Tranquada, Aharon Kapitulnik, and Craig Howald. How to detect fluctuating stripes in the high-temperature superconductors. *Reviews of Modern Physics*, 75(4):1201, 2003.

[KH96] Deepa Kundur and Dimitrios Hatzinakos. Blind image deconvolution. *IEEE Signal Processing Magazine*, 13(3):43–64, 1996.

[KK17] Michael Kech and Felix Krahmer. Optimal injectivity conditions for bilinear inverse problems with applications to identifiability of deconvolution problems. *SIAM Journal on Applied Algebra and Geometry*, 1(1):20–37, 2017.

[KZLW19] Han-Wen Kuo, Yuqian Zhang, Yenson Lau, and John Wright. Geometry and symmetry in short-and-sparse deconvolution. In *International Conference on Machine Learning (ICML)*, June 2019.

[LB95] MH Loke and RD Barker. Least-squares deconvolution of apparent resistivity pseudosections. *Geophysics*, 60(6):1682–1690, 1995.

[LB18] Yanjun Li and Yoram Bresler. Global geometry of multichannel sparse blind deconvolution on the sphere. In *Advances in Neural Information Processing Systems*, pages 1132–1143, 2018.

[LC11] Bonnie O Leung and Keng C Chou. Review of super-resolution fluorescence microscopy for biology. *Applied Spectroscopy*, 65(9):967–980, 2011.

[Lew98] Michael S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78, 1998.

[LGCWY18] Jialin Liu, Cristina Garcia-Cardona, Brendt Wohlberg, and Wotao Yin. First-and second-order methods for online convolutional dictionary learning. *SIAM Journal on Imaging Sciences*, 11(2):1589–1628, 2018.

[Li18] Yanjun Li. *Bilinear inverse problems with sparsity: optimal identifiability conditions and efficient recovery*. PhD thesis, University of Illinois at Urbana-Champaign, 2018.

[LLB16] Yanjun Li, Kiryung Lee, and Yoram Bresler. Identifiability in blind deconvolution with subspace or sparsity constraints. *IEEE Transactions on Information Theory*, 62(7):4266–4275, 2016.

[LLSW18] Xiaodong Li, Shuyang Ling, Thomas Strohmer, and Ke Wei. Rapid, robust, and reliable blind deconvolution via nonconvex optimization. *Applied and Computational Harmonic Analysis*, 2018.

[LPP$^+$17] Jason D Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I Jordan, and Benjamin Recht. First-order methods almost always avoid strict saddle points. *Mathematical Programming*, pages 1–27, 2017.

[LS15] Shuyang Ling and Thomas Strohmer. Self-calibration and biconvex compressive sensing. *Inverse Problems*, 31(11):115002, 2015.

[LWDF11a] Anat Levin, Yair Weiss, Fredo Durand, and William T Freeman. Understanding blind deconvolution algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2354–2367, 2011.

[LWDF11b] Anat Levin, Yair Weiss, Fredo Durand, and William T Freeman. Understanding blind deconvolution algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2354–2367, 2011.

[MCCM18] Jyoti Maggu, Emilie Chouzenoux, Giovanni Chierchia, and Angshul Majumdar. Convolutional transform learning. In *International Conference on Neural Information Processing*, pages 391–398, 2018.

[MK87] Katta G. Murty and Santosh N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

[MWCC17] Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion and blind deconvolution. *arXiv preprint arXiv:1711.10467*, 2017.

[Nes13a] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[Nes13b] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[NN14] Karin Nienhaus and G Ulrich Nienhaus. Fluorescent proteins for live-cell imaging with super-resolution. *Chemical Society Reviews*, 43(4):1088–1106, 2014.

[NP06] Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[NW06] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[PB$^+$14] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[Pol64] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[PRSE17] Vardan Papyan, Yaniv Romano, Jeremias Sulam, and Michael Elad. Convolutional dictionary learning via local processing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5296–5304, 2017.

[PS16] Thomas Pock and Shoham Sabach. Inertial proximal alternating linearized minimization (ipalm) for nonconvex and nonsmooth problems. *SIAM Journal on Imaging Sciences*, 9(4):1756–1787, 2016.

[PSG$^+$16] Eftychios A Pnevmatikakis, Daniel Soudry, Yuanjun Gao, Timothy A Machado, Josh Merel, David Pfau, Thomas Reardon, Yu Mu, Clay Lacefield, Weijian Yang, et al. Simultaneous denoising, deconvolution, and demixing of calcium imaging data. *Neuron*, 89(2):285–299, 2016.

[QLL$^+$10] Tingwei Quan, Pengcheng Li, Fan Long, Shaoqun Zeng, Qingming Luo, Per Niklas Hedde, Gerd Ulrich Nienhaus, and Zhen-Li Huang. Ultra-fast, high-precision image analysis for localization-based super resolution microscopy. *Optics Express*, 18(11):11867–11876, 2010.

[QLZ19] Qing Qu, Xiao Li, and Zhihui Zhu. A nonconvex approach for exact and efficient multichannel sparse blind deconvolution. *arXiv preprint arXiv:1908.10776*, 2019.

[QNBS04] R Quian Quiroga, Zoltan Nadasdy, and Yoram Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Computation*, 16(8):1661–1687, 2004.

[QSW14] Qing Qu, Ju Sun, and John Wright. Finding a sparse vector in a subspace: Linear sparsity using alternating directions. In *Advances in Neural Information Processing Systems*, pages 3401–3409, 2014.

[QZEW17] Qing Qu, Yuqian Zhang, Yonina Eldar, and John Wright. Convolutional phase retrieval. In *Advances in Neural Information Processing Systems*, pages 6086–6096, 2017.

[RBZ06a] Michael J Rust, Mark Bates, and Xiaowei Zhuang. Stochastic optical reconstruction microscopy (storm) provides sub-diffraction-limit image resolution. *Nature Methods*, 3(10):793, 2006.

[RBZ06b] Michael J Rust, Mark Bates, and Xiaowei Zhuang. Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm). *Nature Methods*, 3(10):793, 2006.

[RCG$^+$07] Gregory M Rutter, JN Crain, NP Guisinger, T Li, PN First, and JA Stroscio. Scattering and interference in epitaxial graphene. *Science*, 317(5835):219–222, 2007.

[RPQ15] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. Past, present and future of spike sorting techniques. *Brain Research Bulletin*, 119:106–117, 2015.

[RSP$^+$09] Pedram Roushan, Jungpil Seo, Colin V Parker, Yew San Hor, David Hsieh, Dong Qian, Anthony Richardella, M Zahid Hasan, Robert Joseph Cava, and Ali Yazdani. Topological surface states protected from backscattering by chiral spin texture. *Nature*, 460(7259):1106, 2009.

[SFB18] Andrew H Song, Francisco Flores, and Demba Ba. Spike sorting by convolutional dictionary learning. *arXiv preprint arXiv:1806.01979*, 2018.

[SGG$^+$09] Gleb Shtengel, James A Galbraith, Catherine G Galbraith, Jennifer Lippincott-Schwartz, Jennifer M Gillette, Suliana Manley, Rachid Sougrat, Clare M Waterman, Pakorn Kanchanawong, Michael W Davidson, et al. Interferometric fluorescent super-resolution microscopy resolves 3d cellular ultrastructure. *Proceedings of the National Academy of Sciences*, 106(9):3125–3130, 2009.

[SGHK03] Christoph Stosiek, Olga Garaschuk, Knut Holthoff, and Arthur Konnerth. In vivo two-photon calcium imaging of neuronal networks. *Proceedings of the National Academy of Sciences*, 100(12):7319–7324, 2003.

[SM12] Filip Sroubek and Peyman Milanfar. Robust multichannel blind deconvolution via fast alternating minimization. *IEEE Transactions on Image processing*, 21(4):1687–1700, 2012.

[SMSE18] Oren Solomon, Maor Mutzafi, Mordechai Segev, and Yonina C Eldar. Sparsity-based super-resolution microscopy from correlation information. *Optics Express*, 26(14):18238–18269, 2018.

[SN06] Pinaki Sarder and Arye Nehorai. Deconvolution methods for 3-d fluorescence microscopy images. *IEEE Signal Processing Magazine*, 23(3):32–45, 2006.

[SQW15] Ju Sun, Qing Qu, and John Wright. When are nonconvex problems not scary? *arXiv preprint arXiv:1510.06096*, 2015.

[SQW16a] Ju Sun, Qing Qu, and John Wright. Complete dictionary recovery over the sphere i: Overview and the geometric picture. *IEEE Transactions on Information Theory*, 63(2):853–884, 2016.

[SQW16b] Ju Sun, Qing Qu, and John Wright. Complete dictionary recovery over the sphere ii: Recovery by riemannian trust-region method. *IEEE Transactions on Information Theory*, 63(2):885–914, 2016.

[SQW18] Ju Sun, Qing Qu, and John Wright. A geometric analysis of phase retrieval. *Foundations of Computational Mathematics*, 18(5):1131–1198, 2018.

[STP17] Lorenzo Stella, Andreas Themelis, and Panagiotis Patrinos. Forward–backward quasi-newton methods for nonsmooth optimization problems. *Computational Optimization and Applications*, 67(3):443–487, 2017.

[TBF$^+$16] Lucas Theis, Philipp Berens, Emmanouil Froudarakis, Jacob Reimer, Miroslav Román Rosón, Tom Baden, Thomas Euler, Andreas S Tolias, and Matthias Bethge. Benchmarking spike rate inference in population calcium imaging. *Neuron*, 90(3):471–482, 2016.

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[VPM$^+$10] Joshua T Vogelstein, Adam M Packer, Timothy A Machado, Tanya Sippy, Baktash Babadi, Rafael Yuste, and Liam Paninski. Fast nonnegative deconvolution for spike train inference from population calcium imaging. *Journal of Neurophysiology*, 104(6):3691–3704, 2010.

[WC16] Liming Wang and Yuejie Chi. Blind deconvolution from multiple sparse inputs. *IEEE Signal Processing Letters*, 23(10):1384–1388, 2016.

[WNF09] Stephen J Wright, Robert D Nowak, and Mário AT Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[WZ14] David Wipf and Haichao Zhang. Revisiting bayesian blind deconvolution. *The Journal of Machine Learning Research*, 15(1):3595–3634, 2014.

[XZ13] Lin Xiao and Tong Zhang. A proximal-gradient homotopy method for the sparse least-squares problem. *SIAM Journal on Optimization*, 23(2):1062–1091, 2013.

[YHV17] Florence Yellin, Benjamin D Haeffele, and René Vidal. Blood cell detection and counting in holographic lens-free imaging by convolutional sparse dictionary learning and coding. In *IEEE 14th International Symposium on Biomedical Imaging*, pages 650–653. IEEE, 2017.

[YSE$^+$18] Pierre Yger, Giulia LB Spampinato, Elric Esposito, Baptiste Lefebvre, Stéphane Deny, Christophe Gardella, Marcel Stimberg, Florian Jetter, Guenther Zeck, Serge Picaud, et al. A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo. *Elife*, 7:e34518, 2018.

[YWHM10] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[ZCB$^+$14] Yin Zhou, Hang Chang, Kenneth Barner, Paul Spellman, and Bahram Parvin. Classification of histology sections via multispectral convolutional sparse coding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3081–3088, 2014.

[ZKW18] Yuqian Zhang, Han-wen Kuo, and John Wright. Structured local minima in sparse blind deconvolution. In *Advances in Neural Information Processing Systems*, pages 2328–2337, 2018.

[ZLK$^+$17] Yuqian Zhang, Yenson Lau, Han-Wen Kuo, Sky Cheung, Abhay Pasupathy, and John Wright. On the global geometry of sphere-constrained sparse blind deconvolution. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4381–4389. IEEE, 2017.

[ZSE19] Ev Zisselman, Jeremias Sulam, and Michael Elad. A local block coordinate descent algorithm for the csc model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8208–8217, 2019.

[ZZEH12] Lei Zhu, Wei Zhang, Daniel Elnatan, and Bo Huang. Faster storm using compressed sensing. *Nature Methods*, 9(7):721, 2012.

Appendices

The appendix is organized as follows. In Appendix A, we introduce the basic notations and terms that are used throughout the draft. In Appendix B, we describe the proposed algorithmic pipeline for solving sparse deconvolution problems in very detail. Finally, Appendix C provides all the missing complementary details of implementing the proposed algorithm for solving SaS-BD and SaS-CDL.

A Basic notations

Throughout this paper, all vectors/matrices are written in bold font $\boldsymbol{a}/\boldsymbol{A}$; indexed values are written as a_i, A_{ij}. Vectors with all zero and all one entries are denoted as $\mathbf{0}_m$ and $\mathbf{1}_m$, respectively, with m denoting its length. The i-th canonical basis vector is denoted by $\boldsymbol{e}_i$. We use $\mathbb{S}^{n-1}$ to denote an n-dimensional unit sphere in the Euclidean space $\mathbb{R}^n$. We use $\boldsymbol{z}^{(k)}$ to denote the optimization variable $\boldsymbol{z}$ at kth iteration. We let $[m] = \{1, 2, \cdots, m\}$. For a multivariate function $\Psi(\boldsymbol{a}, \boldsymbol{x})$, we use $\Psi_{\boldsymbol{a}}(\boldsymbol{x})$ and $\Psi_{\boldsymbol{x}}(\boldsymbol{a})$ to denote marginal functions of $\Psi(\boldsymbol{a}, \boldsymbol{x})$ with one variable fixed, respectively. Next, we define several useful operators appear throughout the paper and the appendices.

Some basic operators. We use $\boldsymbol{\iota}_{n\to m}$ to denote a zero-padding operator $\boldsymbol{\iota}_{n\to m}\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v} \\ \mathbf{0}_{n-m} \end{bmatrix}$, which zero-pads a length n vector $\boldsymbol{v} \in \mathbb{R}^n$ to length m $(n \leqslant m)$. Correspondingly, its adjoint operator $\boldsymbol{\iota}_{n\to m}^*$ denotes the restriction of a vector of length-m to its first n coordinate (and $\boldsymbol{\iota}_{n\to m}^* = \boldsymbol{\iota}_{m\to n}$). Similarly, given a subset $\mathcal{I} \subseteq [m]$ and a vector $\boldsymbol{v} \in \mathbb{R}^{|\mathcal{I}|}$, we use $\boldsymbol{\iota}_{\mathcal{I}\to m} : \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}^m$ to denote an operator that maps $\boldsymbol{v}$ to a zero-padded vector whose entries in $\mathcal{I}$ corresponding to those of $\boldsymbol{v}$.

We use $\mathcal{P}_{\boldsymbol{v}}$ and $\mathcal{P}_{\boldsymbol{v}^\perp}$ to denote projections onto $\boldsymbol{v}$ and its orthogonal complement, respectively. We let $\mathcal{P}_{\mathbb{S}^{n-1}}(\cdot)$ to be the ℓ_2-normalization operator. To sum up, for any two vectors $\boldsymbol{v}$ and $\boldsymbol{u} \in \mathbb{R}^n$, we have

$$\mathcal{P}_{\boldsymbol{v}^\perp}\boldsymbol{u} = \boldsymbol{u} - \frac{\boldsymbol{v}\boldsymbol{v}^\top}{\|\boldsymbol{v}\|_2^2}\boldsymbol{u}, \quad \mathcal{P}_{\boldsymbol{v}}\boldsymbol{u} = \frac{\boldsymbol{v}\boldsymbol{v}^\top}{\|\boldsymbol{v}\|_2^2}\boldsymbol{u}, \quad \mathcal{P}_{\mathbb{S}^{n-1}}\boldsymbol{u} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|_2}.$$

Circular convolution and circulant matrices. The convolution operator $\circledast$ is *circular* with modulo-m: $(\boldsymbol{a} \circledast \boldsymbol{x})_i = \sum_{j=0}^{m-1} a_j x_{i-j}$, and we use $\boxast$ to specify the *circular* convolution in 2D. For a vector $\boldsymbol{v} \in \mathbb{R}^m$, let $\mathrm{s}_\ell[\boldsymbol{v}]$ denote the cyclic shift of $\boldsymbol{v}$ with length ℓ. In addition, we use $\widehat{\mathrm{s}}_\ell[\boldsymbol{v}]$ to denote a $3m-2$ length zero-pad shift, i.e.,

$$\widehat{\mathrm{s}}_\ell[\boldsymbol{v}] = \mathrm{s}_\ell[\begin{bmatrix} \mathbf{0}_{m-1} \\ \boldsymbol{v} \\ \mathbf{0}_{m-1} \end{bmatrix}].$$

We introduce the circulant matrix $\boldsymbol{C}_{\boldsymbol{v}} \in \mathbb{R}^{m\times m}$ generated through $\boldsymbol{v} \in \mathbb{R}^m$,

$$\boldsymbol{C}_{\boldsymbol{v}} = \begin{bmatrix} v_1 & v_m & \cdots & v_3 & v_2 \\ v_2 & v_1 & v_m & & v_3 \\ \vdots & v_2 & v_1 & \ddots & \vdots \\ v_{m-1} & & \ddots & \ddots & v_m \\ v_m & v_{m-1} & \cdots & v_2 & v_1 \end{bmatrix} = \begin{bmatrix} \mathrm{s}_0[\boldsymbol{v}] & \mathrm{s}_1[\boldsymbol{v}] & \cdots & \mathrm{s}_{m-1}[\boldsymbol{v}] \end{bmatrix}.$$

Now the circulant convolution can also be written in a simpler matrix-vector product form. For instance, for any $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^n$ $(n \leqslant m)$,

$$\boldsymbol{u} \circledast \boldsymbol{v} = \boldsymbol{C}_{\boldsymbol{u}} \cdot \boldsymbol{\iota}_{n\to m}\boldsymbol{v} = \boldsymbol{C}_{\boldsymbol{\iota}_{n\to m}\boldsymbol{v}} \cdot \boldsymbol{u} = \boldsymbol{v} \circledast \boldsymbol{u}.$$

In addition, the correlation between $\boldsymbol{u}$ and $\boldsymbol{v}$ can be also written in a similar form of convolution operator which reverses one vector before convolution. Let $\check{\boldsymbol{v}}$ denote a *cyclic reversal* of $\boldsymbol{v} \in \mathbb{R}^m$, i.e., $\check{\boldsymbol{v}} = [v_1, v_m, v_{m-1}, \cdots, v_2]^\top$, and define two correlation matrices $\boldsymbol{C}_{\boldsymbol{v}}^*\boldsymbol{e}_j = \mathrm{s}_j[\boldsymbol{v}]$ and $\check{\boldsymbol{C}}_{\boldsymbol{v}}\boldsymbol{e}_j = \mathrm{s}_{-j}[\boldsymbol{v}]$. The two operators satisfy

$$\boldsymbol{C}_{\boldsymbol{\iota}_{n\to m}\boldsymbol{v}}^*\boldsymbol{u} = \check{\boldsymbol{v}} \circledast \boldsymbol{u}, \quad \check{\boldsymbol{C}}_{\boldsymbol{\iota}_{n\to m}\boldsymbol{v}}\boldsymbol{u} = \boldsymbol{v} \circledast \check{\boldsymbol{u}}.$$

Notation for several distributions. We use $i.i.d.$ to denote *identically* and *independently distributed* random variables. In addition, we introduce and denote several distributions as follows.

- We use $\mathcal{N}(\mu, \sigma^2)$ to denote the Gaussian distribution with mean μ and variance σ^2, and use $\mathcal{U}(\mathbb{S}^{n-1})$ to denote a uniform distribution over the sphere $\mathbb{S}^{n-1}$;
- we use $\mathcal{B}(\theta)$ to denote the Bernoulli distribution with parameter θ controling the nonzero probability;
- we use $\mathcal{BG}(\theta)$ to denote Bernoulli-Gaussian distribution, i.e., if $u \sim \mathcal{BG}(\theta)$, then $u = b \cdot g$ with $b \sim \mathcal{B}(\theta)$ and $g \sim \mathcal{N}(0,1)$;
- we use $\mathcal{BR}(\theta)$ to denote Bernoulli-Rademacher distribution, i.e., if $u \sim \mathcal{BR}(\theta)$, then $u = b \cdot r$ with $b \sim \mathcal{B}(\theta)$ and r follows Rademacher distribution.

B Algorithmic Pipeline

In this part of appendix, we introduce a general algorithmic pipeline for solving sparse deconvolution problems, including SaSD and SaS-CDL. We describe the optimization problem in a more general form here. Namely, we consider the following problem

$$\min_{\boldsymbol{a},\boldsymbol{x}} \ \Psi(\boldsymbol{a},\boldsymbol{x}) \ = \ \psi(\boldsymbol{a},\boldsymbol{x}) \ + \ \lambda \cdot g(\boldsymbol{x}), \qquad \text{s.t.} \quad \boldsymbol{a} \ \in \ \mathcal{M}, \tag{B.1}$$

where $\psi(\boldsymbol{a},\boldsymbol{x})$ is a data fidelity term that we to be twice continuously differentiable, $g(\boldsymbol{x})$ is a convex (possibly nonsmooth) sparse promoting penalty, and $\mathcal{M}$ is a smooth Riemannian manifold. Again, the penalty $\lambda > 0$ balances the weights of two terms $\psi(\boldsymbol{a},\boldsymbol{x})$ and $g(\boldsymbol{x})$. The objective in Equation (B.1) generalizes the Bilinear Lasso formulation for SaSD and SaS-CDL problems:

- **SaSD.** Recall from Equation (3.1), we have

$$\psi(\boldsymbol{a},\boldsymbol{x}) \ = \ \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{a} \circledast \boldsymbol{x}\right\|_2^2, \quad g(\boldsymbol{x}) \ = \ \left\|\boldsymbol{x}\right\|_1, \quad \mathcal{M} \ = \ \mathbb{S}^{n-1}.$$

- **SaS-CDL.** Let $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{a}_1 & \cdots & \boldsymbol{a}_N\end{bmatrix}$ and $\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1 & \cdots & \boldsymbol{x}_N\end{bmatrix}$, by Equation (4.6),

$$\psi(\boldsymbol{A},\boldsymbol{X}) \ = \ \frac{1}{2}\left\|\boldsymbol{y} - \sum_{k=1}^{N} \boldsymbol{a}_k \circledast \boldsymbol{x}_k\right\|_2^2, \quad g(\boldsymbol{X}) \ = \ \left\|\boldsymbol{X}\right\|_1, \quad \mathcal{M} \ = \ \left\{\boldsymbol{A} \in \mathbb{R}^{n \times N} \ \mid \ \boldsymbol{a}_k \in \mathbb{S}^{n-1}, \ 1 \leqslant k \leqslant N\right\}.$$

For the rest of this appendix, we introduce our algorithms based on the general formulation in Equation (B.1) for the ease of exposition. We defer more implementation details for SaSD and SaS-CDL to Appendix C.

B.1 Alternating descent method

B.1.1 Vanilla ADM

We begin this part of appendix by introducing a *vanilla* first-order method for solving Equation (B.1) based on alternating descent method (ADM). The method minimizes the objective by alternating between taking descent steps on one variable with the other fixed. The basic algorithm pipeline is summarized in Algorithm 2.

Fix $\boldsymbol{a}$ and take a proximal gradient step on $\boldsymbol{x}$. Given $\boldsymbol{a}$ being fixed, the marginal function of $\Psi(\boldsymbol{a},\boldsymbol{x})$,

$$\Psi_{\boldsymbol{a}}(\boldsymbol{x}) = \psi_{\boldsymbol{a}}(\boldsymbol{x}) + \lambda \cdot g(\boldsymbol{x}),$$

is *nonsmooth* w.r.t. $\boldsymbol{x}$. A classical way to deal with nonsmoothness is by considering its *smooth envelope* [STP17], and take a proximal gradient step on the smooth variant [PB+14],

$$\boldsymbol{x}^{(k+1)} \ = \ \mathcal{P}_t(\boldsymbol{x}^{(k)}) \ = \ \boldsymbol{x}^{(k)} - t\mathcal{G}_t(\boldsymbol{x}^{(k)}), \qquad \mathcal{G}_t(\boldsymbol{x}) \ = \ t^{-1}\left(\boldsymbol{x} - \mathcal{P}_t(\boldsymbol{x})\right), \tag{B.2}$$

Algorithm 2 Alternating Descent Method (ADM)

```
Input:   Measurement $\boldsymbol{y} \in \mathbb{R}^m$; stepsizes $t_0$ and $\tau_0$; penalty $\lambda > 0$.
Output:  Final iterate $\boldsymbol{a}_\star$, $\boldsymbol{x}_\star$.
  Initialize $\boldsymbol{a}^{(0)}$ using Equation (3.5), $\boldsymbol{x}^{(0)} \leftarrow \mathbf{0}_m$, and $k \leftarrow 0$.
  while not converged do
    Fix $\boldsymbol{a}^{(k)}$ and take a proximal gradient step on $\boldsymbol{x}$ with stepsize $t_k$
        $\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} - t_k \mathcal{G}_{t_k}\left(\boldsymbol{x}^{(k)}\right),$
    Fix $\boldsymbol{x}^{(k+1)}$ and take a Riemannian gradient step on $\boldsymbol{a}$ with stepsize $\tau_k$
        $\boldsymbol{a}^{(k+1)} \leftarrow \mathcal{R}^{\mathcal{M}}_{\boldsymbol{a}^{(k)}}\left(-\tau \cdot \operatorname{grad} \psi_{\boldsymbol{x}^{(k+1)}}(\boldsymbol{a}^{(k)})\right),$
    Update $k \leftarrow k + 1$.
  end while
```

where $\mathcal{G}_t(\boldsymbol{x})$ is termed as *composite gradient mapping*, and $\mathcal{P}_t(\boldsymbol{x}^{(k)})$ is a proximal mapping that we introduce in the following. For any $t > 0$, consider a quadratic approximation of $\Psi_{\boldsymbol{a}}(\boldsymbol{x})$ at a given point $\overline{\boldsymbol{x}}$,

$$Q^t_{\boldsymbol{a}}(\boldsymbol{x}, \overline{\boldsymbol{x}}) \;=\; \psi_{\boldsymbol{a}}(\overline{\boldsymbol{x}}) + \langle \boldsymbol{x} - \overline{\boldsymbol{x}}, \nabla \psi_{\boldsymbol{a}}(\overline{\boldsymbol{x}}) \rangle + \frac{1}{2t}\|\boldsymbol{x} - \overline{\boldsymbol{x}}\|_2^2 + \lambda \cdot g(\boldsymbol{x}).$$

For a convex $g(\cdot)$, $Q^t_{\boldsymbol{a}}(\boldsymbol{x}, \overline{\boldsymbol{x}})$ admits a unique minimizer via the proximal mapping

$$\mathcal{P}_t(\overline{\boldsymbol{x}}) \;=\; \arg\min_{\boldsymbol{x}} Q^t_{\boldsymbol{a}}(\boldsymbol{x}, \overline{\boldsymbol{x}}) \;=\; \operatorname{prox}_g^{\lambda t}\left(\overline{\boldsymbol{x}} - t \nabla \psi_{\boldsymbol{a}}(\overline{\boldsymbol{x}})\right),$$

where we denote the *proximal operator* of $g(\cdot)$ by

$$\operatorname{prox}_g^{\rho}(\boldsymbol{x}) \;\doteq\; \arg\min_{\boldsymbol{z}} \left\{ \rho \cdot g(\boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2 \right\}.$$

By plugging $\mathcal{P}_t(\overline{\boldsymbol{x}})$ back into $Q^t_{\boldsymbol{a}}(\boldsymbol{x}, \overline{\boldsymbol{x}})$, it gives the so-called *forward-backward envelope* [STP17] of $\Psi_{\boldsymbol{a}}(\boldsymbol{x})$ as

$$F^t_{\boldsymbol{a}}(\overline{\boldsymbol{x}}) \;=\; \min_{\boldsymbol{x}} \left\{ Q^t_{\boldsymbol{a}}(\boldsymbol{x}, \overline{\boldsymbol{x}}) \right\} \;=\; Q^t_{\boldsymbol{a}}\left(\mathcal{P}_t(\overline{\boldsymbol{x}}), \overline{\boldsymbol{x}}\right),$$

which serves as a smooth upper bound (envelope), i.e.,

$$F^t_{\boldsymbol{a}}(\boldsymbol{x}) \;=\; Q^t_{\boldsymbol{a}}\left(\mathcal{P}_t(\boldsymbol{x}), \boldsymbol{x}\right) \;\geqslant\; \Psi_{\boldsymbol{a}}(\mathcal{P}_t(\boldsymbol{x})) \tag{B.3}$$

for any $t \in (0, 1/L_\psi)$, where L_ψ is the Lipschitz constant of $\nabla \psi_{\boldsymbol{a}}(\boldsymbol{x})$ [Nes13b, Bec17]. Indeed, the composite gradient mapping $\mathcal{G}_t(\boldsymbol{x})$ in Equation (B.2) can be interpreted as the gradient on the smooth envelope $F^t_{\boldsymbol{a}}(\boldsymbol{x})$, so that the proximal step in Equation (B.2) can be viewed as a gradient descent method. Additionally, we can show that the function value produced by the proximal gradient in Equation (B.2) is nonincreasing $\Psi_a(\boldsymbol{x}^{(k+1)}) \leqslant \Psi_a(\boldsymbol{x}^{(k)})$, when $t \in (0, 1/L_{\psi_{\boldsymbol{a}}})$ [BT09, Bec17].

The parameter t is usually set to be $1/L_{\psi_{\boldsymbol{a}}}$ for fast convergence. However, computing $L_{\psi_{\boldsymbol{u}}}$ for each iteration can be expensive. Instead, we use a backtracking rule (see Algorithm 3) to adaptively choose t based on the inequality in Equation (B.3).

Fix $\boldsymbol{x}$ and take a Riemannian gradient step on $\boldsymbol{a}$. As our optimization variable $\boldsymbol{a}$ in constraint over the Riemannian manifold $\mathcal{M}$, we consider the Riemannian derivative on $\psi_{\boldsymbol{x}}(\boldsymbol{a})$ [AMS09]. Starting from an iterate $\boldsymbol{a}^{(k)}$, we take a Riemannian gradient step on $\psi_{\boldsymbol{x}}(\boldsymbol{a})$ by

$$\boldsymbol{a}^{(k+1)} = \mathcal{R}^{\mathcal{M}}_{\boldsymbol{a}^{(k)}}\left(-\tau \cdot \operatorname{grad} \psi_{\boldsymbol{x}}(\boldsymbol{a}^{(k)})\right), \tag{B.4}$$

Algorithm 3 Backtracking rule for stepsize t

```
Input:  $\boldsymbol{a}$, $\boldsymbol{x}$, $t_0$, $\beta \in (0,1)$
Output:  $t$, $\mathcal{P}_t(\boldsymbol{x})$,
  Set $t \leftarrow t_0$ and compute $\mathcal{P}_t(\boldsymbol{x})$.
  while $\Psi_{\boldsymbol{a}}(\mathcal{P}_t(\boldsymbol{x})) \geqslant Q^t_{\boldsymbol{a}}(\mathcal{P}_t(\boldsymbol{x}), \boldsymbol{x})$ do
      Set $t \leftarrow \beta t$ and update $\mathcal{P}_t(\boldsymbol{x})$.
  end while
```

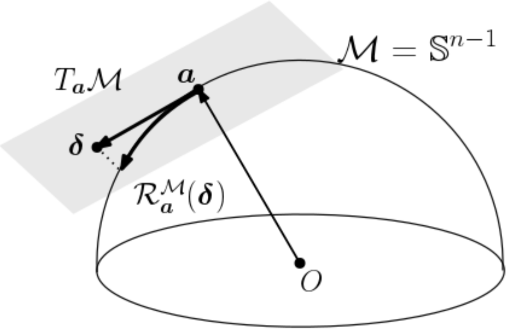

$\mathcal{M} = \mathbb{S}^{n-1}$
$T_{\boldsymbol{a}}\mathcal{M}$
$\boldsymbol{a}$
$\boldsymbol{\delta}$
$\mathcal{R}^{\mathcal{M}}_{\boldsymbol{a}}(\boldsymbol{\delta})$
O

Figure 31: An illustration of manifold optimization with $\mathcal{M} = \mathbb{S}^{n-1}$. $T_{\boldsymbol{a}}\mathcal{M}$ denote the tangent space of $\mathcal{M}$ at the point $\boldsymbol{a}$, and $\mathcal{R}^{\mathcal{M}}_{\boldsymbol{a}}(\boldsymbol{\delta})$ denote the retraction operator at the point $\boldsymbol{a} \in \mathcal{M}$.

where τ is the stepsize, which can be adaptively chosen by the *Riemannian linesearch* based on *Armijo condition* (see Algorithm 4). We use $\operatorname{grad} \psi_{\boldsymbol{x}}(\boldsymbol{a})$ to denote the Riemannian gradient of $\psi_{\boldsymbol{x}}(\boldsymbol{a})$, which is defined over the *tangent space* of $\mathcal{M}$ at the point $\boldsymbol{a}$,

$$\operatorname{grad} \psi_{\boldsymbol{x}}(\boldsymbol{a}) \;=\; \boldsymbol{P}_{T_{\boldsymbol{a}}\mathcal{M}} \nabla \psi_{\boldsymbol{x}}(\boldsymbol{a}),$$

where $\boldsymbol{P}_{T_{\boldsymbol{a}}\mathcal{M}}$ is the projection operator onto the tangent space $T_{\boldsymbol{a}}\mathcal{M}$. On the other hand, $\mathcal{R}^{\mathcal{M}}_{\boldsymbol{a}}(\boldsymbol{\delta})$ denotes the retraction operator, which pulls a vector $\boldsymbol{\delta}$ from the tangent space $T_{\boldsymbol{a}}\mathcal{M}$ to its closest point on the Riemannian manifold $\mathcal{M}$. Figure 31 provides an illustration of the tangent space $T_{\boldsymbol{a}}\mathcal{M}$ and the retraction operator $\mathcal{R}^{\mathcal{M}}_{\boldsymbol{a}}(\boldsymbol{\delta})$ when $\mathcal{M} = \mathbb{S}^{n-1}$, we refer readers to Chapter 3 and 4 of [AMS09] for more detailed definitions.

Algorithm 4 Riemannian linesearch for stepsize τ

```
Input:  $\boldsymbol{a}$, $\boldsymbol{x}$, $\tau_0$, $\eta \in (0.5,1)$, $\beta \in (0,1)$,
Output:  $\tau$, $\mathcal{R}^{\mathcal{M}}_{\boldsymbol{a}}(-\tau \boldsymbol{P}_{T_{\mathcal{M}}} \nabla \psi_{\boldsymbol{x}}(\boldsymbol{a}))$
  Initialize $\tau \leftarrow \tau_0$.
  while $\psi_{\boldsymbol{x}}(\mathcal{R}^{\mathcal{M}}_{\boldsymbol{a}}(-\tau \cdot \operatorname{grad} \psi_{\boldsymbol{x}}(\boldsymbol{a}))) \geqslant \psi_{\boldsymbol{x}}(\boldsymbol{a}) - \tau \cdot \eta \cdot \|\operatorname{grad} \psi_{\boldsymbol{x}}(\boldsymbol{a})\|_2^2$ do
      $\tau \leftarrow \beta\tau$.
  end while
```

B.1.2 Accelerated ADM via momentum method

As aforementioned in Section 4.2, problems in practice often raise additional challenges. The kernel $\boldsymbol{a}_0$ we encounter in practice is often smooth, so that the underlying kernel $\boldsymbol{a}_0$ is of large incoherence $\mu_s(\boldsymbol{a}_0)$. This results in ill-conditioned problems and slow convergence of first-order methods [Nes13b, B$^+$15, Bec17]. As we have discussed in Section 4.2.1, a natural idea to improve solution accuracy and convergence speed is to employ a momentum acceleration strategy, which can be traced back to the *heavy ball* method of Polyak [Pol64].

For our particular problem in Equation (B.1), we apply momentum acceleration to sub-iterations of ADM on both $\boldsymbol{a}$ and $\boldsymbol{x}$. When updating $\boldsymbol{x}$ with $\boldsymbol{a}$ fixed, recall from Equation (4.4) in Section 4.2.1, we modify the original iteration by adding an inertial term $\boldsymbol{w}^{(k)}$, which incorporates information from previous updates.

Algorithm 5 Inertial Alternating Descent Method (iADM)

```
Input:    measurement $\boldsymbol{y}$; initial values $\boldsymbol{a}^{(0)}$, $\boldsymbol{x}^{(0)}$; penalty $\lambda > 0$; momentum parameter $\beta \in [0, 1)$.
Output:   Final iterate $\boldsymbol{a}_\star$, $\boldsymbol{x}_\star$.
  Initialize $k \leftarrow 0$, set $\boldsymbol{a}^{(-1)} = \boldsymbol{a}^{(0)}$, $\boldsymbol{x}^{(-1)} = \boldsymbol{x}^{(0)}$.
  while not converged do
      Fix $\boldsymbol{a}^{(k)}$, and update $\boldsymbol{x}$ using proximal gradient descent with momentum
          $\boldsymbol{w}^{(k)} = \boldsymbol{x}^{(k)} + \beta \cdot \left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right),$
          $\boldsymbol{x}^{(k+1)} = \operatorname{prox}_g^{\lambda t}\left(\boldsymbol{w}^{(k)} - t_k \cdot \nabla \psi_{\boldsymbol{a}^{(k)}}\left(\boldsymbol{w}^{(k)}\right)\right).$
      Fix $\boldsymbol{x}^{(k+1)}$, and update $\boldsymbol{a}$ by using the Riemannian gradient descent with momentum
          $\boldsymbol{z}^{(k)} = \mathcal{R}_{\boldsymbol{a}^{(k)}}^{\mathcal{M}}\left(\beta \cdot \left(\mathcal{R}_{\boldsymbol{a}^{(k-1)}}^{\mathcal{M}}\right)^{-1}\left(\boldsymbol{a}^{(k)}\right)\right),$
          $\boldsymbol{a}^{(k+1)} = \mathcal{R}_{\boldsymbol{z}^{(k)}}^{\mathcal{M}}\left(-\tau_k \cdot \operatorname{grad} \psi_{\boldsymbol{x}^{(k+1)}}\left(\boldsymbol{z}^{(k)}\right)\right),$
      Set $k \leftarrow k + 1$.
  end while
```

Similarly, when we update $\boldsymbol{a}$ with $\boldsymbol{x}$ fixed, we modify the Riemannian gradient step in Equation (B.4) by

$$\boldsymbol{z}^{(k)} = \mathcal{R}_{\boldsymbol{a}^{(k)}}^{\mathcal{M}}\Big(\beta \cdot \underbrace{\left(\mathcal{R}_{\boldsymbol{a}^{(k-1)}}^{\mathcal{M}}\right)^{-1}\left(\boldsymbol{a}^{(k)}\right)}_{\text{inertial term}}\Big), \tag{B.5}$$

$$\boldsymbol{a}^{(k+1)} = \mathcal{R}_{\boldsymbol{z}^{(k)}}^{\mathcal{M}}\left(-\tau_k \cdot \operatorname{grad} \psi_{\boldsymbol{x}^{(k+1)}}\left(\boldsymbol{z}^{(k)}\right)\right).$$

Here, $\left(\mathcal{R}_{\boldsymbol{a}}^{\mathcal{M}}\right)^{-1}(\boldsymbol{b}) : \mathcal{M} \to T_{\boldsymbol{a}}\mathcal{M}$ denotes the *inverse retraction operator*, i.e., $\mathcal{R}_{\boldsymbol{a}}^{\mathcal{M}}\left(\left(\mathcal{R}_{\boldsymbol{a}}^{\mathcal{M}}\right)^{-1}(\boldsymbol{b})\right) \equiv \boldsymbol{b}$. It maps a point $\boldsymbol{b} \in \mathcal{M}$ to the tangent space $T_{\boldsymbol{a}}\mathcal{M}$ of $\boldsymbol{a}$. Intuitively, when β is small, and $\boldsymbol{a}^{(k-1)}$ and $\boldsymbol{a}^{(k)}$ are close, we approximately have

$$\boldsymbol{z}^{(k)} \approx \boldsymbol{a}^{(k)} + \beta \cdot \boldsymbol{\delta}^{(k)}, \quad \boldsymbol{\delta}^{(k)} = \left(\mathcal{R}_{\boldsymbol{a}^{(k-1)}}^{\mathcal{M}}\right)^{-1}\left(\boldsymbol{a}^{(k)}\right) \approx \boldsymbol{a}^{(k)} - \boldsymbol{a}^{(k-1)},$$

which reduces to the standard update in the Euclidean space. The overall algorithmic pipeline is summarized in Algorithm 5, and we term the algorithm inertial ADM (iADM). Similarly to the ADM, we can either set the stepsizes t_k and τ_k to be constants, or choose them via backtracking (linesearch) (Algorithms 3 and 4). The parameter $\beta \in [0, 1)$ controls the weight of inertial term. Empirically, good choices for β lie somewhere between 0.8 to 0.9, and iADM reverts to ADM when β is set to zero. An iteration-dependent schedule for β is also discussed in [PS16]. Unlike the ADM algorithm which decreases the function value $\Psi(\boldsymbol{a}, \boldsymbol{x})$ monotonically, the iterates of iADM exhibit some oscillation effects and they can diverge when β is chosen too large.

B.2 Adaptive update of the penalty λ through the solution path

For sparse deconvolution problems, the parameter λ controls the sparsity of the solution $\boldsymbol{x}$: the larger λ is, the sparser $\boldsymbol{x}$ is produced, and vice versa. [KZLW19] suggests that a good choice could be $\lambda = \mathcal{O}\left(1/\left(\theta n_0\right)\right)$, where θ is the parameter of Bernoulli distribution controlling the sparsity level. However, the sparsity level θ is often not known ahead of time for many real applications, but the choice of λ is crucial for convergence speed and recovery accuracy. In this subsection, we introduce two schemes to adaptively update λ.

(i) **Homotopy continuation method,** which improves both algorithmic convergence speed and recovery accuracy by shrinking the λ through the solution path;

(ii) **Reweighting method**, which improves robustness against noise by adaptively enforcing different penalties λ on different entries of $\boldsymbol{x}$.

Algorithm 6 Homotopy continuation method

```
Input:   Measurement $\boldsymbol{y} \in \mathbb{R}^m$; initial and final sparse penalties $\lambda_0$, $\lambda_\star$ ($\lambda_0 > \lambda_\star$); initialization $(\boldsymbol{a}^{(0)}, \boldsymbol{x}^{(0)})$;
  decay penalty parameter $\eta \in (0,1)$; precision factor $\delta \in (0,1)$ and tolerance $\varepsilon_\star$.
Output:   final solution $(\boldsymbol{a}_\star, \boldsymbol{x}_\star)$.
  Initialize $k \leftarrow 0$, $\lambda^{(0)} \leftarrow \lambda_0$, $\varepsilon^{(0)} \leftarrow \delta\lambda^{(0)}$.
  Set $K \leftarrow \lfloor \log(\lambda_\star/\lambda_0) / \log(\eta) \rfloor$.
  while $k \leqslant K$ do
    Solve Equation (B.1) with $\lambda^{(k)}$ to $\left(\boldsymbol{a}^{(k+1)}, \boldsymbol{x}^{(k+1)}\right)$ of precision $\varepsilon^{(k)}$, using $\left(\boldsymbol{a}^{(k)}, \boldsymbol{x}^{(k)}\right)$ as warm restart.
    Update the parameters: $\lambda^{(k+1)} \leftarrow \eta\lambda^{(k)}$, and $\varepsilon^{(k+1)} \leftarrow \delta\lambda^{(k+1)}$.
    Update $k \leftarrow k+1$.
  end while
  Final round: from $\left(\boldsymbol{a}^{(K+1)}, \boldsymbol{x}^{(K+1)}\right)$, solve Equation (B.1) with penalty $\lambda_\star$ to $(\boldsymbol{a}_\star, \boldsymbol{x}_\star)$ of precision $\varepsilon_\star$.
```

Homotopy continuation method. As discussed in Section 4.2.2, the geometric intuition (see Figure 7) suggests a homotopy continuation approach [HYZ08, WNF09, XZ13], which chooses a solution path for $(\boldsymbol{a}, \boldsymbol{x})$ by adaptively decreasing λ. The overall algorithmic pipeline is summarized in Algorithm 6. More concretely, we start by solving Equation (B.1) with a large penalty λ_0 (e.g., $\lambda_0 = \left\|\check{\boldsymbol{a}}^{(0)} \circledast \boldsymbol{y}\right\|_\infty$), and correspondingly choose a large solution tolerance $\varepsilon = \delta\lambda_0$. The problem in Equation (B.1) can be solved using any local descent methods (e.g., ADM and iADM described in the previous section). Once Equation (B.1) is solved with given λ and ε, we sequentially decrease the penalty λ by η and the solution tolerance ε. We use an approximate solution for $(\boldsymbol{a}, \boldsymbol{x})$ at the end of each stage to warm restart the next stage, and repeatedly solves Equation (B.1) until the target penalty $\lambda_\star$ and precision $\varepsilon_\star$ reached.

In practice, we usually set the parameters $\eta = 0.9$ and $\delta = 10^{-1}$. As we show in Section 5, we observe linear convergence for the *homotopy* continuation method works for SaSD. For SaS-CDL problem, we observe that the homotopy continuation method could occasionally produce duplicated kernels, because a large penalty λ in the beginning stage could attract multiple different kernels to the same solution initially.

Algorithm 7 Reweighting method

```
Input:   Measurement $\boldsymbol{y} \in \mathbb{R}^m$; penalty $\lambda > 0$; initialization $(\boldsymbol{a}^{(0)}, \boldsymbol{x}^{(0)})$.
Output:   final solution $(\boldsymbol{a}_\star, \boldsymbol{x}_\star)$.
  Initialize $k \leftarrow 0$, $\boldsymbol{w}^{(0)} = \mathbf{1}_m$.
  while not converged do
    Solve a weighted subproblem

      $$\min_{\boldsymbol{a},\boldsymbol{x}} \ \Psi^{\boldsymbol{w}^{(k)}}(\boldsymbol{a}, \boldsymbol{x}) \ = \ \psi(\boldsymbol{a}, \boldsymbol{x}) + \lambda \cdot g\left(\boldsymbol{w}^{(k)} \odot \boldsymbol{x}\right), \qquad \boldsymbol{a} \in \mathcal{M} \tag{B.6}$$

  to a solution $\left(\boldsymbol{a}^{(k+1)}, \boldsymbol{x}^{(k+1)}\right)$, by using a warm restart $\left(\boldsymbol{a}^{(k)}, \boldsymbol{x}^{(k)}\right)$.
    Update the weights: Compute $\varepsilon^{(k)}$ using Equation (B.9). Update the weight $\boldsymbol{w}^{(k+1)}$ by

      $$w_i^{(k+1)} \ = \ \frac{1}{\left|x_i^{(k)}\right| + \varepsilon^{(k)}}, \qquad 1 \leqslant i \leqslant m. \tag{B.7}$$

    Update $k \leftarrow k+1$.
  end while
```

Reweighting. Real data is often contaminated by noise, it is preferred to set large λ on zero entries of $\boldsymbol{x}_0$ to suppress the noise, and set small λ on nonzero entries of $\boldsymbol{x}_0$ to promote sparse solutions. This inspires us to introduce the reweighing scheme [CWB08] (see Algorithm 7), the basic idea is to adaptively adjust the penalty λ for each entry of $\boldsymbol{x}$ by considering a weighted variant of the problem (B.1),

$$\min_{\boldsymbol{a},\boldsymbol{x}} \ \Psi^{\boldsymbol{w}}(\boldsymbol{a}, \boldsymbol{x}) \ = \ \psi(\boldsymbol{a}, \boldsymbol{x}) + \lambda \cdot g\left(\boldsymbol{w} \odot \boldsymbol{x}\right), \qquad \boldsymbol{a} \in \mathcal{M}. \tag{B.8}$$

where $\boldsymbol{w} \in \mathbb{R}_+^m$ is the weight and $\odot$ denotes Hadamard products. Taking SaSD problem for instance, when $g(\cdot) = \|\cdot\|_1$, the desired choice of the weight $\boldsymbol{w}$ is expected to be inversely proportional to the magnitude of the true signal $\boldsymbol{x}_0$,

$$w_i \;=\; \begin{cases} \frac{1}{|x_{0,i}|}, & x_{0,i} \neq 0, \\ +\infty, & x_{0,i} = 0, \end{cases} \tag{B.9}$$

which makes $\|\boldsymbol{w} \odot \boldsymbol{x}_0\|_1 = \|\boldsymbol{x}_0\|_0$ The large (actually infinite) entries in w_i force the solution $\boldsymbol{x}$ to concentrate on the indices where w_i is small (actually finite), and by construction these correspond precisely to the indices where x_0 is nonzero. This suggests more generally that large weights could be used to discourage nonzero entries in the recovered signal, while small weights could be used to encourage nonzero entries. Although it is impossible to construct the precise weights in Equation (B.9) without knowing the signal $\boldsymbol{x}_0$ itself, we consider an iterative procedure (as shown in Algorithm 7) that alternates between estimating $\boldsymbol{x}_0$ and refining the weights $\boldsymbol{w}$.

In the following, we take $g(\cdot) = \|\cdot\|_1$ for an example, and provide more details of solving the weighted subproblem in Equation (B.6) in Algorithm 7 with a given $\boldsymbol{w}$. This subproblem can be solved by either ADM or iADM without much modification. The new objective does not affect the update for $\boldsymbol{a}$ when $\boldsymbol{x}$ is fixed. When we update $\boldsymbol{x}$ with $\boldsymbol{a}$ fixed. Notice that the ℓ_1-penalty is separable, so that we have

$$\Psi^{\boldsymbol{w}}(\boldsymbol{a}, \boldsymbol{x}) \;=\; \psi(\boldsymbol{a}, \boldsymbol{x}) + \lambda \cdot \|\boldsymbol{w} \odot \boldsymbol{x}\|_1 \;=\; \psi(\boldsymbol{a}, \boldsymbol{x}) + \sum_{i=1}^{m} \underbrace{\lambda w_i}_{\lambda_i} |x_i| \,.$$

The separability of ℓ_1-penalty implies that we can just update each entry x_i with different penalty λ_i using proximal gradient.

For weight refinement in Equation (B.7), we introduce a scalar $\varepsilon > 0$ in order to provide algorithmic stability, ensuring that a zero-valued component in $\boldsymbol{x}$ does not strictly prohibit a nonzero estimate in the next step. Let $\left\{|x|_{(i)}\right\}$ denote a descent reordering of $\{|x_i|\}$, we empirically set

$$\varepsilon = \max\left\{|x|_{(i_0)}, 10^{-3}\right\},$$

where $i_0 = \lceil n/\log(m/n)\rceil$. In general, the reweighing method tends to be reasonably robust to the choice of small ε.

B.3 Miscellaneous

For the remaining of this part of the appendix, we discuss about various aspects of practical issues in solving Equation (B.1). We first discuss about the initialization strategy for SaSD and SaS-CDL. Second, problems in practice often possess extra structures beyond the general form we considered here (e.g., nonnegativity, bias), and our solutions often require post-processing. We discuss about these issues in more technical details.

Data-driven initialization. For the SaSD problem, we usually initialize $\boldsymbol{x}$ by $\boldsymbol{x}^{(0)} = \mathbf{0}$, so that our initialization is sparse. For the optimization variable $\boldsymbol{a} \in \mathbb{R}^n$, recall from Section 3.2 that it is desirable to obtain an initialization $\boldsymbol{a}^{(0)}$ which is close to $\mathcal{S}_{\mathcal{I}}$ spanned by a few shifts of $\boldsymbol{a}_0$ (as described in Equation (3.3)). When $\boldsymbol{x}_0$ is sparse, Equation (3.4) implies that our measurement $\boldsymbol{y}$ is a linear combination of a few shifts of $\boldsymbol{a}_0$. Therefore, intuitively an arbitrary consecutive length-n_0 truncation $\widehat{\boldsymbol{y}}$ of the data $\boldsymbol{y}$ should be not far away from such a subsphere $\mathcal{S}_{\mathcal{I}}$. As illustrated in Figure 32, one step of the generalized power method [KZLW19]

$$\boldsymbol{a}^{(0)} = \mathcal{P}_{\mathbb{S}^{n-1}} \nabla \left(-\nabla \varphi_{\mathrm{DQ}} \left(\begin{bmatrix} \mathbf{0}_{n-1} \\ \widehat{\boldsymbol{y}} \\ \mathbf{0}_{n-1} \end{bmatrix} \right) \right) \tag{B.10}$$

produces a refined initialization that is very close to $\mathcal{S}_{\mathcal{I}}$ with $|\mathcal{I}| \leqslant \mathcal{O}(\theta n_0)$. Moreover, in practice, we find that even a simpler initialization $\boldsymbol{a}^{(0)}$ (without the power iteration) in Equation (3.5) works stably well for solving SaSD.

When dealing with multiple kernel SaS-CDL problems, we take several independent random truncations of $\boldsymbol{y}$, and repeat the procedure Equation (3.5) to initialize different kernels.

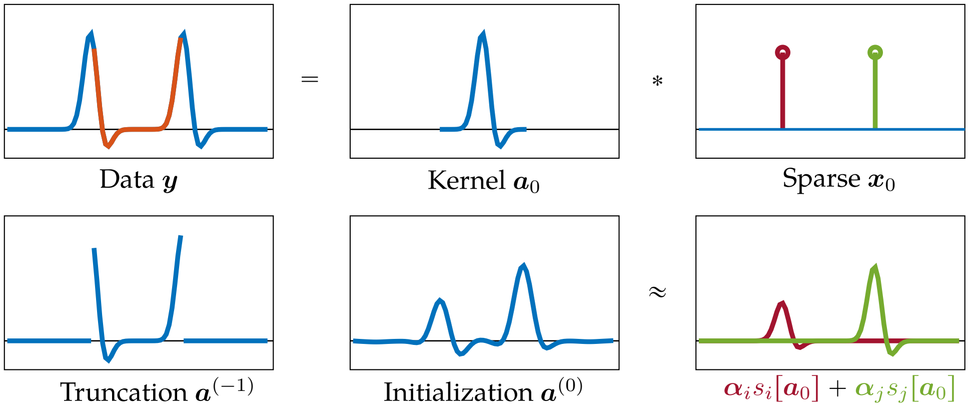


Figure 32: Illustration of data-driven initialization for $\boldsymbol{a}$: using a piece of the observed data $\boldsymbol{y}$ to generate a good initial point $\boldsymbol{a}^{(0)}$. Top: data $\boldsymbol{y} = \boldsymbol{a}_0 \circledast \boldsymbol{x}_0$ is a superposition of shifts of the true kernel $\boldsymbol{a}_0$. Bottom: a length-n_0 window contains pieces of just a few shifts. Bottom middle: one step of the generalized power method approximately fills in the missing pieces, yielding an initialization that is close to a linear combination of shifts of $\boldsymbol{a}_0$ (right).

Nonnegativity constraint. When the signal $\boldsymbol{x}_0$ is nonnegative, we add an extra nonnegativity constraint to the original problem

$$\min_{\boldsymbol{a},\boldsymbol{x}} \Psi(\boldsymbol{a},\boldsymbol{x}) \;=\; \psi(\boldsymbol{a},\boldsymbol{x}) \;+\; \lambda \cdot g(\boldsymbol{x}), \qquad \text{s.t.} \quad \boldsymbol{a} \in \mathcal{M}, \quad \boldsymbol{x} \geqslant 0. \tag{B.11}$$

To enforce the nonnegativity constraint on $\boldsymbol{x}$ in ADM or iADM, we simply modify the proximal gradient step for updating $\boldsymbol{x}$ in Algorithms 2 and 5 by

$$\boldsymbol{x}^{(k+1)} \;=\; \max\left\{ \operatorname{prox}_g^{\lambda t}\left(\boldsymbol{x}^{(k)} - t\nabla\psi_{\boldsymbol{a}}(\boldsymbol{x}^{(k)}) \right),\; 0 \right\},$$

which projects the solution to the nonnegative orthant.

Removing bias components. In practice, the measurement $\boldsymbol{y}$ often contains a constant direct current (DC) component. Taking SaSD as an example, we often have the measurement

$$\boldsymbol{y} \;=\; \boldsymbol{a}_0 \circledast \boldsymbol{x}_0 \;+\; b_0 \mathbf{1}_m,$$

where b_0 describes the magnitude of the DC component. To deal with this issue, it is natural to reformulate the Bilinear Lasso problem in Equation (3.1) as

$$\min_{\boldsymbol{a},\,\boldsymbol{x},b} \Psi(\boldsymbol{a},\boldsymbol{x},b) \;=\; \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{a} \circledast \boldsymbol{x} - b\mathbf{1}_m\right\|_2^2 \;+\; \lambda\left\|\boldsymbol{x}\right\|_1, \quad \text{s.t.} \quad \boldsymbol{a} \in \mathbb{S}^{n-1},$$

and modify the optimization methods accordingly. For ADM and iADM in Algorithms 2 and 5, we initialize b as the mean value of the sequence $\boldsymbol{y}$, and update $\boldsymbol{a}$ and $\boldsymbol{x}$ in the original way with b fixed. For optimizing variable b, we simply add an extra step after updating $\boldsymbol{a}$ and $\boldsymbol{x}$ by

$$b^{(k)} = \frac{1}{m}\left\langle \mathbf{1}_m, \boldsymbol{y} - \boldsymbol{a}^{(k)} \circledast \boldsymbol{x}^{(k)} \right\rangle.$$

Shift correction. The shift symmetry implies that we can only solve sparse deconvolution problems up to a shift ambiguity. However, as predicting the precise activation locations $\boldsymbol{x}_0$ could be mission critical in many applications, post-processing is often needed to correct the shift ambiguity by exploiting the structure of the data $\boldsymbol{y}$.

As aforementioned in Section 3, our optimization space $n = 3n_0 - 2$ for the kernel $\boldsymbol{a}_0$ is larger the original dimension of $\boldsymbol{a}_0$, due to shift truncations. We need to truncation the solution $\boldsymbol{a}$ produced by our algorithm to obtain an approximation of the original kernel. A natural idea is to find a length-n_0 subvector (submatrix) of the produced solution $\boldsymbol{a}_\star$ that maximizes the Frobenius norm across all length-n_0 subvectors. Therefore, we simply shift $\boldsymbol{a}_\star$ so that the chosen length-n_0 window is in the top left, and remove other zero-padding entries if needed. Correspondingly, the solution $\boldsymbol{x}_\star$ can be corrected by shifting the same amount of length in the opposite direction. However, using Frobenius norm could be unstable when large noise presents. Algorithm 8 presents an alternative approach based on the reconstruction error, which turns out to be more reliable in some cases.

Algorithm 8 Shift correction

```
Input:   observation $\boldsymbol{y}$, optimal solution $(\boldsymbol{a}_\star, \boldsymbol{x}_\star) \in \mathbb{R}^n \times \mathbb{R}^m$.
Output:  Solution $(\boldsymbol{a}, \boldsymbol{x}) \in \mathbb{R}^{n_0} \times \mathbb{R}^m$ after shift correction.
  for $i = 1 : 2n_0 + 1$ do
      Set $\widehat{\boldsymbol{a}} = \boldsymbol{\iota}_{n \to n_0} \mathrm{s}_{-i+1}[\boldsymbol{a}_\star]$, $\widehat{\boldsymbol{x}} = \mathrm{s}_{i-1}[\boldsymbol{x}_\star]$;
      Compute $\widehat{\boldsymbol{y}}_i = \widehat{\boldsymbol{a}} \circledast \widehat{\boldsymbol{x}}$;
  end for
  Find $i_\star = \arg\min_i \{\|\widehat{\boldsymbol{y}}_i - \boldsymbol{y}\|_2\}$;
  Set $\boldsymbol{a} = \boldsymbol{\iota}_{n \to n_0} \mathrm{s}_{-i_\star+1}[\boldsymbol{a}_\star]$, $\boldsymbol{x} = \mathrm{s}_{i_\star-1}[\boldsymbol{x}_\star]$.
```

C Implementation details for SaSD and SaS-CDL

Finally, we provide missing implementation details of the proposed ADM and iADM algorithms for both SaSD and SaS-CDL. We show how to solve these problems for both cases when the observation is 1-dimensional (1D) and 2-dimensional (2D).

C.1 Technical details for solving 1D problems

C.1.1 Implementations details for SaSD

We optimize Equation (B.1) in Appendix B for the SaSD problem,

$$\min_{\boldsymbol{a},\boldsymbol{x}} \; \Psi(\boldsymbol{a},\boldsymbol{x}) \;=\; \underbrace{\frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{a}\circledast\boldsymbol{x}\right\|_2^2}_{\psi(\boldsymbol{a},\boldsymbol{x})} + \lambda\cdot\underbrace{\left\|\boldsymbol{x}\right\|_1}_{g(\boldsymbol{x})}, \qquad \text{s.t.} \quad \boldsymbol{a}\in\underbrace{\mathbb{S}^{n-1}}_{\mathcal{M}},$$

where $\psi(\boldsymbol{a},\boldsymbol{x}) = \frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{a}\circledast\boldsymbol{x}\right\|_2^2$, $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$, and $\mathcal{M} = \mathbb{S}^{n-1}$. Next, we provide missing implementation details (e.g., exact forms of the gradients) of SaSD for ADM and iADM in Algorithm 2 and Algorithm 5.

Update $\boldsymbol{x}$ with $\boldsymbol{a}$ fixed. For the proximal gradient step on $\boldsymbol{x}$ in Equation (B.2), the proximal operator of $g(\cdot) = \|\cdot\|_1$ is the soft thresholding operator

$$\operatorname{prox}_{\|\cdot\|_1}^{\lambda t}(z) \;=\; \mathcal{S}_{\lambda t}(z), \qquad \mathcal{S}_{\lambda t}(z) \;=\; \operatorname{sign}(z)\cdot\max\left\{|z|-\lambda t, 0\right\}.$$

The gradient of $\psi_{\boldsymbol{a}}(\boldsymbol{x})$ is

$$\nabla\psi_{\boldsymbol{a}}(\boldsymbol{x}) \;=\; \check{\boldsymbol{a}}\circledast\left(\boldsymbol{a}\circledast\boldsymbol{x}-\boldsymbol{y}\right).$$

Update $\boldsymbol{a}$ with $\boldsymbol{x}$ fixed. For the Riemannian manifold $\mathcal{M} = \mathbb{S}^{n-1}$, its tangent space $T_{\boldsymbol{a}}\mathbb{S}^{n-1}$ and the projection onto $T_{\boldsymbol{a}}\mathbb{S}^{n-1}$ are

$$T_{\boldsymbol{a}}\mathbb{S}^{n-1} \;=\; \left\{\boldsymbol{z}\in\mathbb{R}^n \;\middle|\; \boldsymbol{a}^\top\boldsymbol{z}=0\right\}, \qquad \mathcal{P}_{T_{\boldsymbol{a}}\mathbb{S}^{n-1}} = \mathcal{P}_{\boldsymbol{a}^\perp} = \boldsymbol{I}_n - \frac{1}{\|\boldsymbol{a}\|_2^2}\boldsymbol{a}\boldsymbol{a}^\top.$$

For the Riemannian gradient step on $\boldsymbol{a}$ presented in Equation (B.4), the Riemannian gradient of $\psi_{\boldsymbol{x}}(\boldsymbol{a})$ over $\mathbb{S}^{n-1}$ is

$$\operatorname{grad} \psi_{\boldsymbol{x}}(\boldsymbol{a}) \;=\; \mathcal{P}_{\boldsymbol{a}^\perp} \nabla \psi_{\boldsymbol{x}}(\boldsymbol{a}), \qquad \nabla \psi_{\boldsymbol{x}}(\boldsymbol{a}) \;=\; \boldsymbol{\iota}_{n \to m}^{*} \check{\boldsymbol{x}} \circledast (\boldsymbol{a} \circledast \boldsymbol{x} - \boldsymbol{y}) \,.$$

In addition, for $\mathcal{M} = \mathbb{S}^{n-1}$, the retraction operator $\mathcal{R}_{\boldsymbol{a}}^{\mathbb{S}^{n-1}}(\boldsymbol{\delta})$ for $\boldsymbol{\delta} \in T_{\boldsymbol{a}}\mathbb{S}^{n-1}$, and its inversion $\left(\mathcal{R}_{\boldsymbol{a}}^{\mathbb{S}^{n-1}}\right)^{-1}(\boldsymbol{b})$ for $\boldsymbol{b} \in \mathbb{S}^{n-1}$, can be specified as

$$\mathcal{R}_{\boldsymbol{a}}^{\mathbb{S}^{n-1}}(\boldsymbol{\delta}) \;=\; \boldsymbol{a} \cdot \cos\left(\|\boldsymbol{\delta}\|_2\right) + \frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|_2} \sin\left(\|\boldsymbol{\delta}\|_2\right), \quad \left(\mathcal{R}_{\boldsymbol{a}}^{\mathbb{S}^{n-1}}\right)^{-1}(\boldsymbol{\delta}) \;=\; \frac{\alpha}{\sin \alpha} \mathcal{P}_{\boldsymbol{a}^\perp} \boldsymbol{\delta}, \tag{C.1}$$

where $\alpha = \arccos\left(\boldsymbol{a}^\top \boldsymbol{\delta}\right)$.

C.1.2 Implementations details for SaS-CDL

Since the SaSD problem can be considered a special case of SaS-CDL with $N = 1$, the derivations we obtained for SaSD can be easily extended to SaS-CDL. Recall from Equation (4.6), the general objective in Equation (B.1) can specified as

$$\min_{\boldsymbol{A},\, \boldsymbol{X}} \; \Psi(\boldsymbol{A}, \boldsymbol{X}) = \underbrace{\frac{1}{2} \left\| \boldsymbol{y} - \sum_{k=1}^{N} \boldsymbol{a}_k \circledast \boldsymbol{x}_k \right\|_2^2}_{\psi(\boldsymbol{A},\boldsymbol{X})} + \lambda \underbrace{\|\boldsymbol{X}\|_1}_{g(\boldsymbol{X})}, \qquad \text{s.t.} \quad \boldsymbol{A} \in \underbrace{\mathcal{OB}(n, N)}_{\mathcal{M}},$$

where $\mathcal{OB}(n, N)$ is called the *oblique* manifold with

$$\mathcal{OB}(n, N) = \left\{ \boldsymbol{Z} \in \mathbb{R}^{n \times N} \;\mid\; \boldsymbol{Z} = \begin{bmatrix} \boldsymbol{z}_1 & \cdots & \boldsymbol{z}_N \end{bmatrix}, \; \boldsymbol{z}_i \in \mathbb{S}^{n-1}, \; 1 \leqslant i \leqslant N \right\} \;=\; \underbrace{\mathbb{S}^{n-1} \times \cdots \times \mathbb{S}^{n-1}}_{N},$$

which is essentially a product of N spheres. Next, we provide missing implementation details of solving SaS-CDL by ADM and iADM in Algorithm 2 and Algorithm 5.

Update $\boldsymbol{X}$ with $\boldsymbol{A}$ fixed. First, for the proximal gradient step on $\boldsymbol{X}$ presented in Equation (B.2), similarly we have $\operatorname{prox}_{\|\cdot\|_1}^{\lambda t}(z) = \mathcal{S}_{\lambda t}(z)$ and

$$\begin{aligned} \nabla \psi_{\boldsymbol{A}}(\boldsymbol{X}) \;&=\; \begin{bmatrix} \nabla_{\boldsymbol{x}_1} \psi_{\boldsymbol{A}}(\boldsymbol{X}) & \nabla_{\boldsymbol{x}_2} \psi_{\boldsymbol{A}}(\boldsymbol{X}) & \cdots & \nabla_{\boldsymbol{x}_N} \psi_{\boldsymbol{A}}(\boldsymbol{X}) \end{bmatrix}, \\ \nabla_{\boldsymbol{x}_i} \psi_{\boldsymbol{A}}(\boldsymbol{X}) \;&=\; \check{\boldsymbol{a}}_i \circledast \left(\sum_{j=1}^{N} \boldsymbol{a}_j \circledast \boldsymbol{x}_j - \boldsymbol{y} \right), \quad 1 \leqslant i \leqslant N. \end{aligned}$$

Second, for the Riemannian manifold $\mathcal{M} = \mathcal{OB}(n, N)$, its tangent space $T_{\boldsymbol{A}}\mathcal{OB}(n, N)$ and the projection onto $T_{\boldsymbol{A}}\mathcal{OB}(n, N)$ are

$$T_{\boldsymbol{A}}\mathcal{OB}(n, N) \;=\; T_{\boldsymbol{a}_1}\mathbb{S}^{n-1} \times \cdots \times T_{\boldsymbol{a}_N}\mathbb{S}^{n-1}, \qquad \mathcal{P}_{T_{\boldsymbol{A}}\mathcal{OB}}(\boldsymbol{Z}) = \begin{bmatrix} \boldsymbol{P}_{\boldsymbol{a}_1^\perp} \boldsymbol{z}_1 & \boldsymbol{P}_{\boldsymbol{a}_2^\perp} \boldsymbol{z}_1 & \cdots & \boldsymbol{P}_{\boldsymbol{a}_N^\perp} \boldsymbol{z}_N \end{bmatrix}.$$

Update $\boldsymbol{A}$ with $\boldsymbol{X}$ fixed. For the Riemannian gradient step on $\boldsymbol{A}$ presented in Equation (B.4), we have the Riemannian gradient of $\psi_{\boldsymbol{X}}(\boldsymbol{A})$ over $\mathcal{OB}(n, N)$ as

$$\begin{aligned} \operatorname{grad} \psi_{\boldsymbol{X}}(\boldsymbol{A}) \;&=\; \left[\operatorname{grad}_{\boldsymbol{a}_1} \psi_{\boldsymbol{X}}(\boldsymbol{A}), \operatorname{grad}_{\boldsymbol{a}_2} \psi_{\boldsymbol{X}}(\boldsymbol{A}), \cdots, \operatorname{grad}_{\boldsymbol{a}_N} \psi_{\boldsymbol{X}}(\boldsymbol{A}) \right], \\ \operatorname{grad}_{\boldsymbol{a}_i} \psi_{\boldsymbol{X}}(\boldsymbol{A}) \;&=\; \mathcal{P}_{\boldsymbol{a}_i^\perp} \nabla_{\boldsymbol{a}_i} \psi_{\boldsymbol{X}}(\boldsymbol{A}) = \mathcal{P}_{\boldsymbol{a}_i^\perp} \boldsymbol{\iota}_{n \to m}^{*} \check{\boldsymbol{x}}_i \circledast \left(\sum_{j=1}^{N} \boldsymbol{a}_j \circledast \boldsymbol{x}_j - \boldsymbol{y} \right), \; 1 \leqslant i \leqslant N, \end{aligned}$$

and the retraction operator $\mathcal{R}_{\boldsymbol{A}}^{\mathcal{OB}(n,N)}(\boldsymbol{\Delta})$ for $\boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{\delta}_1 & \boldsymbol{\delta}_2 & \cdots & \boldsymbol{\delta}_N \end{bmatrix} \in T_{\boldsymbol{A}}\mathcal{OB}(n, N)$ can be specified as

$$\mathcal{R}_{\boldsymbol{A}}^{\mathcal{OB}(n,N)}(\boldsymbol{\Delta}) \;=\; \begin{bmatrix} \mathcal{R}_{\boldsymbol{a}_1}^{\mathbb{S}^{n-1}}(\boldsymbol{\delta}_1) & \mathcal{R}_{\boldsymbol{a}_2}^{\mathbb{S}^{n-1}}(\boldsymbol{\delta}_2) & \cdots & \mathcal{R}_{\boldsymbol{a}_N}^{\mathbb{S}^{n-1}}(\boldsymbol{\delta}_N) \end{bmatrix},$$

where $\mathcal{R}_{\boldsymbol{a}}^{\mathbb{S}^{n-1}}(\boldsymbol{\delta})$ is the retractor operator over the sphere as introduced in Equation (C.1). The inverse retraction $\left(\mathcal{R}_{\boldsymbol{A}}^{\mathcal{OB}(n,N)}\right)^{-1}(\boldsymbol{B})$ for $\boldsymbol{B} \in \mathcal{OB}(n, N)$ can be constructed similarly by using Equation (C.1).

C.2 Brief technical details of solving 2D problems

The derivative of 2D problems is slightly different from the 1D case, which we briefly introduce below.

Implementations details for SaSD. Let $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{A}}_0 \boxast \boldsymbol{\mathcal{X}}_0 \in \mathbb{R}^{m_1 \times m_2}$ be a 2D circular convolution of a kernel $\boldsymbol{\mathcal{A}}_0 \in \mathbb{R}^{n_1 \times n_2}$ and activation map $\boldsymbol{\mathcal{X}}_0 \in \mathbb{R}^{m_1 \times m_2}$, where $\boxast$ denotes the 2D circular convolution. For the 2D SaSD problem, we consider

$$\min_{\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{X}}} \Psi\left(\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{X}}\right) \;=\; \underbrace{\frac{1}{2}\left\|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{A}} \boxast \boldsymbol{\mathcal{X}}\right\|_{\mathrm{F}}^2}_{\psi(\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{X}})} \;+\; \underbrace{\lambda \left\|\boldsymbol{\mathcal{X}}\right\|_1}_{g(\boldsymbol{\mathcal{X}})}, \quad \text{s.t.} \quad \underbrace{\left\|\boldsymbol{\mathcal{A}}\right\|_{\mathrm{F}} = 1}_{\mathcal{M}},$$

where $\left\|\cdot\right\|_{\mathrm{F}}$ denotes the Frobenius norm. Similarly, we have the gradients

$$\begin{aligned} \nabla_{\boldsymbol{\mathcal{X}}} \psi_{\boldsymbol{\mathcal{A}}}\left(\boldsymbol{\mathcal{X}}\right) &= \widetilde{\boldsymbol{\mathcal{A}}} \boxast \left(\boldsymbol{\mathcal{A}} \boxast \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\right), \\ \nabla_{\boldsymbol{\mathcal{A}}} \psi_{\boldsymbol{\mathcal{X}}}\left(\boldsymbol{\mathcal{A}}\right) &= \boldsymbol{\iota}_{n_1 \to m_1}^{*} \widetilde{\boldsymbol{\mathcal{X}}} \boxast \left(\boldsymbol{\mathcal{A}} \boxast \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\right) \boldsymbol{\iota}_{n_2 \to m_2}, \end{aligned}$$

where $\widetilde{\boldsymbol{\mathcal{Z}}}$ denotes a flip operator that flips a matrix $\boldsymbol{\mathcal{Z}}$ both vertically and horizontally, i.e.,

$$\widetilde{\boldsymbol{\mathcal{Z}}}(i,j) = \boldsymbol{\mathcal{Z}}(m_1 - i + 1, m_2 - j + 1).$$

Note that $\widetilde{\boldsymbol{\mathcal{Z}}} \boxast \boldsymbol{\mathcal{V}}$ is essentially 2D auto-correlation of $\boldsymbol{\mathcal{Z}}$ and $\boldsymbol{\mathcal{V}}$, so that we can rewrite

$$\begin{aligned} \nabla_{\boldsymbol{\mathcal{X}}} \psi_{\boldsymbol{\mathcal{A}}}\left(\boldsymbol{\mathcal{X}}\right) &= \mathcal{F}^{-1}\left[\mathcal{F}^{*}\left(\boldsymbol{\mathcal{A}}\right) \odot \mathcal{F}\left(\boldsymbol{\mathcal{A}} \boxast \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\right)\right] \\ \nabla_{\boldsymbol{\mathcal{A}}} \psi_{\boldsymbol{\mathcal{X}}}\left(\boldsymbol{\mathcal{A}}\right) &= \boldsymbol{\iota}_{n_1 \to m_1}^{*} \mathcal{F}^{-1}\left[\mathcal{F}^{*}\left(\boldsymbol{\mathcal{X}}\right) \odot \mathcal{F}\left(\boldsymbol{\mathcal{A}} \boxast \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}\right)\right] \boldsymbol{\iota}_{n_2 \to m_2}, \end{aligned}$$

where $\mathcal{F}$ denotes the 2D Fourier transform operator, and $\mathcal{F}^{*}$ is its adjoint operator. Finally, we have the Riemannian gradient

$$\operatorname{grad} \psi_{\boldsymbol{\mathcal{X}}}\left(\boldsymbol{\mathcal{A}}\right) \;=\; \boldsymbol{P}_{\boldsymbol{\mathcal{A}}^{\perp}} \nabla_{\boldsymbol{\mathcal{A}}} \psi_{\boldsymbol{\mathcal{X}}}\left(\boldsymbol{\mathcal{A}}\right), \quad \boldsymbol{P}_{\boldsymbol{\mathcal{A}}^{\perp}} \boldsymbol{\mathcal{Z}} \;=\; \boldsymbol{\mathcal{Z}} - \frac{\boldsymbol{\mathcal{A}}}{\left\|\boldsymbol{\mathcal{A}}\right\|_{\mathrm{F}}^2} \left\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{Z}} \right\rangle.$$

The retraction operator and its inversion remain the same as Equation (C.1).

Implementations details for SaS-CDL. For the multiple kernel deconvolution problem $\boldsymbol{\mathcal{Y}} = \sum_{k=1}^{N} \boldsymbol{\mathcal{A}}_{0k} \boxast \boldsymbol{\mathcal{X}}_{0k}$, let the optimization variable $\overline{\boldsymbol{\mathcal{A}}} \in \mathbb{R}^{n_1 \times n_2 \times N}$ and $\overline{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{m_1 \times m_2 \times N}$ be 3-way tensors, with

$$\overline{\boldsymbol{\mathcal{A}}}(:,:,k) = \boldsymbol{\mathcal{A}}_k, \quad \overline{\boldsymbol{\mathcal{X}}}(:,:,k) = \boldsymbol{\mathcal{X}}_k, \quad 1 \leqslant k \leqslant N.$$

Similar to the 1D case in Equation (4.6), we optimize the following problem

$$\min_{\overline{\boldsymbol{\mathcal{A}}},\, \overline{\boldsymbol{\mathcal{X}}}} \Psi\left(\overline{\boldsymbol{\mathcal{A}}}, \overline{\boldsymbol{\mathcal{X}}}\right) \underbrace{\frac{1}{2}\left\|\boldsymbol{\mathcal{Y}} - \sum_{k=1}^{N} \boldsymbol{\mathcal{A}}_k \boxast \boldsymbol{\mathcal{X}}_k\right\|_{\mathrm{F}}^2}_{\psi(\overline{\boldsymbol{\mathcal{A}}},\overline{\boldsymbol{\mathcal{X}}})} + \lambda \underbrace{\left\|\overline{\boldsymbol{\mathcal{X}}}\right\|_1}_{g(\overline{\boldsymbol{\mathcal{X}}})}, \quad \underbrace{\text{s.t. } \left\|\boldsymbol{\mathcal{A}}_k\right\|_{\mathrm{F}} = 1 \; (1 \leqslant k \leqslant N)}_{\mathcal{M}}.$$

The gradient of $\psi_{\overline{\boldsymbol{\mathcal{A}}}}\left(\overline{\boldsymbol{\mathcal{X}}}\right)$ and $\psi_{\overline{\boldsymbol{\mathcal{X}}}}\left(\overline{\boldsymbol{\mathcal{A}}}\right)$ can be computed in a similar manner as SaSD. Let $\nabla \psi_{\overline{\boldsymbol{\mathcal{A}}}}\left(\overline{\boldsymbol{\mathcal{X}}}\right)$ and $\nabla \psi_{\overline{\boldsymbol{\mathcal{X}}}}\left(\overline{\boldsymbol{\mathcal{A}}}\right)$ denote the gradient of $\psi_{\overline{\boldsymbol{\mathcal{A}}}}\left(\overline{\boldsymbol{\mathcal{X}}}\right)$ and $\psi_{\overline{\boldsymbol{\mathcal{X}}}}\left(\overline{\boldsymbol{\mathcal{A}}}\right)$, then we have

$$\begin{aligned} \nabla \psi_{\overline{\boldsymbol{\mathcal{A}}}}\left(\overline{\boldsymbol{\mathcal{X}}}\right)(:,:,i) &= \widetilde{\boldsymbol{\mathcal{A}}}_i \boxast \left(\sum_{j=1}^{N} \boldsymbol{\mathcal{A}}_j \boxast \boldsymbol{\mathcal{X}}_j - \boldsymbol{\mathcal{Y}}\right), \quad 1 \leqslant i \leqslant N \\ \nabla \psi_{\overline{\boldsymbol{\mathcal{X}}}}\left(\overline{\boldsymbol{\mathcal{A}}}\right)(:,:,i) &= \boldsymbol{\iota}_{n_1 \to m_1}^{*} \widetilde{\boldsymbol{\mathcal{X}}}_i \boxast \left(\sum_{j=1}^{N} \boldsymbol{\mathcal{A}}_j \boxast \boldsymbol{\mathcal{X}}_j - \boldsymbol{\mathcal{Y}}\right) \boldsymbol{\iota}_{n_2 \to m_2}, \quad 1 \leqslant i \leqslant N. \end{aligned}$$

The Riemannian gradient $\operatorname{grad} \psi_{\overline{\boldsymbol{\mathcal{X}}}}\left(\overline{\boldsymbol{\mathcal{A}}}\right)$, and the retraction operator can be generalized from 1D case in a very similar fashion. We omit the details here.