Supervised learning of analysis-sparsity priors with automatic differentiation

Hashem Ghanem, Joseph Salmon, Nicolas Keriven, Samuel Vaiter

Supervised learning of analysis-sparsity priors with automatic differentiation

Hashem Ghanem, Joseph Salmon, Nicolas Keriven, and Samuel Vaiter

***Abstract*—Sparsity priors are commonly used in denoising and image reconstruction. For analysis-type priors, a dictionary defines a representation of signals that is likely to be sparse. In most situations, this dictionary is not known, and is to be recovered from pairs of ground-truth signals and measurements, by minimizing the reconstruction error. This defines a hierarchical optimization problem, which can be cast as a bi-level optimization. Yet, this problem is unsolvable, as reconstructions and their derivative *w.r.t.* the dictionary have no closed-form expression. However, reconstructions can be iteratively computed using the Forward-Backward splitting (FB) algorithm. In this paper, we approximate reconstructions by the output of the aforementioned FB algorithm. Then, we leverage automatic differentiation to evaluate the gradient of this output *w.r.t.* the dictionary, which we learn with projected gradient descent. Experiments show that our algorithm successfully learns the 1D Total Variation (TV) dictionary from piecewise constant signals. For the same case study, we propose to constrain our search to dictionaries of 0-centered columns, which removes undesired local minima and improves numerical stability.**

***Index Terms*—Sparsity, dictionary learning, total variation, bi-level optimization, automatic differentiation.**

I. Introduction

DENOISING is a widely-tackled problem that emerges in many fields, where the goal is to restore signals from noisy observations. This includes that the model of the imaging system is *a priori* known: $\boldsymbol{y} = \boldsymbol{w} + \boldsymbol{\varepsilon}$, where $\boldsymbol{w}, \boldsymbol{y} \in \mathbb{R}^p$ are the true and the measured signals, respectively, and $\boldsymbol{\varepsilon} \in \mathbb{R}^p$ is additive noise. In addition, a prior hypothesis on the nature of the signal, or an embedding of it, might be available, like sparsity [10]. Such extra knowledge can be incorporated in the optimization process to get better reconstructions (*e.g.,* a higher Signal to Noise Ratio (SNR)).

Sparsity priors exist in two forms [6]: *i)* synthesis (traditional) sparsity where $\boldsymbol{w} = \boldsymbol{D}\boldsymbol{u}$, $\boldsymbol{D}$ is a linear operator, and $\boldsymbol{u}$ is sparse; *ii)* analysis sparsity where $\boldsymbol{v} = \boldsymbol{D}^\top \boldsymbol{w} \in \mathbb{R}^m$ is sparse. This paper is interested in the latter. As a convex surrogate, this prior is enforced on $\boldsymbol{w}$ by adding the term $\|\boldsymbol{D}^\top \boldsymbol{w}\|_1$ to the (generally quadratic) loss function [9], where $\|\cdot\|_1$ is the ℓ_1 norm. Putting all together, the problem consists in finding: $\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{w}\|_2^2 + \lambda \|\boldsymbol{D}^\top \boldsymbol{w}\|_1$ for some regularization amplitude parameter $\lambda \geq 0$. The linear operator $\boldsymbol{D}$ is either user-defined, or learnt directly from data.

The main problem we tackle in this work is to extract both $\boldsymbol{D}$ and λ from data with supervised learning. Having that $\lambda \|\boldsymbol{D}^\top \boldsymbol{w}\|_1 = \|(\lambda \boldsymbol{D})^\top \boldsymbol{w}\|_1$, this problem is equivalent to extracting the product $\lambda \boldsymbol{D}$ as one object, thus we stop notating λ explicitly from now on and keep $\boldsymbol{D}$. To avoid trivial solutions and undesired local minima, it is common to restrict the search space to an admissible set $\mathcal{C} \subset \mathbb{R}^{p \times m}$, where dictionaries have a specific property. In [12] for instance, $\boldsymbol{D}$ is forced to be orthogonal, *i.e.,* $\mathcal{C} = \{\boldsymbol{D}; \boldsymbol{D}\boldsymbol{D}^\top = \boldsymbol{I}_p\}$. In [11], $\boldsymbol{D}$ is constrained to be a convolution dictionary. However, it is still challenging to find an admissible set that performs well in all applications of this problem [17]. In this work, we don't commit to find such universal set. Formally, we suppose we have a dataset $(\boldsymbol{y}_l, \boldsymbol{w}_l)_{l=1}^L$, that includes L pairs of measurements and their associated ground truth signals. Knowing such pairs, our task is to find the operator $\boldsymbol{D}$ that minimizes the mean squared error between reconstructions and ground truth signals:

$$\hat{\boldsymbol{D}} \in \underset{\boldsymbol{D} \in \mathbb{R}^{p \times m}}{\arg\min}\, \mathcal{E}(\boldsymbol{D}) = \sum_{l=1}^{L} \left\| \hat{\boldsymbol{w}}\left(\boldsymbol{D}, \boldsymbol{y}_l\right) - \boldsymbol{w}_l \right\|_2^2 + \iota_{\mathcal{C}}(\boldsymbol{D}) \quad \text{(1a)}$$

s.t.

$$\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y}) = \underset{\boldsymbol{w} \in \mathbb{R}^p}{\arg\min}\, \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{w}\|_2^2 + \|\boldsymbol{D}^\top \boldsymbol{w}\|_1 \ , \quad \text{(1b)}$$

where $\mathcal{C}$ is an admissible set of dictionaries, and $\iota_{\mathcal{C}}$ is the indicator function: $\iota_{\mathcal{C}}(\boldsymbol{D}) = 0$ if $\boldsymbol{D} \in \mathcal{C}$ and $+\infty$ otherwise.

One can recast Equation (1) as a *bilevel optimization problem*: in the outer problem, we learn the model $\boldsymbol{D}$ as in Eq. (1a) while in the inner problem, we denoise measurements following Eq. (1b). We already know that the inner part can be solved applying the Forward-Backward splitting (FB) algorithm on the dual problem [5]. However, due to the ℓ_1 norm, neither the solution nor its gradient *w.r.t.* $\boldsymbol{D}$ have closed-form expressions. Thus, the minimizer $\hat{\boldsymbol{D}}$ cannot be derived analytically nor obtained with gradient-based methods.

To illustrate how our algorithm can recover a dictionary $\hat{\boldsymbol{D}}$, we consider the well-known problem of *1D piecewise constant signals reconstruction* as a case-study [5], where this prior indicates that $(w_2 - w_1, \ldots, w_1 - w_p)^\top$ is sparse. The estimator is often written as an instance of Eq. (1b), with $\boldsymbol{D} = \boldsymbol{D}_{TV}$ is the dictionary associated to the 1D Total Variation (TV) regularization: for all $i \in \{1, \ldots, p\}$; $\boldsymbol{D}_{i,i} = -1, \boldsymbol{D}_{i+1,i} = 1$, and 0 otherwise, up to rescaling.

Contribution We approximately recover the analysis-sparsity operator $\hat{\boldsymbol{D}}$ by: *i)* replacing the true minimizer $\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y})$ by the output of the FB algorithm applied on the dual problem; *ii)* deploying *automatic differentiation*, a technique capable of evaluating the gradient of an algorithm *w.r.t.* input variables, to solve Eq. (1a) with projected gradient descent. We empirically show that our method recovers the TV dictionary $\boldsymbol{D}_{TV}$ from piecewise constant signals. Finally, for the same case study, we reduce the admissible set $\mathcal{C}$ to dictionaries with

The authors acknowledge the support of ANR Grava ANR-18-CE40-0005, ANR GRandMa ANR-21-CE23-0006 and ANER BFC RAGA.

HG is with CNRS, IMB, Univ. de Bourgogne; JS is with Univ. Montpellier; NK is with CNRS, GIPSA-lab; SV is with CNRS, LJAD, Univ. Côte d'Azur.

columns summing up to zero, and empirically prove that this increases stability and extracts the TV operator with higher quality than previous methods.

Related work The bilevel problem was first posed in [11], where the authors smoothed the ℓ_1 norm so that the derivative of $\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y})$ *w.r.t.* $\boldsymbol{D}$ has a closed-form expression. Then, they applied gradient descent to find a local minimizer $\boldsymbol{D}$. However, the sought for sparsity is degraded with this methodology. In [14], a relaxation regime of ℓ_1 with the smooth ℓ_2 norm is adopted, with a relaxation parameter easy to assign. Recently in [10], a formula of $\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y})$ is obtained under some conditions. This formula includes inverting a large matrix, which is computed iteratively while Automatic Differentiation (AD) is used to evaluate gradients. In [16], and without a proof of convergence, they restrict $\boldsymbol{D}$ to have unit columns norm, that also satisfy $\boldsymbol{D}\boldsymbol{D}^\top = \boldsymbol{I}$, in order to avoid trivial solutions (*e.g.,* repeated columns). The work in [4] solves the 2D piecewise signals reconstruction problem using AD, constrained by learning *convolution-type* dictionaries of kernels with small support, which improves the quality of standard 2D TV. Such strong constraints are not considered in our work, however we will see that simple column-centering suffices to learn a high-quality dictionary in the 1D version of this problem.

II. Automatic Differentiation (AD)

To compute gradients, one often manually writes down its analytical expression. Yet, this can only be performed for functions with closed-form expression, which is not the case in Eq. (1b). An alternative is symbolic differentiation, automatically performed by computer tools like Mathematica [7] when dealing with syntax tree expressions. A third approach consists in approximating gradients using numerical differentiation. It is easy to implement though exposed to round-off errors [8] and expensive in case of a high number of variables (here $p \times m$).

Automatic Differentiation (AD), that is of interest in this work, mitigates the previous drawbacks. AD manipulates computation flow in a computer program, with all numerical computations reduced to compositions of elementary operations, for which derivative rules are known [15]. In such computer programs, a value must be assigned to each input variable, thus, any operation in the code will result in a variable with a numerical value. During the execution of the program, AD consists in [1]: 1) keeping a trace to all intermediate variables that are dependent on the ones we want to differentiate for; 2) once an operation is performed on a dependent variable, say v_i, to evaluate another v_j, directly computing the derivative value dv_j/dv_i; 3) accumulating derivatives in step 2 through the chain rule. This gives the derivative value of the whole composition *w.r.t.* a chosen variable. AD can trace numerical computations in recursion functions and in control-flow (*if, while, for*) statements. Thus, AD efficiently differentiates not only closed-form formulas, but also implemented algorithms. This makes AD suitable for gradient-based optimization. AD can be implemented in two ways: *forward* mode and *reverse* mode. In our work, we adopt the reverse mode[1]. Let us assume having $y = (y_1, \dots, y_m)$ as a function of the variable $x = (x_1, \dots, x_n)$, with v as an intermediate variable. The backward scheme takes in input the vector $\bar{\boldsymbol{y}} = (\bar{y}_1, \dots, \bar{y}_m)$, and accumulates gradients as follows: first $\bar{\boldsymbol{v}} = \boldsymbol{J}_y^\top(v)\bar{\boldsymbol{y}}$, then $\bar{\boldsymbol{x}} = \boldsymbol{J}_v^\top(x)\bar{\boldsymbol{v}}$, where $(\boldsymbol{J}_y(v))_{i,j} = dy_i/dv_j$. The output $\bar{\boldsymbol{x}}$ is nothing but the transpose Jacobian-vector product $\boldsymbol{J}_y^\top(x)\bar{\boldsymbol{y}}$.

III. Proposed algorithm

We solve the *dual* problem of Eq. (1b) with Forward-Backward splitting (FB). Using Automatic Differentiation, we obtain gradients of the previous FB algorithm *w.r.t.* $\boldsymbol{D}$. These gradients are used to learn a local minimum $\hat{\boldsymbol{D}}$ using gradient descent, while projecting $\boldsymbol{D}$ on the admissible set $\mathcal{C}$ at every iteration.

A. Deriving the dual problem of Eq. (1b)

The term $\|\boldsymbol{D}^\top \boldsymbol{w}\|_1$ in Eq. (1b) is neither differentiable *w.r.t.* $\boldsymbol{w}$, nor has a simple proximal operator that can be efficiently performed. Hence, we cannot apply the gradient descent or the FB algorithm directly to evaluate $\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y})$. However, the latter is possible if we tackle this optimization from the dual perspective [13]. In fact, one can prove that Eq. (1b) is equivalent to its dual problem [5]:

$$\hat{\boldsymbol{z}}(\boldsymbol{D}, \boldsymbol{y}) = \underset{\boldsymbol{z} \in \mathbb{R}^m}{\arg\min} \ \frac{1}{2}\big\|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{y}\big\|_2^2 \ + \ \iota_{B_\infty}(\boldsymbol{z}) \ , \quad (2)$$

where $B_\infty = \{\boldsymbol{z} \in \mathbb{R}^m; \ |z_i| \leq 1, \ \forall i \in [m]\}$ is the unit ball of the ℓ_∞ norm. The target recovery $\hat{\boldsymbol{w}}$ is then given by:

$$\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{D} \ \hat{\boldsymbol{z}}(\boldsymbol{D}, \boldsymbol{y}). \quad (3)$$

B. Solving the dual problem with the FB algorithm

Fortunately, the term $\iota_{B_\infty}(\boldsymbol{z})$ has a simple proximal function given by Π_{B_∞}: the orthogonal projection on the ball B_∞. So indeed, we solve Eq. (2) with an accelerated FB algorithm, namely the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [2]. At each iteration we update $\boldsymbol{z}$ as follows:

$$\begin{aligned} \boldsymbol{z}_{i+1} &= \operatorname{prox}_{\iota_{B_\infty}}\big(\boldsymbol{q}_i - \eta_1 \nabla(\frac{1}{2}\big\|\boldsymbol{D}\boldsymbol{q}_i - \boldsymbol{y}\big\|_2^2)\big) \\ &= \Pi_{B_\infty}\big(\boldsymbol{q}_i - \eta_1 \boldsymbol{D}^\top(\boldsymbol{D}\boldsymbol{q}_i - \boldsymbol{y})\big) \end{aligned} \quad (4)$$

s.t. $\boldsymbol{q}_1 = \boldsymbol{z}_0$ is the initialization of $\boldsymbol{z}$, η_1 is the step size, and:

$$\begin{aligned} \boldsymbol{q}_{i+1} &= \boldsymbol{z}_i + \frac{t_i - 1}{t_{i+1}}(\boldsymbol{z}_i - \boldsymbol{z}_{i-1}) \\ t_{i+1} &= \frac{1}{2}(1 + \sqrt{1 + 4t_i^2}) \ ; t_1 = 1. \end{aligned} \quad (5)$$

To conclude with the inner part: having the matrix $\boldsymbol{D}$ and the vector $\boldsymbol{y}$ as *input*, one can compute $\hat{\boldsymbol{z}}(\boldsymbol{D}, \boldsymbol{y})$ with a sufficient number of updates as in Eq. (4), then get $\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y})$ *in output* using Eq. (3).

[1] this is the mode implemented in the PyTorch framework

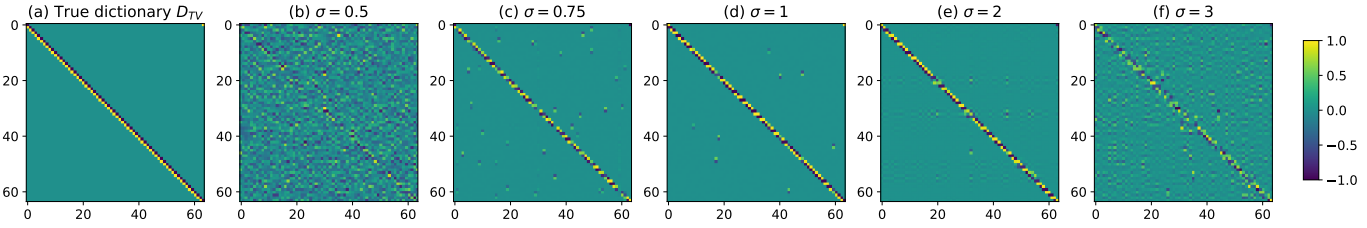

Fig. 1: Performance of our projected gradient descent algorithm 1, *w.r.t.* to noise level. We plot a sorted view of the dictionary $\hat{\boldsymbol{D}}$. We rescale all dictionaries to $[-1, 1]$ for a better visualization of the structure recovered in $\hat{\boldsymbol{D}}$. (a) $\boldsymbol{D}_{TV}$ our algorithm is expected to learn. From (b) to (f): $\hat{\boldsymbol{D}}$ for different values of σ, the standard deviation of noise. Jumps in the dataset have the same amplitude: $0 \to 10$ or $10 \to 0$.

C. Outer loop design to learn the dictionary

Let us first notice that when the admissible set $\mathcal{C} = \mathbb{R}^{p\times m}$, the solution $\hat{\boldsymbol{D}}$ is *not unique* as $\|\cdot\|_1$ is invariant to the coefficients order in a vector, *e.g.*, $\|(u_1, u_2)^\top\|_1 = \|(u_2, u_1)^\top\|_1$. Thus, permuting columns in $\boldsymbol{D}$ will lead to the same $\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y})$ in Eq. (1b), which means the same cost $\mathcal{E}(\boldsymbol{D})$. In this paper, we consider any solution that is a minimizer of $\mathcal{E}(\boldsymbol{D})$. We also assume that the second dimension of $\boldsymbol{D}$ is given, while optimizing for it might be the subject of a future work.

Towards our goal, we use AD to get the gradient of the mean squared error (MSE) term in $\mathcal{E}(\boldsymbol{D})$, by tracing the FB algorithm that solves Eq. (1b). In fact, to reduce computational cost, we do not compute the full MSE but sample a batch of training signals at each iteration (see Alg. 1). We denote this term by $\mathcal{E}_{\text{MSE}}(\boldsymbol{D})$. Since proving the convergence of AD's Jacobian to the variational one is complicated in such non-smooth setting, we replace the true reconstruction $\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y})$ with the output produced by the FB algorithm, and empirically show that it is a good proxy. For simplicity, we keep the same notation for this output. We randomly initialize $\boldsymbol{D}_0$, then we start each iteration t by setting *PyTorch AD framework* to record operations on $\boldsymbol{D}_t$. We can then compute the output $\hat{\boldsymbol{w}}(\boldsymbol{D}_t, \boldsymbol{y})$, and $\mathcal{E}_{\text{MSE}}(\boldsymbol{D}_t)$. Now, we use the PyTorch AD to get the gradient $\nabla\mathcal{E}_{\text{MSE}}(\boldsymbol{D}_t)$, and we update the estimated dictionary as

$$\boldsymbol{D}_{t+1} = \boldsymbol{D}_t - \eta_2 \nabla\mathcal{E}_{\text{MSE}}(\boldsymbol{D}_t) \ , \tag{6}$$

where $\eta_2 > 0$ is a step size. Lastly, we project $\boldsymbol{D}_t$ on the admissible set $\mathcal{C}$ by computing $\Pi_{\mathcal{C}}(\boldsymbol{D}_t)$. We obtain Algorithm 1.

Algorithm 1: AD-based projected gradient descent

Input: $\{(\boldsymbol{w}_l, \boldsymbol{y}_l)\}_{l\in\{1,\dots,L\}}$: dataset.
Output: $\hat{\boldsymbol{D}}$: minimizer to $\mathcal{E}(\boldsymbol{D})$ in Eq. (1a).
Params: $m, \eta_1, \eta_2, max\_itr1, max\_itr2, batch\_sz$
Algorithm:
Initialize $\boldsymbol{D}$ *iid* from $\mathcal{N}(0, 10^{-4})$.
Set PyTorch AD to track computations on $\boldsymbol{D}$.
for t *in* $\{0, \dots, max\_itr2 - 1\}$*:*
do
  $\mathcal{E}_{MSE}(\boldsymbol{D}) \leftarrow 0$
  for $l \in \{t * batch\_sz, \dots, (t+1) * batch\_sz - 1\}$
  do
    Initialize $\boldsymbol{q}$ *iid* from $\mathcal{N}(0, 1)$.
    for i *in* $\{1, \dots, max\_itr1\}$*:*
    do
      $\boldsymbol{z} \leftarrow \Pi_{B_\infty}\big(\boldsymbol{q} - \eta_1 \boldsymbol{D}^\top(\boldsymbol{D}\boldsymbol{q} - \boldsymbol{y}_l)\big)$.
      Update $\boldsymbol{q}$ as in Eq. (5).
    $\hat{\boldsymbol{w}}(\boldsymbol{D}, \boldsymbol{y}_l) \leftarrow \boldsymbol{y}_l - \boldsymbol{D}\boldsymbol{z}$.
    $\mathcal{E}_{MSE}(\boldsymbol{D}) \leftarrow \mathcal{E}_{MSE}(\boldsymbol{D}) + \left\|\hat{\boldsymbol{w}}\left(\boldsymbol{D}, \boldsymbol{y}_l\right) - \boldsymbol{w}_l\right\|_2^2$.
  $\nabla\mathcal{E}_{MSE}(\boldsymbol{D}) \leftarrow$ PyTorch AD gradient.
  $\boldsymbol{D} \leftarrow \Pi_{\mathcal{C}}\big(\boldsymbol{D} - \frac{\eta_2}{batch\_sz}\nabla\mathcal{E}_{MSE}(\boldsymbol{D})\big)$.
Return $\hat{\boldsymbol{D}} = \boldsymbol{D}$

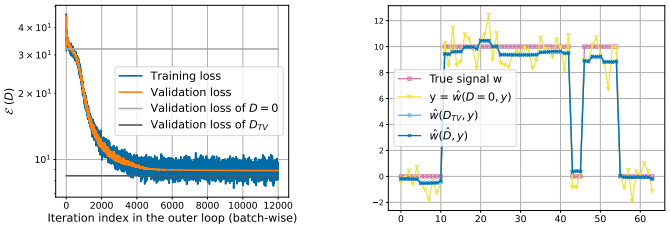

Fig. 2: Performance curves of the learnt $\hat{\boldsymbol{D}}$ in the case $\sigma = 1$ in Fig. 1. Left: training and validation losses as in Eq. (1a) as a function of the iteration index. We plot the loss incurred by $\boldsymbol{D}_{TV}$ and $\boldsymbol{D} = \boldsymbol{0}$ on the validation set. Right: Recovered signals with $\boldsymbol{D}_{TV}$, $\boldsymbol{D} = \boldsymbol{0}$ and $\hat{\boldsymbol{D}}$ as in Eq. (1b) from a random signal $\boldsymbol{y}$ from the validation set.

IV. Experiments

We consider the 1D piecewise constant signals reconstruction problem, with $p = m = 64$ in all experiments. Ground-truth examples $\boldsymbol{w}$ are generated s.t. they have 4 discontinuities, thus 4 constant pieces. The coefficients where discontinuities take place are randomly chosen in each $\boldsymbol{w}$. Their amplitude is either fixed ($0 \to 10$ or $10 \to 0$), or randomly sampled s.t. they happen between two values in $[0, 10]$, to be mentioned when necessary. Observations $\boldsymbol{y}$ are constructed by adding a noise vector to each ground-truth signal, such vector is sampled from $\mathcal{N}(0, \sigma^2\boldsymbol{I}_p)$, where σ varies through experiments. The "true" underlying dictionary $\boldsymbol{D}_{TV}$ is shown in Fig. 1 (a), up to permuting its columns, and to rescaling with λ, which we compute in each experiment with a grid search solving Eq. (1a) *w.r.t.* λ. Matrices $\hat{\boldsymbol{D}}$ shown in this section have their columns sorted by magnitudes, to ease the comparison with $\boldsymbol{D}_{TV}$. We consider *stochastic gradient descent* updates with batch size 64. Training set size: 640000, validation set size: 256. We adopt random white noise initialization with varying variance.

Denoiser setup: The value of η_1 in Eq. (4) is assigned auto-

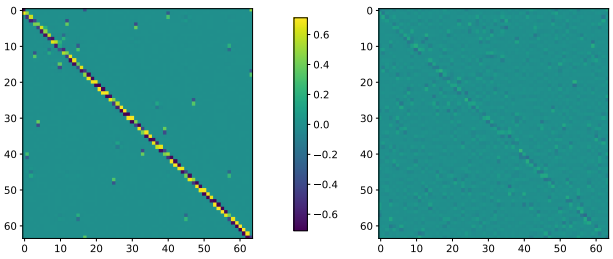

Fig. 3: Showing the difference made by our proposed admissible set $\mathcal{C}$. We plot $\hat{\boldsymbol{D}}$ produced by our algorithm. Left: with our proposed $\mathcal{C}$. Right: with $\mathcal{C} = \mathbb{R}^{p\times p}$. Dataset: Jumps in the dataset can randomly occur between any two values in $[0, 10]$, thus they have random amplitudes. $\sigma = 0.5$.

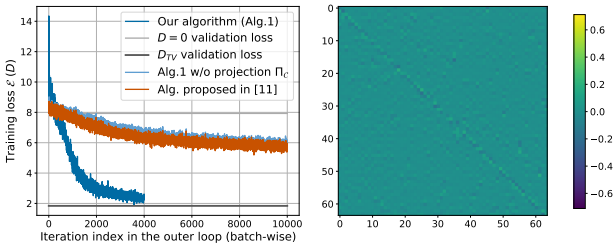

Fig. 4: Benchmark against the unconstrained optimization with ℓ_1-smoothening proposed in [11], with identical problem setting to Fig. 3. Left: training loss curves of different algorithms. Right: $\hat{\boldsymbol{D}}$ produced by the algorithm in [11]. Dataset: same as in Fig. 3.

matically while guaranteeing convergence of $\boldsymbol{z}_i$. This value is $\eta_1 = 0.95/\|\boldsymbol{D}^\top\boldsymbol{D}\|_2$, where $\|\boldsymbol{D}^\top\boldsymbol{D}\|_2$ is the Lipschitz constant of $\nabla\frac{1}{2}\|\boldsymbol{D}\boldsymbol{z}-\boldsymbol{y}\|_2^2$. In all experiments, a tolerance threshold is used to determine the number of iterations, specifically, we assume convergence if $\|\boldsymbol{z}_{i+1} - \boldsymbol{z}_i\|_\infty/\|\boldsymbol{z}_i\|_\infty < 10^{-4}$.

A. Projection proposed for this case study

We reduce the admissible set $\mathcal{C}$ in our proposed algorithm 1 to dictionaries whose columns sum up to zero, *i.e.,* $\mathcal{C} = \{\boldsymbol{D}|\mathbf{1}_p^\top\boldsymbol{D} = \mathbf{0}_m\}$. This property is seen in the prior operator $\boldsymbol{D}_{TV}$ our algorithm is expected to learn, and in the more general family of problems known as graph total variation [3]. This very simple prior greatly improves the results by filtering out many local minima. Different priors for other cases will be the goal of future investigations. Referring to the c-th column of $\boldsymbol{D}$ by $\boldsymbol{D}[:, c]$, and by $\mathrm{mean}(\boldsymbol{D}[:, c])$ to the mean value of this column, we project $\boldsymbol{D}_{t+1}$ as follows:

$$\boldsymbol{D}_{t+1}[:, c] = \boldsymbol{D}_{t+1}[:, c] - \mathrm{mean}\left(\boldsymbol{D}_{t+1}[:, c]\right),\ \forall c \in [m]. \quad (7)$$

In Fig. 3 and Fig. 4(left), we show the essential role of the centering projection in our proposed algorithm. We run our algorithm twice: *i)* with the projection; *ii)* ignoring the projection step; on the same dataset, and we plot the learned dictionary $\hat{\boldsymbol{D}}$ in both cases. Discontinuities in the used dataset are of random magnitudes (between any two values in [0,10]), and the noise has a standard deviation $\sigma = 0.5$. Our algorithm coupled with the projection successfully captures $\boldsymbol{D}_{TV}$-like structure from the dataset, unlike when the projection is not considered, which shows its capability in this problem setting.

B. Sensitivity w.r.t. to the noise level

When our observations are noise-free, *i.e.,* $\sigma = 0$ and $\boldsymbol{y} = \boldsymbol{w}$, it is clear $\boldsymbol{D} = \mathbf{0}$ is the optimal dictionary, as no regularization is required to retrieve the true signal. Therefore, when σ is small, we don't expect our algorithm to learn the structure in $\boldsymbol{D}_{TV}$, since in such case, its magnitude is of the same level as numerical errors, see Fig. 1 (b).

On the other hand, when the noise level is high, the piecewise constant prior is degraded and is poorly seen in observations. As a result, the learnt dictionary is a distorted version of $\boldsymbol{D}_{TV}$, as in Fig. 1 (f).

In between, it is important to verify that our algorithm is stable, and can extract $\boldsymbol{D}_{TV}$ out from data. As shown in Fig. 1 (c-e), this is indeed true when $\sigma \in [0.75, 2]$, which spans a non-trivial range in SNR scale $[12, 89]$, given that our signals $\boldsymbol{w}$ take values in $\{0, 10\}$.

For $\sigma = 1$, we show in Fig. 2 the evolution of training/validation losses as a function of the batch index through training, while benchmarking it against the same loss incurred when $\boldsymbol{D} = \boldsymbol{D}_{TV}$ with optimal λ and $\boldsymbol{D} = \mathbf{0}$ as a reference. We also present the same benchmark but *w.r.t.* the denoising performance, as in Eq. (1b), applied on a signal from the validation set.

C. Benchmark against the algorithm proposed in [11]

In Fig. 4, we compare our output to the one produced by the unconstrained learning algorithm based on smoothening ℓ_1 in [11]. We fix the ℓ_1-smoothening parameter $\epsilon = 10^{-3}$. The dataset has discontinuities of random magnitudes in [0,10], and the noise standard deviation $\sigma = 0.5$, *i.e.,* same setup as in Fig. 3. Unlike our projected gradient descent, the opponent algorithm fails to inspect the $\boldsymbol{D}_{TV}$ structure from data.

V. Conclusion and Future Work

To alleviate the difficulty of deriving gradients to learn analysis-sparsity dictionaries, that are optimized through a bilevel problem, we proposed to make use of automatic differentiation, a technique shown to have high proficiency in machine learning and deep learning. In the absence of a theoretical link between the automatic differentiation and the analytical one in our setting, our experiments on the piecewise constant signals reconstruction problem showed a proof-of-concept, and a promising methodology that can be applied in other problems setting. For the same case study, we also incorporated a simple column-wise centering projection, which significantly increased the stability of the algorithm and the quality of the learned dictionary.

Further investigations can be done to generalize our algorithm to 2D signals, particularly the 2D piecewise constant signals reconstruction, while trying to combine other constraints. Moreover, it would be interesting to study the problem again when the dimensions of the dictionary are not given, for instance to learn graph incidence matrices.

REFERENCES

[1] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18, 2018.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009.

[3] P. Berger, G. Hannak, and G. Matz. Graph signal recovery via primal-dual algorithms for total variation minimization. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):842–855, 2017.

[4] A. Chambolle and T. Pock. Learning consistent discretizations of the total variation. 2020.

[5] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[6] M. Elad, P. Milanfar, and R. Rubinstein. Analysis versus synthesis in signal priors. *Inverse problems*, 23(3):947, 2007.

[7] J. Grabmeier and E. Kaltofen. *Computer Algebra Handbook: Foundations, Applications, Systems*. Springer Science & Business Media, 2003.

[8] M. E. Jerrell. Automatic differentiation and interval arithmetic for estimation of disequilibrium models. *Computational Economics*, 10(3):295–316, 1997.

[9] L. Mancera and J. Portilla. L0-norm-based sparse representation through alternate projections. In *2006 International Conference on Image Processing*, pages 2089–2092, 2006.

[10] M. T. McCann and S. Ravishankar. Supervised learning of sparsity-promoting regularizers for denoising. *arXiv preprint arXiv:2006.05521*, 2020.

[11] G. Peyré and J. M. Fadili. Learning analysis sparsity priors. In *Sampta'11*, pages 4–pp, 2011.

[12] S. Ravishankar and Y. Bresler. Sparsifying transform learning with efficient optimal updates and convergence guarantees. *IEEE Transactions on Signal Processing*, 63 (9):2389–2404, 2015.

[13] R. T. Rockafellar. *Conjugate duality and optimization*. SIAM, 1974.

[14] P. Sprechmann, R. Litman, T. Ben Yakar, A. M. Bronstein, and G. Sapiro. Supervised sparse analysis and synthesis operators. *Advances in Neural Information Processing Systems*, 26:908–916, 2013.

[15] A. Verma. An introduction to automatic differentiation. *Current Science*, pages 804–807, 2000.

[16] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies. Analysis operator learning for overcomplete cosparse representations. In *2011 19th European Signal Processing Conference*, pages 1470–1474. IEEE, 2011.

[17] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies. Noise aware analysis operator learning for approximately cosparse signals. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5409–5412. IEEE, 2012.