IMA Journal of Numerical Analysis (2025) **00**, 1–29
https://doi.org/10.1093/imanum/draf037

Learning a Gaussian mixture for sparsity regularization in inverse problems

Giovanni S. Alberti
MaLGa Center, Department of Mathematics, University of Genoa, Via Dodecaneso 35, I-16146 Genova, Italy

Luca Ratti*
Department of Mathematics, University of Bologna, Piazza di Porta San Donato 5, I-40126 Bologna, Italy
*Corresponding author: luca.ratti5@unibo.it

and

Matteo Santacesaria and Silvia Sciutto
MaLGa Center, Department of Mathematics, University of Genoa, Via Dodecaneso 35, I-16146 Genova, Italy

[Received on 2 February 2024; revised on 28 March 2025]

In inverse problems it is widely recognized that the incorporation of a sparsity prior yields a regularization effect on the solution. This approach is grounded on the *a priori* assumption that the unknown can be appropriately represented in a basis with a limited number of significant components while most coefficients are close to zero. This occurrence is frequently observed in real-world scenarios, such as with piecewise smooth signals. In this study we propose a probabilistic sparsity prior formulated as a mixture of degenerate Gaussians, capable of modelling sparsity with respect to a generic basis. Under this premise we design a neural network that can be interpreted as the Bayes estimator for linear inverse problems. Additionally, we put forth both a supervised and an unsupervised training strategy to estimate the parameters of this network. To evaluate the effectiveness of our approach we conduct a numerical comparison with commonly employed sparsity-promoting regularization techniques, namely Least Absolute Shrinkage and Selection Operator (LASSO), group LASSO, iterative hard thresholding and sparse coding/dictionary learning. Notably, our reconstructions consistently exhibit lower mean square error values across all one-dimensional datasets utilized for the comparisons, even in cases where the datasets significantly deviate from a Gaussian mixture model.

Keywords: sparse optimization; statistical learning; inverse problems; Gaussian mixture; dictionary learning; Bayes estimator.

1. Introduction

A signal is said to be *sparse* if it can be represented as a linear combination of a small number of vectors in a known family (e.g. a basis or a frame). Sparsity has played a major role in the last decades in signal processing and statistics as a way to identify key quantities and find low-dimensional representations of many families of signals, including natural images (Candès *et al.*, 2006; Donoho, 2006; Mallat, 2008). In inverse problems sparse optimization has had a significant impact in many applications, notably in accelerated magnetic resonance imaging via compressed sensing (Lustig *et al.*, 2008).

The key ingredient needed in sparse optimization is the knowledge of a suitable dictionary that can well represent the unknown quantities with as few elements as possible. If these quantities cannot

be described analytically and are measured from experiments, given a sufficient number of samples, machine learning techniques can be used to infer the dictionary (Lee *et al.*, 2006; Horesh & Haber, 2009; Huang *et al.*, 2012). Dictionary learning in the context of sparse optimization, i.e. sparse coding, can be seen as the problem of finding the best change of basis that makes the data as sparse as possible. Thus, it is a special case of the problem of learning a regularization functional (Lunz *et al.*, 2018; Schwab *et al.*, 2019; Alberti *et al.*, 2021) and the more general framework of learning operators between function spaces (see (de Hoop *et al.*, 2023; Kovachki *et al.*, 2023) and references therein).

In this work we propose a new approach for dictionary learning for sparse optimization motivated by the problem of learning a regularization functional for solving inverse problems. We consider a linear inverse problem in a finite-dimensional setting:

$$y = Ax + \varepsilon, \tag{1}$$

where $x \in \mathbb{R}^n$, $\varepsilon \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times n}$. We assume a statistical perspective: namely, x and ε are realizations of the random variables X and E on $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. Let ρ_X and ρ_E denote their probability distributions.

Given some partial knowledge about the probability distributions of X and E we aim to recover an estimator $R\colon \mathbb{R}^m \to \mathbb{R}^n$ that has statistical guarantees and encodes information on the sparsity of X. While sparsity is well established in a deterministic setting, there is no clear consensus on how to define a probability distribution of sparse vectors. One option comes from the Bayesian interpretation of ℓ^1 minimization: it can be seen as a maximum *a posteriori* estimator under a Laplacian prior in the presence of additive white Gaussian noise. However, this is unsatisfactory if the aim is to generate truly sparse signals. Recently, it has been shown that specific hierarchical Bayesian models can be used as priors for sparse signals (Calvetti *et al.*, 2019, 2023).

Instead, in this work we consider a mixture of degenerate Gaussians as a statistical model for a distribution of sparse vectors X. The dimensions of the degenerate support of each Gaussian naturally represent the sparsity levels of the signals. We also assume that the noise is Gaussian.

To find the estimator R we consider a statistical learning framework. We employ the mean squared error (MSE; also referred to as expected risk), namely,

$$L(R) = \mathbb{E}_{X\sim\rho_X, E\sim\rho_E} \|R(AX+E) - X\|^2_{\mathbb{R}^m} = \mathbb{E}_{(X,Y)\sim\rho} \|R(Y) - X\|^2_{\mathbb{R}^m}, \tag{2}$$

where ρ is the joint probability distribution of (X, Y) on $\mathbb{R}^n \times \mathbb{R}^m$. By minimizing L among all possible measurable functions R we get the so-called minimum MSE (MMSE) estimator, or Bayes estimator.

The Bayes estimator cannot be directly employed as a solution to the statistical inverse problem because it requires the knowledge of the full probability distribution ρ. Supervised learning techniques can be used to find a good approximation when a finite amount of data is available. However, a suitable class of functions (hypothesis space) must be chosen to obtain a good approximation.

Our main contribution is the development of supervised and unsupervised learning schemes to approximate the Bayes estimator when a (degenerate) Gaussian Mixture model is assumed as a prior on X. This is done due to an explicit expression for the MMSE, which is usually not available for general probability distributions underlying the data and the noise. The proposed training schemes are based on the observation that the explicit expression of the MMSE can be seen as a two-layer neural network, which can be easily trained using stochastic optimization and back-propagation. In particular, we find a strong similarity of our network with a single self-attention layer, a building block of the well-known

transformer architecture (Vaswani *et al.*, 2017): our results provide a novel statistical interpretation of the attention mechanism.

As carefully presented in Section 4 the supervised technique performs an empirical risk minimization within the hypothesis class $\mathcal{H}$ of Bayes estimators of Gaussian Mixture models, the means and the covariance matrices of the mixture being learnable parameters. The unsupervised one, instead, approximates the Bayes estimator relying on suitable clustering techniques and empirical estimates for the means and covariances.

After the training our network/algorithm can be used as an alternative to classical sparse optimization approaches. We perform extensive numerical comparisons of our approach with well-known sparsity-promoting algorithms: Least Absolute Shrinkage and Selection Operator (LASSO), group LASSO, iterative hard thresholding (IHT) and sparse coding/dictionary learning. The tests are done by performing denoising and deblurring on several datasets of (discretized) one-dimensional signals. In all experiments our method outperforms the others, in the MSE. Our method can be seen as a new paradigm for both sparse optimization and dictionary learning for sparse data.

The main limitation of our strategy is represented by the number of parameters that need to be estimated to identify the optimal regularizer. It is easy to show that this number grows as Ln^2, being L the number of components of the Gaussian mixture model. If the unknown X is simply assumed to be s-sparse, namely, that its support is contained in any possible s-dimensional coordinate subspace in $\mathbb{R}^n$, we should consider $L = \binom{n}{s}$, which is prohibitive in most applications. This is why our method is best suited for cases of *structured* sparsity, in which the support of X consists of a restricted number of subspaces, which is the case of group sparsity (Yuan & Lin, 2006), for instance.

The paper is organized as follows. In Section 2 we provide an overview of the concept of Bayes estimator and recall an explicit formula in the case of linear inverse problems with Gaussian mixture prior. Afterwards, Section 3 is dedicated to the interpretation of this estimator as a neural network, highlighting its potential utility for sparse recovery. We introduce a supervised and an unsupervised approach for learning the neural network representation of the Bayes estimator in Section 4. In Section 5 we describe some baseline sparsity-promoting algorithms, against which our methods are evaluated. This is done in Section 6, where we conduct numerical comparisons on denoising and deblurring problems across various datasets of one-dimensional signals. Conclusions are drawn in Section 7. Further contents are reported in the appendices, including the proof of the main theoretical result (Appendix A), implementation details about the baseline algorithms (Appendix B) and extended numerical comparisons (Appendix C).

2. Statistical learning and Bayes estimator for inverse problems

In this section we introduce our approach and motivation, collecting some results that are already known about inverse problems, statistical estimation and mixtures of Gaussian random variables. The main elements of novelty are reported in the next section.

We recall the inverse problem introduced in (1), which we can also interpret as a realization of the following equation involving the random variables X on $\mathbb{R}^n$ and E, Y on $\mathbb{R}^m$:

$$Y = AX + E. \tag{3}$$

Throughout the paper we make the following assumptions on the random variables X and E.

ASSUMPTION 2.1. We assume that

- The distribution of the noise, ρ_E, is known and zero-mean, i.e. $\mathbb{E}[E] = \mathbf{0}$. The covariance Σ_E is invertible.
- The random variables X and E are independent.

Our goal is to identify a function $R\colon \mathbb{R}^m \to \mathbb{R}^n$ (which we will refer to as a regularizer or an estimator) such that the reconstructions $R(Ax + \varepsilon)$ are close to the corresponding x, when x, ε are sampled from X, E and the squared error $\|R(Ax + \varepsilon) - R(x)\|^2$ is used as a metric.

This problem is in general different from the deterministic formulation of inverse problems: the recovery of a single x from $y = Ax + \varepsilon$. The task we are considering also differs from the one of Bayesian inverse problems. In that scenario, indeed, the goal is not to retrieve a function R (or a regularized solution $R(Y)$), but rather to determine a probability distribution, in particular the conditional distribution of $X|Y$, and to ensure that, as the noise E converges to $\mathbf{0}$ in some suitable sense, this (posterior) distribution converges to the distribution of X.

Our perspective on the inverse problem is instead rather related to the statistical theory of estimation, or statistical inference. Indeed, based on some partial knowledge about the probability distributions of X and E we aim to recover an estimate $R(Y)$ of X close to X w.r.t. the MSE. In particular, we set our discussion at the intersection of two areas of statistical inference, namely the Minimum Mean Square Error estimation and parametric estimation.

The first field is determined by the choice of (2) as the metric according to which the random variables $R(Y)$ and X are considered close to each other or not. The minimizer of L among all possible measurable functions R is defined as the MMSE estimator, or Bayes estimator:

$$R^\star \in \arg\min\{L(R) : \ R\colon \mathbb{R}^m \to \mathbb{R}^n, R \text{ measurable}\}. \tag{4}$$

As it is easy to show (see Cucker & Smale (2002)) the solution to such a problem is the conditional mean of X given Y, so that

$$R^\star(y) = \mathbb{E}[X|Y = y] = \int_{\mathbb{R}^n} x\, p(x|y)\, \mathrm{d}x.$$

The Bayes estimator $R^\star$ is not always the most straightforward choice. Indeed, its exact computation requires the knowledge of the joint probability distribution ρ, which depends on the distributions of the noise E and of X. Notice that the distribution of X is not a user-crafted prior, but should encode the full statistical model of the ground truth X, which may not be fully accessible when solving the inverse problem.

To overcome this issue one usually assumes to have access to a rather large set of training data, namely of pairs (x_j, y_j) sampled from the joint distribution ρ. It is then possible to substitute the integrals appearing in the definition of L and of $R^\star$ by means of Monte-Carlo quadrature rules. This approach is usually referred to as *supervised statistical learning*. In order to avoid overfitting and preserve stability one typically restricts the choice of the possible estimators R to a specific class of functions $\mathcal{H}$, which introduces an implicit bias on the estimator and possibly a regularizing effect. This leads us to the framework of parametric estimation.

In Alberti *et al.* (2021), for example, the hypothesis class $\mathcal{H}$ consists of a family of affine functions in y, solutions of suitably parametrized quadratic minimization problems. Minimizing over such a class

(which corresponds to the task of learning the optimal generalized Tikhonov regularizer) is motivated by theoretical reasons: if the distributions of X and E are Gaussian then $R^\star$ belongs to $\mathcal{H}$. However, in general the minimizer of L within $\mathcal{H}$ would deviate from $R^\star$, and such a bias could be in some cases undesired. On top of that, a significant outcome of the analysis in Alberti *et al.* (2021) is a closed-form expression of the optimal regularizer (also in infinite dimension), which in particular can be computed if the mean and the covariance of X are known. This paves the way to an *unsupervised statistical learning* approach: namely, to approximate such an estimator one would not need a training set $(x_j, y_j)_j$ sampled from the probability distribution ρ, but only a set $\{x_j\}$ sampled from ρ_X.

This raises the question if there exist other possible prior distributions X that may lead to a simple choice of a hypothesis class $\mathcal{H}$ containing the corresponding Bayes estimators, possibly endowed with an unsupervised technique to approximate them. If we consider a variational approach, in which $\mathcal{H}$ is a set of solution maps of suitable minimization problems, Gribonval (2011) showed that the conditional mean $R^\star$, i.e. the MMSE estimator, can always be represented as the minimizer of a functional Φ_{MMSE}, related to the probability distribution p_X. Nevertheless, in most cases, such a functional is not convex, which makes it critical to define $\mathcal{H}$ and to solve a minimization problem in it.

In this paper, without considering a variational point of view, we propose a strategy through which it is possible to associate a hypothesis class with (rather general) prior distributions of X. The most prominent outcome of this approach is the thorough treatment of the case in which X is a mixture of Gaussian random variables, which may be employed as a model of sparsity.

Our approach is based on the idea of parametric estimation in statistical inference: the distribution of the exact solution X is unknown, but belongs to a known class of distributions, parametrized by a suitable set of parameters:

$$\{\rho_X(\theta) : \theta \in \Theta\},$$

for some $\Theta \subseteq \mathbb{R}^p$. For example, we can assume that X is a Gaussian random variable of unknown mean μ and covariance Σ, the pair (μ, Σ) being the parameter θ. Analogously, we can describe more complicated probability distributions, possibly employing a larger set of parameters.

In this context it is natural to construct a hypothesis class $\mathcal{H}$ that contains the Bayes estimators $R^\star$ of every possible prior in the parametric class, namely,

$$\mathcal{H} = \{R_\theta = \mathbb{E}_{\rho(\theta)}[X|Y = \cdot] : \theta \in \Theta\},$$

where $\rho(\theta)$ is the joint probability distribution for (X, Y) when $X \sim \rho_X(\theta)$, $Y = AX + E$ and $E \sim \rho_E$. Recall that, by Assumption 2.1, the distribution of the noise E is known, and so it is considered fixed. However, the choice of this hypothesis class is nonstandard, if compared with deep learning approaches exploiting the approximation power of neural networks. The two main motivations for its adoption are that the Bayes estimator has sharp statistical properties in terms of MSE, and moreover it is well suited for both supervised and unsupervised learning approaches, as we explain below.

Notice that this same strategy can also be employed for nonlinear inverse problems: we formulated it in this narrower context for the ease of notation, since the content of the next sections is only related to linear problems.

Despite this approach being rather general, it is useful only if it is possible to provide a closed-form expression of R_θ, which might entail a learning approach for its approximation. This can be easily done when X belongs to fairly simple classes, such as Gaussian random variables or, as we show in

this section, mixtures of Gaussians. Other possible examples entailing closed-form representation of the posterior distribution (thus, of the Bayes estimator) are represented by the families of conjugate priors associated with the specified likelihood, i.e. with the noise model: for example, in the case of a Poisson likelihood, if the prior has a Gamma distribution then also the posterior does.

2.1 *Bayes estimator for linear inverse problems with a Gaussian Mixture prior*

We first recall some basic definitions on mixtures of random variables. Then, in Theorem 2.4 we show the formula of the Bayes estimator for linear inverse problems with a Gaussian mixture prior.

DEFINITION 2.2. A random variable X in $\mathbb{R}^n$ is a *mixture of random variables* if it can be written as

$$X = \sum_{i=1}^{L} X_i \mathbb{1}_{\{i\}}(I),$$

where X_i are random variables in $\mathbb{R}^n$, I is a random variable on $\{1, ..., L\}$ independent of X_i and $L \in \mathbb{N}^+$ is the number of elements in the mixture. The indicator function $\mathbb{1}_{\{i\}}(I)$ is equal to 1 when $I = i$ and 0 otherwise: as a consequence, the role of the discrete random variable I is selecting the random variable X_i, therefore $w_i := \mathbb{P}(I = i)$ are informally called the *weights of the mixture*.

DEFINITION 2.3. A random variable X in $\mathbb{R}^n$ is a *Gaussian mixture* if it is a mixture, as defined in Definition 2.2, and $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ are Gaussian random variables in $\mathbb{R}^n$.

THEOREM 2.4. Let X be a Gaussian mixture in $\mathbb{R}^n$, as in Definition 2.3, and let $E \sim \mathcal{N}(\mathbf{0}, \Sigma_E)$ be such that Assumption 2.1 is verified, i.e. Σ_E is invertible and E is independent of X_i for every i, and of I. Set $Y = AX + E$, as in (3). The corresponding Bayes estimator is

$$R^\star(y) = \mathbb{E}[X|Y = y] = \sum_{i=1}^{L} \frac{c_i}{\sum_{j=1}^{L} c_j} \left(\mu_i + \Sigma_i A^T (A\Sigma_i A^T + \Sigma_E)^{-1}(y - A\mu_i)\right), \tag{5}$$

where

$$c_i = \frac{w_i}{\sqrt{(2\pi)^n |A\Sigma_i A^T + \Sigma_E|}} \exp\Big(-\frac{1}{2}\|(A\Sigma_i A^T + \Sigma_E)^{-\frac{1}{2}}(y - A\mu_i)\|_2^2\Big), \tag{6}$$

with the notation $|B| = \det B$.

Note that the forward map A and the covariance of the noise Σ_E are considered known and fixed, and we assume that Σ_E is invertible, so that $A\Sigma_i A^T + \Sigma_E$ is invertible for every i. A proof of Theorem 2.4 can be found in Kundu *et al.* (2008). For completeness we provide a similar proof in Appendix A.

3. A neural network for sparse recovery

In this section we provide a novel interpretation both of the resulting formula (5) and of the Gaussian mixture model itself, in view of an application to sparsity-promoting learned regularization. We start by showing that the expression derived in (5) can be understood as a neural network, whose architecture has strong connections with the well-known attention mechanism used in transformers (Vaswani *et al.*, 2017).

Then, we describe how the Gaussian mixture model assumption can be used to encode a (group) sparsity prior on the unknown random variable X.

3.1 *Bayes estimator as a neural network*

We wish to interpret the expression of the Bayes estimator (5) for Gaussian mixture models as a neural network from $\mathbb{R}^m$ to $\mathbb{R}^n$.

We start by identifying the parameters: in particular, we define

$$\theta = \left(\{w_i\}_{i=1}^L, \{\mu_i\}_{i=1}^L, \{\Sigma_i\}_{i=1}^L\right), \tag{7}$$

collecting all the weights, means and covariances of the mixture. We can now denote the function defined in (5) as R_θ, parametrized according to (7). The resulting hypothesis class is

$$\mathcal{H} = \{R_\theta : \theta \in \Theta\}; \quad \Theta \subseteq [0,1]^L \times (\mathbb{R}^n)^L \times (\mathbb{R}^{n\times n})^L. \tag{8}$$

In order for R_θ to be well defined and for the parameters to have a precise statistical interpretation we impose that

$$\Theta = \left\{ \theta \text{ defined in (7)} : \sum_{i=1}^L w_i = 1 \text{ and } \Sigma_i \succcurlyeq 0 \text{ symmetric for } i = 1, ..., L \right\},$$

where $\Sigma_i \succcurlyeq 0$ denotes that Σ_i is positive semidefinite. Notice that the set $\mathcal{H}$ is the collection of the Bayes estimators of all possible Gaussian mixture models of L components in $\mathbb{R}^n$.

We now focus on describing formula (5) in terms of a neural network's architecture (see Fig. 1). We have the following equivalent formula for the Bayes estimator.

Proposition 3.1. We have that

$$R_\theta(y) = \sum_{i=1}^L W_i t_i, \tag{9}$$

where

$$W_i = \text{softmax}(z)_i := \frac{e^{z_i}}{\sum_{j=1}^L e^{z_j}},$$

with

$$z_i = f_i(y) := \log\left(\frac{w_i}{\sqrt{(2\pi)^n |A\Sigma_i A^T + \Sigma_E|}}\right) - \frac{1}{2}\|(A\Sigma_i A^T + \Sigma_E)^{-\frac{1}{2}}(y - A\mu_i)\|_2^2, \tag{10}$$

and

$$t_i = g_i(y) := \mu_i + \Sigma_i A^T (A\Sigma_i A^T + \Sigma_E)^{-1}(y - A\mu_i). \tag{11}$$

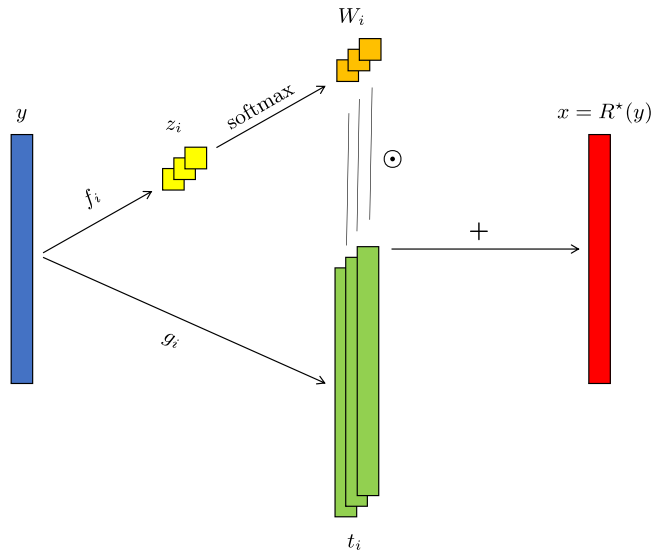

FIG. 1. Architecture of the neural network representing the Bayes estimator for denoising with $L = 3$.

The proof is a straightforward computation.

Note that c_i in (6) is $e^{f_i(y)}$ and that $g_i \colon \mathbb{R}^m \to \mathbb{R}^n$ is an affine map in y, whereas $f_i \colon \mathbb{R}^m \to \mathbb{R}$ is a *generalized quadratic function* (Mantini & Shah, 2021), namely,

$$f_i(y) = \gamma_i + b_i^T y + y^T A_i y,$$

for some $\gamma_i \in \mathbb{R}$, $b_i \in \mathbb{R}^m$ and $A_i \in \mathbb{R}^{m \times m}$. Second-order functions have already been used as neural network layers (Lee Giles & Maxwell, 1987).

Adopting the vocabulary of machine learning practitioners R_θ is a two-layer feed-forward neural network, with a single hidden layer that involves nonstandard operations on the input variables. Such a layer has some connections with an attention mechanism with L channels, as discussed in the next paragraph.

3.1.1 *Connections with the attention mechanism.* The transformer architecture, which uses the attention mechanism (Vaswani *et al.*, 2017), has greatly improved machine learning, especially in natural language processing (NLP). This mechanism helps models focus on the most important parts of the input, boosting performance in various tasks, including NLP and computer vision. Instead of treating all input elements the same the attention mechanism gives different weights to different parts, allowing models to allocate resources better and enhance both interpretability and performance.

More precisely, we consider the self-attention mechanism, which consists of three main components: queries, keys and values. The queries represent the elements that require attention, while the keys are compared with the queries to determine their relevance. The values correspond to the associated information or representations of the input elements. Attention scores are computed by measuring compatibility or similarity between the queries and the keys, often employing techniques like dot product, additive or multiplicative attention. Softmax normalization is then used to obtain attention weights.

Mathematically, an input $\eta \in \mathbb{R}^{L\times m}$ is linearly transformed into $Q = \eta U_Q$, $K = \eta U_K$ and $V = \eta U_V$, where U_Q, U_K *and* $U_V \in \mathbb{R}^{m\times m}$ are linear maps. The elements $Q, K, V \in \mathbb{R}^{L\times m}$ represent the L queries, keys and values of dimension m, respectively. The attention mechanism is then

$$\mathrm{Att}(\eta) = \mathrm{Att}(Q, K, V) := \mathrm{softmax}\left(\frac{QK^T}{\sqrt{L}}\right)V \in \mathbb{R}^{L\times m}, \tag{12}$$

where the softmax acts column-wise, namely,

$$\mathrm{softmax}(M)_{i,j} = \frac{e^{M_{i,j}}}{\sum_{l=1}^{N_c} e^{M_{i,l}}}, \quad M \in \mathbb{R}^{N_r\times N_c}.$$

In our case we have the following result, as an immediate consequence of Proposition 3.1.

Proposition 3.2. The Bayes estimator for a Gaussian mixture can be written as follows:

$$R^\star(y) = \overline{\mathrm{Att}}(\eta) = \overline{\mathrm{Att}}(Q, K, V) := \mathrm{softmax}\left((Q \odot K)\mathbf{1}_m^T\right)^T V \in \mathbb{R}^n, \tag{13}$$

where $\odot$ represents the Hadamard product (or element-wise product), and the matrices Q, K *and* $V \in \mathbb{R}^{L\times m}$ have rows defined, for $i = 1, ..., L$, by

- $Q_i = \frac{l_i^{\frac{1}{2}}}{\sqrt{m}}\mathbf{1}_m + \frac{1}{\sqrt{2}}M_i\eta_i \in \mathbb{R}^m$,
- $K_i = \frac{l_i^{\frac{1}{2}}}{\sqrt{m}}\mathbf{1}_m - \frac{1}{\sqrt{2}}M_i\eta_i \in \mathbb{R}^m$,
- $V_i = \mu_i + \Sigma_i A^T M_i^2 \eta_i \in \mathbb{R}^n$,

where

- $\eta_i = y - A\mu_i \in \mathbb{R}^m$,
- $l_i = \log\left(\frac{w_i}{\sqrt{(2\pi)^n |A\Sigma_i A^T + \Sigma_E|}}\right)$,
- $M_i = \left(A\Sigma_i A^T + \Sigma_E\right)^{-\frac{1}{2}}$,
- $\mathbf{1}_m = (1, \ldots, 1) \in \mathbb{R}^m$.

Therefore, we have found the queries, the keys and the values to view our network as an alternative version of the attention mechanism. The difference from the classical attention mechanism is the fact that the three transformations for finding Q, K *and* V are affine, and not linear, and that we use the element-wise product of matrices, instead of the matrix product, which leads to a different size of the network output.

3.2 *(Degenerate) Gaussian mixture as sparsity prior*

The Gaussian mixture prior can be viewed as a sparsity prior by taking X_i in Definition 2.3 as a degenerate Gaussian. This means that X_i is a Gaussian variable whose support is contained in an s-dimensional

subspace of $\mathbb{R}^n$, with $s \ll n$ denoting the sparsity level. In other words the covariance matrix Σ_i is degenerate, with $\dim(\ker \Sigma_i)^\perp \leq s$. Note that this setting, in which the covariances are not full rank, is compatible with the model considered so far, and with the corresponding estimator written as a neural network, because Σ_E is invertible.

Let us briefly discuss why this setting corresponds to a sparsity prior. Take $\mu_i = \mathbf{0}$. We write Σ_i with respect to its eigenvectors $\{\varphi_k^i\}_k$ and eigenvalues $\{\sigma_k^i\}_k$:

$$\Sigma_i = \sum_{k=1}^{s} \sigma_k^i \, \varphi_k^i \otimes \varphi_k^i.$$

This allows us to expand X_i as

$$X_i = \sum_{k=1}^{s} a_k^i \varphi_k^i,$$

where $a_k^i \sim \mathcal{N}(0, (\sigma_k^i)^2)$. Therefore, with probability w_i, we have $X = X_i$, and X_i is a random linear combination of $\varphi_1^i, \ldots, \varphi_s^i$, and so is an s-sparse vector.

In the case when the number of elements in the mixture, L, is equal to the number of all possible subsets of cardinality s of $\{1, \ldots, n\}$, $\binom{n}{s}$, and the eigenvectors φ_k^i are all chosen from a fixed orthonormal basis $\mathcal{B}$ of $\mathbb{R}^n$ the mixture generates all vectors that are s-sparse with respect to $\mathcal{B}$. However, $\binom{n}{s}$ grows very fast in s and n, and so this becomes unfeasible even with relatively small values of s and n. Therefore, we are led to take $L \ll \binom{n}{s}$, which corresponds to selecting *a priori* a subset of all possible s-dimensional subspaces of $\mathbb{R}^n$, a setting that is commonly referred to as group sparsity (Yuan & Lin, 2006). On the other hand, we have additional flexibility in the choice of the eigenvectors φ_k^i, which need not be chosen from a fixed basis.

While Gaussian distributions work well to represent smoothness priors, they are not well adapted to model sparsity. Here, we propose to use (degenerate) Gaussian mixture models to represent (group) sparsity prior. In the context of imaging inverse problems the use of Gaussian Mixture Models to describe structured sparsity has been leveraged in Guoshen *et al.* (2011), although focusing on maximum *a posteriori* rather than Bayes estimation. An alternative approach using hierarchical Bayesian models is considered in Calvetti *et al.* (2019, 2023). We also note that our unsupervised approach (introduced below) shares a similar clustering step with the dictionary learning strategy presented in Bocchinfuso *et al.* (2024) within a hierarchical model framework.

4. Proposed algorithms: supervised and unsupervised approaches

In this section we propose two different techniques to learn the neural network representation (9) of the Bayes estimator (5) to solve the statistical linear inverse problem of retrieving X from Y in (3). We assume that the random variables X and E satisfy Assumption 2.1, and write X as a mixture of L random variables

$$X = \sum_{i=1}^{L} X_i \mathbb{1}_{\{i\}}(I),$$

where the weights $w_{X_i} = \mathbb{P}(I = i)$, the means $\mu_{X_i} = \mathbb{E}[X_i]$ and the covariances $\Sigma_{X_i} = \mathrm{Cov}[X_i]$ are in general unknown.

We propose two possible regularizers of the form R_θ (cfr. (7) and (8)), for two ideal choices of parameters θ. The first one, which we will call *supervised*, is

$$\theta^* \in \operatorname{argmin}\left\{L(R_\theta) : \theta \in \Theta, \|\theta\|_\infty \leq \varrho\right\}, \tag{14}$$

for some $\varrho > 0$, being $L(R_\theta)$ defined as in (2) and $\|\theta\|_\infty$ the ℓ^∞ norm of the vectorized θ. Notice that, since the minimization problem (14) is restricted to a closed subset of a compact ball and $L(R_\theta)$ is continuous in θ, the minimum θ^* exists.

The second parameter choice, which corresponds to an *unsupervised* approach, is instead simply

$$\theta_X = (\{w_{X_i}\}_{i=1}^L, \{\mu_{X_i}\}_{i=1}^L, \{\Sigma_{X_i}\}_{i=1}^L), \tag{15}$$

where the involved quantities are defined above.

We immediately remark that both θ^* and θ_X depend on the probability distribution of X, which is in general unknown. In Section 4.1 we show how to approximate θ^* by taking advantage of a training set of the form $\{(x_j, y_j)\}_{j=1}^N$ sampled from (X, Y). This qualifies the strategy as a supervised learning algorithm. Instead, in Section 4.2, we show how to approximate θ_X by means of a training set of the form $\{x_j\}_{j=1}^N$, i.e. in an unsupervised way.

We also observe that, if we further assume that E is a Gaussian random variable and that X is a mixture of Gaussian random variables, then the outcomes of the two strategies coincide, provided that $\|\theta_X\|_\infty \leq \varrho$. Indeed, by Theorem 2.4, we know that R_{θ_X} is the Bayes estimator R^*, and so

$$R^* = R_{\theta_X} = R_{\theta^*}.$$

Nevertheless, the trained networks R_{θ^*} and R_{θ_X} could also be good estimators even in contexts that slightly deviate from the setting of Gaussian random variables. In particular, we are interested in testing them in cases in which X is group-sparse, but we are unsure if it is distributed as a Gaussian mixture model. If this is the case the two parameters are in general distinct—and also different from R^*, thus they are not theoretically guaranteed to be good estimators—and the two approximation strategies might achieve different levels of performance.

Notice also that, despite we suppose to deal with the sparsity prior provided by the degenerate Gaussian mixture model, the proposed strategies hold for a general mixture.

4.1 *Supervised approach*

Let $\{(x_j, y_j)\}_{j=1}^N$ be a training set, where $x_j \sim X$ are i.i.d., $\varepsilon_j \sim E$ are i.i.d. and $y_j = Ax_j + \varepsilon_j$. The parameter $\theta = (w_i, \mu_i, \Sigma_i)_{i=1}^L$ of the neural network defined in Section 3.1 can be learned by minimizing the empirical risk

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{j=1}^N \|x_j - R_\theta(y_j)\|_2^2, \tag{16}$$

which is the squared MSE between the original signals, x_j, and the reconstructions provided by the neural network, $R_\theta(y_j)$. In order to enforce sparsity it is possible to add a second term to the empirical risk, namely, to consider

$$\mathcal{L}_s(\theta) = \mathcal{L}(\theta) + \lambda \mathcal{J}(\theta),$$

where λ is a regularization parameter, $\mathcal{J}(\theta) = \sum_{i=1}^{L} \|\Sigma_i\|_*$, and $\|\cdot\|_*$ is the nuclear norm. Such a regularization term can be equivalently defined as the ℓ^1 norm of the singular values of Σ_i, hence it promotes low-rank covariances without requiring the knowledge of their eigenvectors.

The evaluation of the nuclear norm is computationally expensive and should be done L times for each step of the minimization process. Another option is taking $\mathcal{J}(\theta) = \sum_{i=1}^{L} \|\Sigma_i\|_{\mathrm{F}}^2$, where $\|\cdot\|_{\mathrm{F}}$ represents the Frobenius norm. This choice is computationally convenient, but does not promote any kind of sparsity on the covariances. However, in the numerical simulations, we do not observe significant differences when using the nuclear norm, the Frobenius norm or no regularization term (see Section 6.3.2). Therefore, in the numerical results of Section 6 we did not make use of any regularization term.

4.2 *Unsupervised approach*

We suppose to be in an unsupervised setting, i.e. to have a training set $\{x_j\}_{j=1}^N$, where x_j are sampled i.i.d. from the mixture $X = \sum_{i=1}^{L} X_i \mathbb{1}_{\{i\}}(I)$. We now wish to find an approximation for the means μ_{X_i}, the covariances Σ_{X_i} and the weights w_{X_i} of the mixture. We propose the following two-step procedure.

1. **Subspace clustering**. The elements of the training set $\{x_j\}_{j=1}^N$ sampled from different degenerate distributions X_i belong to different subspaces of $\mathbb{R}^n$. We propose to cluster these points by using subspace clustering and, in particular, the technique provided in Lu *et al.* (2012). As a result the training set is partitioned into $\widehat{L}$ subsets. Ideally, we should have $\widehat{L} = L$, and each subset should correspond to one Gaussian of the mixture.
2. **The parameters**. We compute the empirical means $\{\widehat{\mu}_i\}_{i=1}^{\widehat{L}}$ and the empirical covariances $\{\widehat{\Sigma}_i\}_{i=1}^{\widehat{L}}$ of each cluster, and we estimate the weights $\{\widehat{w}_i\}_{i=1}^{\widehat{L}}$ of the mixture by using the number of elements in the clusters.

Once this is done we can use the neural network defined in Section 3.1 with these parameters. More precisely, this is the neural network $R_{\widehat{\theta}_X}$, where

$$\widehat{\theta}_X = \left(\{\widehat{w}_i\}_{i=1}^{\widehat{L}}, \{\widehat{\mu}_i\}_{i=1}^{\widehat{L}}, \{\widehat{\Sigma}_i\}_{i=1}^{\widehat{L}}\right).$$

5. Baseline algorithms

In this section, we describe some of the most popular regularization techniques used to solve linear inverse problems with a sparsity prior. These include LASSO, Group LASSO, IHT and Dictionary Learning. We report here the formulation and general idea of such methods and postpone the discussion on the implementation aspects to Appendix B. In the numerical experiments of Section 6 we will compare our two proposed approaches against the methods presented in this section.

5.1 *LASSO*

LASSO (Tibshirani, 1996) is one of the most acknowledged techniques for sparsity promotion, also in the context of inverse problems. It involves the minimization of a functional composed by the sum of a data fidelity term and a regularization term that promotes sparsity with respect to a given basis. In particular, the regularization term is the ℓ^1 norm of the components of the unknown in the sparsifying basis.

More precisely, we suppose that the unknown x in (1) is sparse with respect to an orthonormal basis $\mathcal{B}$, and we denote by $M \in \mathbb{R}^{n\times n}$ the (orthogonal) matrix representing the change of basis from $\mathcal{B}$ to the canonical one, also known as the synthesis operator. Then, the functional to be minimized is

$$\mathcal{F}(\beta) = \frac{1}{2}\|y - AM\beta\|_2^2 + \lambda\|\beta\|_1, \tag{17}$$

where $\lambda > 0$ is a regularization parameter. The minimization of $\mathcal{F}(\beta)$ over $\mathbb{R}^n$, also known as synthesis formulation of LASSO, is discussed in Appendix B. The main drawbacks of this method are the choice of the regularization parameter λ, and the required prior knowledge of the synthesis operator M (i.e. of the basis $\mathcal{B}$ with respect to which the unknown is sparse). In Section 5.4, we show how to combine dictionary learning techniques to fill this gap. Other alternatives, explored in the numerical experiments in Section 6, are to leverage prior knowledge on M, or to infer it via the singular value decomposition (SVD).

5.2 *Group LASSO*

An extension of LASSO is Group LASSO (Friedman *et al.*, 2010), which encodes the idea that the features of the possible unknowns can be organized into groups. This is achieved by introducing L different coordinate systems, each represented by an orthogonal synthesis matrix $M_i \in \mathbb{R}^{n\times n}$, and by minimizing the following functional:

$$\mathcal{F}(\beta) = \frac{1}{2}\left\|y - A\sum_{i=1}^{L} M_i\beta_i\right\|_2^2 + \lambda\sum_{i=1}^{L}\|\beta_i\|_{K_i}, \tag{18}$$

where β_i is the coefficient vector corresponding to the ith group, $\beta \in \mathbb{R}^{nL}$ is the concatenation of the vectors $\beta_1, \ldots, \beta_L$ and $\lambda > 0$ is a regularization parameter. The second term of (18) is a regularization functional encouraging entire groups of features to be either included or excluded from the model. The weighted norm $\|\beta_i\|_2$ with $\|\beta_i\|_{K_i} := (\beta_i^T K_i \beta_i)^{\frac{1}{2}}$ is employed to promote prior information regarding the sparse representation of each group. The minimization of (18) can be performed via the iterative algorithm proposed in (Yuan & Lin, 2006, proposition 1), as discussed in Appendix B. As for LASSO, also for Group LASSO one has to choose the regularization parameter λ, and to know *a priori* the group bases $\mathcal{B}_i$ for $i = 1, ..., L$.

5.3 *IHT*

The IHT algorithm, as presented in Blumensath (2013), can be employed as a reconstruction method for inverse problems in which the unknown belongs to the union of L subspaces. This can be seen as a model of sparsity. Once an orthogonal synthesis matrix $M \in \mathbb{R}^{n\times n}$ is introduced the coefficients β such that $x = M\beta$ are supposed to satisfy $\beta \in \mathcal{S} = \cup_{i=1}^{L}\mathcal{S}_i$, where each $\mathcal{S}_i$ is a coordinate subspace in $\mathbb{R}^n$ of

dimension s or smaller, i.e. the span of a subset of cardinality at most s of the canonical basis. IHT then involves the minimization of

$$\mathcal{F}(\beta) = \frac{1}{2}\|y - AM\beta\|_2^2 + \chi_{\mathcal{S}}(\beta), \tag{19}$$

where

$$\chi_{\mathcal{S}}(\beta) = \begin{cases} 0 & \beta \in \mathcal{S}, \\ +\infty & \text{otherwise}, \end{cases}$$

forces the unknown to belong to $\mathcal{S}$.

As for LASSO and Group LASSO, the subspaces $\mathcal{S}_i$, and especially the basis given by M, should be known *a priori* or should be inferred. In this setting the sparsity level s can play the role of a regularization parameter (such as λ of LASSO and Group LASSO) and is tuned by suitable heuristic methods.

5.4 *Dictionary learning*

The baseline algorithms proposed in the previous sections need the *a priori* knowledge of the basis with respect to which the unknown is sparse. A possible approach to infer it is *sparse dictionary learning* (also known as sparse coding), where the dictionary is learned from the data (Julien *et al.*, 2009). Note that this dictionary does not necessarily have to be a basis. More precisely, given a training set $\{x_j\}_{j=1}^N$ with $x_j \in \mathbb{R}^n$, a sparsifying dictionary $D \in \mathbb{R}^{n\times d}$, where d is the number of elements of the dictionary (the columns of the matrix D), can be found as

$$\min_{D\in\mathbb{R}^{n\times d},\, \beta\in\mathbb{R}^{d\times N}} \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{2}\|x_j - D\beta_j\|_2^2 + \lambda\|\beta_j\|_1\right), \tag{20}$$

where $\lambda > 0$ is a regularization parameter and $\beta_j \in \mathbb{R}^d$ is the sparse representation of x_j with respect to the dictionary D. If the dictionary is fixed the functional in (20) resembles the LASSO functional (17), except for the fact that the dictionary may not be a basis and that we are summing over the elements of the training set. However, in this case, the goal is mainly to learn D. In order to solve the minimization problem in (20) we rely on the *online method* used in Julien *et al.* (2009), which iteratively minimizes over one variable keeping the other ones fixed. We update the sparse representation β using the least angle regression algorithm (LARS) and the dictionary D by block coordinate descent. As reported in Julien *et al.* (2009) the computational cost of each iteration of the online method is dominated by the update of β, and the complexity of LARS applied on $D \in \mathbb{R}^{n\times d}$ is of order $d^3 + d^2 n$ (see Efron *et al.* (2004)).

We assume that $D \in \mathbb{R}^{n\times d}$ is a full-rank matrix. In general, the number of atoms d is allowed to be larger than the original dimension of the signal n. Such a redundancy may arise with frames or with the union of multiple bases. Nevertheless, here we consider the nonredundant setting, with $d \leq n$. This is the case for all the numerical experiments reported in this work.

Once the dictionary is learned, the solution to the linear inverse problem (3) can be computed by combining the baseline algorithms of the previous sections. Under the assumptions previously introduced, the matrix D is injective, and the pseudo-inverse $D^+ = (D^T D)^{-1} D^T$ is its left inverse. We

thus tackle the minimization of the following functional:

$$\mathcal{F}(x) = \frac{1}{2}\|y - Ax\|^2 + \lambda G(x), \qquad G(x) = \chi_{\mathrm{Im}(D)}(x) + \|D^+ x\|_1, \tag{21}$$

where $\chi_{\mathrm{Im}(D)}(x)$ is the characteristic function of the range $\mathrm{Im}(D)$.

Since the degenerate Gaussian mixture prior represents a group sparsity prior it is useful to compare our algorithms with a 'group' version of sparse dictionary learning. This has been explored with the name 'Block-sparse' or 'Group-sparse' dictionary learning (Li *et al.*, 2012; Zelnik-Manor *et al.*, 2012).

Those techniques, however, do not share the same perspective on group sparsity as the one inspiring our proposed algorithms. The common feature behind all of them is the assumption that, when the signals are represented in a suitable basis or dictionary, there exists groups, or blocks, of coordinates that are simultaneously activated. In our degenerate Gaussian mixture approach, though, the signals can be clustered according to which group of coordinates they activate: each signal is thus associated with a single group. In block-sparse dictionary learning (and, similarly, in the Group LASSO approach previously exposed) the active components of each signal can also belong to (a small number of) different groups.

For this reason we propose an alternative dictionary learning technique that is closer to our set-up and can be employed for more direct comparisons. We simply call it 'Group Dictionary Learning': after doing a subspace clustering of the training set as explained in Section 4.2 it is possible to learn a dictionary D_i for each group obtained with the clustering. Assuming that each $D_i \in \mathbb{R}^{n\times d_i}$ is injective for $i = 1, ..., L$, and adopting the same approach as in (21), we tackle the minimization of the following functional:

$$\begin{aligned} \mathcal{F}(x) &= \frac{1}{2}\|y - Ax\|^2 + \lambda \hat{G}(x), \\ \hat{G}(x) &= \min_{i=1,\dots,L} G_i(x), \qquad G_i(x) = \chi_{\mathrm{Im}(D_i)}(x) + \|D_i^+ x\|_1. \end{aligned} \tag{22}$$

Remark 5.1. Differently from dictionary learning the approaches proposed in Section 4 do not focus on the reconstruction of a dictionary D. However, such a dictionary might be obtained as a by-product, e.g. by computing the singular vectors of each covariance Σ_i and by collecting them in a single matrix, but the theoretical properties of such an object are out of the scope of this work. Nevertheless, we wish to underline that, as in the problem of dictionary learning, our algorithms do not require the knowledge of the dictionary D as an input. It is worth noting that the parameters $\theta = (w_i, \mu_i, \Sigma_i)_{i=1}^L$ of our network are $O(n^2 \times L)$, while the dictionary has only $n \times d$ parameters, where d is usually chosen as $O(n)$. Therefore, one of the reasons our methods seem more effective may be that they employ more parameters, a well-known (but possibly not fully understood) phenomenon related to overparametrization in deep learning.

6. Numerical results

In this section we compare the methods proposed in Section 4 with the baseline algorithms discussed in Section 5 for one-dimensional denoising and deblurring problems with three datasets. Our experiments primarily aim to compare our methods with classical sparsity-promoting techniques on simplified signal classes, rather than striving for state-of-the-art results. The exploration of more complex inverse problems and the incorporation of real-world data fall outside the scope of this paper and are designated for future research.

6.1 *Datasets*

We consider three datasets with increasing complexity.

1. *Gaussian mixture model*: this dataset contains samples from a degenerate Gaussian mixture variable (2.3) where I is a uniform variable (all the Gaussians of the mixture have the weight $w_i = \frac{1}{L}$). Each X_i is a Gaussian random variable in $\mathbb{R}^n$, having mean $\mu_i = \mathbf{0}$ and whose covariance matrix Σ_i has zero entries everywhere, except from s elements on the diagonal, which are set to 1. This indicates that we are considering a mixture of degenerate variables, each of which has support on an s-dimensional coordinate hyperplane of $\mathbb{R}^n$ and whose components are independent standard Gaussian random variables. In our experiments we consider $n = 1000$, $s = \operatorname{rank} \Sigma_i = 20$ and $L = 10$. Therefore, this dataset consists of very sparse random variables in a rather large ambient space. The sparsity is nevertheless very structured, since only $L = 10$ combinations of s active coefficients are considered.

2. *Sinusoidal functions with one discontinuity*: this dataset contains functions with support in $[0, 4\pi]$ of the type

$$x(\tau) = \begin{cases} A \sin(\omega\tau) + B & 0 \leq \tau \leq \tau_i \\ A \sin(\omega\tau) + B + C & \tau_i < \tau \leq 4\pi, \end{cases}$$

where the amplitude $A \sim \mathcal{U}(0.05, 0.1)$, the angular frequency $\omega \sim \mathcal{U}(1, 2)$ and the vertical translations $B \sim \mathcal{U}(\frac{1}{2}, 3)$, $C \sim \mathcal{N}(0, 0.2^2)$. The discontinuity points τ_i with $i = 1, ..., L$ and $L = 10$ correspond to different subspaces and are equispaced points in $[0, 4\pi]$. Each signal x is discretized on 1000 equispaced points in $[0, 4\pi]$. These functions can be approximately seen as samples from a degenerate Gaussian mixture in a wavelet basis. While the coarse scale coefficients are expected to be nonzero for all the signals, the wavelet coefficients at the finer scales will be relevant only in the locations of the discontinuity, and negligible in the smooth regions. Therefore, each discontinuity determines a coordinate subspace with respect to the wavelet basis. Considering that the amplitude of the jump C follows a Gaussian distribution we posit that the fine-scale coefficients can be effectively modelled as degenerate Gaussian random vectors. We can also infer an estimate of the sparsity level in terms of the size of the support of the mother wavelet, and of the considered resolution scales.

3. *Truncated Fourier series with two discontinuities*: this dataset contains functions with support in $[0, 4\pi]$ of the form

$$x(\tau) = \begin{cases} \sum_{d=1}^{4} a_d \cos(2\pi d\tau) + b_d \sin(2\pi d\tau) & 0 \leq \tau \leq \tau_i(1) \\ \sum_{d=1}^{4} a_d \cos(2\pi d\tau) + b_d \sin(2\pi d\tau) + C_1 & \tau_i(1) < \tau \leq \tau_i(2) \\ \sum_{d=1}^{4} a_d \cos(2\pi d\tau) + b_d \sin(2\pi d\tau) + C_2 & \tau_i(2) < \tau \leq 4\pi. \end{cases}$$

where the Fourier coefficients $a_d, b_d \sim \mathcal{N}(0.1, 0.1^2)$ for $d = 1, ..., 4$ and the vertical translations $C_1, C_2 \sim \mathcal{N}(0, 0.2^2)$. As for dataset 2 the discontinuity points are 10 equispaced points in $[0, 4\pi]$ and τ is discretized with 1000 equispaced points in $[0, 4\pi]$. However, each function has either one or two discontinuities. More precisely, if $\tau_i(1) = \tau_i(2)$ the function has one discontinuity, otherwise it has two discontinuities. Therefore, the subspaces are the ones representing functions with two discontinuities (all the possible combinations of 10 elements in groups of 2) and the ones

representing functions with 1 discontinuity (10 subspaces), so the total number of subspaces is $L = \binom{10}{2} + 10 = 55$. For the same reasons discussed above for dataset 2 these functions can be seen as approximate samples from a degenerate Gaussian mixture distribution in the wavelet domain. The level of sparsity can be estimated as in the case of the previous dataset, updating it to the presence of two distinct singularities.

6.2 *Methods*

In the following experiments we test our supervised and unsupervised approaches described in Sections 4.1 and 4.2 and compare them with the baseline algorithms described in Section 5. More precisely, we consider the following methods.

(A) *Supervised* (Section 4.1). We consider both the cases: one where the weights of the network are randomly initialized, denoted as (A), and the one in which they are initialized by the values obtained as an outcome of the unsupervised procedure, denoted as (A+B).

(B) *Unsupervised* (Section 4.2).

(C) *Dictionary learning* (Section 5.4). We learn a sparsifying dictionary D for the training set, using the Lasso LARS algorithm to solve (20); then, we employ the learned D to denoise the test signal by minimizing (21). We choose the number of elements of the dictionary as $d = \frac{n}{2}$, where n is the signal length, in order to balance expressivity and numerical efficiency.

(D) *Group dictionary learning* (Section 5.4). After dividing the training set into L clusters and learning a sparsifying dictionary D_i for each group, we minimize (22). In this case, we choose $d = \frac{n}{2L}$.

(E) *IHT with SVD basis* (Section 5.3). We infer the basis with respect to which the unknown is sparse by computing the SVD of the empirical covariance matrix of the training set and by considering the orthonormal basis composed by the eigenvectors. Then, we choose $\mathcal{S}$ as the union of all the s-dimensional coordinate subspaces in $\mathbb{R}^n$ w.r.t. the inferred sparsity basis. Here, we choose the degree of sparsity s by minimizing the relative MSE (computed w.r.t. the norm of the original signals) over the training set.

(F) *IHT with SVD bases of groups* (Section 5.3). We divide the training set into L clusters as explained in Section 4.2. Then, for each group provided by the clustering, we infer the basis with respect to which that group is sparse by computing the SVD of its empirical covariance matrix and by considering the eigenvectors corresponding to the s largest eigenvalues, where the degree of sparsity s is chosen as for (E). Finally, we choose $\mathcal{S}$ as the union of the s-dimensional subspaces spanned by the bases inferred.

(G) *IHT with known basis* (Section 5.3). We suppose to know the basis with respect to which the unknown is sparse. In particular, for Dataset 1, we consider the canonical basis, while Datasets 2 and 3 require a more complicated treatment. Indeed, as discussed in Section 6.1, their representation in a suitable wavelet basis is sparse at fine scales. For this reason we preprocess the datasets by applying the wavelet transform, keep the low-scale coefficients fixed and apply IHT only to the fine scales. We choose the wavelet basis generated by the Daubechies wavelet with six vanishing moments.

(H) *LASSO with SVD basis* (Section 5.1). We infer the basis with respect to which the unknown is sparse as in (E). Then, we minimize (17).

(I) *Group LASSO with SVD bases* (Section 5.2). We divide the training set into L clusters as explained in Section 4.2. Then, for each group provided by the clustering, we infer the basis with respect to which that group is sparse as explained in (F). Here, however, we consider the complete bases provided by the SVDs of the empirical covariance matrices of the groups. We choose the power $-\frac{1}{2}$ of the empirical covariance matrices of each group as penalty matrices K_i.

(J) *LASSO with known basis* (Section 5.1). We incorporate in LASSO the knowledge of the basis with respect to which the unknown is sparse, as already explained for (G). This implies that, for Dataset 1, we solve LASSO with the canonical basis, whereas for Datasets 2 and 3 we first compute the wavelet transform and then solve LASSO only on the high-scale coefficients.

Whenever needed (methods (C), (D), (H), (I) and (J)) we always choose the optimal regularization parameter λ, namely, the one minimizing the relative MSE over the training set. Further, whenever needed (methods (A), (B), (D), (E), and (I)) we always suppose to know the number of Gaussians L in the mixture.

Methods (B), (D), (E) and (I) rely on clustering. The subspace clustering procedure described in Section 4.2 is applied to the signals for Dataset 1 and to their derivatives computed as finite-difference approximations for Datasets 2 and 3. Indeed, for the latest datasets, the derivative of the signals highlights the discontinuity points and allows for a more efficient clustering.

It is worth noting that methods (G) and (J) take advantage of the knowledge of the basis with respect to which the unknown is sparse. For this reason they do not constitute a fair comparison for all the other algorithms, which do not use this piece of information. However, we are interested to see if they can outperform our methods.

6.3 *Denoising*

In this section we focus on the denoising problem with 10% of noise, namely, $y = x + \varepsilon$, where ε is sampled from $\mathcal{N}(\mathbf{0}, \sigma^2 I)$ and σ is the 10% of the maximum of the amplitudes of the training set signals, i.e. $\max_f(\max f - \min f)$, where f is a training sample and the maximum is taken for each dataset. We compare the performances of the methods reported in Section 6.2.

In Table 1 we show the mean over 2000 signals of the test set of the relative MSE between the original signals and the different reconstructions. We observe that our unsupervised technique (B) provides the best reconstructions for Datasets 1 and 3, and for Dataset 2 it is only outperformed by group dictionary learning (D). However, dictionary learning is computationally more expensive than our unsupervised method. Indeed, although both techniques involve a preliminary clustering step, group dictionary learning also requires computing L sparsifying dictionaries, i.e. solving L optimization problems as (20). In our experiments, for each dictionary, the online method presented in Section 5.4 performed nearly 300 iterations, with the cost of a single update depending on the size of both the dictionary and the sample, as discussed in Section 5.4. On the other hand, the unsupervised technique only requires estimating L means and covariances prior to the application of (9), and the difference in performance is minimal (as it can be seen in Fig. 2). From Table 1 it seems clear that the best baseline algorithms to which we can compare our approaches are dictionary learning (C), among the methods that do not use clustering, and group dictionary learning (D), among the methods using the groups obtained by the clustering of the training set. For this reason, in Fig. 2, we show a qualitative comparison between the reconstructions provided by our methods, dictionary learning and group dictionary learning. For completeness we provide the remaining qualitative comparisons in Fig. C4 of Appendix C. We notice that for Datasets 2 and 3 the algorithms learn all the discontinuities and try to remove those not present in the signal.

TABLE 1 *Relative MSE values for the denoising problem with* 10% *noise*

	Dataset 1	Dataset 2	Dataset 3
Supervised (A)	1.97%	7.03×10^{-3}%	2.71×10^{-2}%
Supervised (A + B)	**0.98**%	1.79×10^{-3}%	6.38×10^{-3}%
Unsupervised (B)	**0.98**%	1.78×10^{-3}%	$\mathbf{6.32 \times 10^{-3}}$%
Dictionary learning (C)	4.18%	3.43×10^{-3}%	7.13×10^{-3}%
Group dictionary learning (D)	0.99%	$\mathbf{1.70 \times 10^{-3}}$%	2.42×10^{-2}%
IHT with SVD basis (E)	9.93%	1.25×10^{-2}%	3.97×10^{-1}%
IHT with SVD bases of groups (F)	0.99%	2.04×10^{-3}%	3.97×10^{-1}%
IHT with known basis (G)	2.78%	1.22×10^{-2}%	2.46×10^{-2}%
LASSO with SVD basis (H)	9.24%	1.55×10^{-2}%	3.07×10^{-1}%
Group LASSO with SVD bases (I)	3.01%	3.71×10^{-3}%	8.31×10^{-1}%
LASSO with known basis (J)	4.69%	1.00×10^{-2}%	2.15×10^{-2}%

TABLE 2 *Relative MSE values for the denoising problem with* 10% *noise with the unsupervised approach (B) when using exact vs. random clustering*

	Dataset 1	Dataset 2	Dataset 3
Exact	0.97%	1.66×10^{-3}%	3.37×10^{-3}%
Learned	0.98%	1.80×10^{-3}%	6.32×10^{-3}%
Random	8.25 ± 0.04%	$(3.46 \pm 0.25) \times 10^{-3}$%	$(5.70 \pm 0.55) \times 10^{-3}$%

6.3.1 *Does clustering really matter?.* Considering that accurately clustering the training set is challenging (see e.g. Dataset 3), we question whether precise clustering is truly necessary for employing the unsupervised algorithm proposed in Section 4.2. For this purpose, in Table 2, we show the mean over 2000 signals of the test set of the relative MSE between the original signals and the reconstructions provided by the unsupervised approach with exact clustering and with random clustering. We observe that correct clustering is important for Dataset 1, but not so important for Datasets 2 and 3. We believe that this is due to the fact that in Datasets 2 and 3 the energy of the signals is shared between the smooth part, which is independent of the clustering, and the discontinuities. In Dataset 1 the absence of the (nonzero) smooth part makes the dependence on the clustering stronger.

6.3.2 *Does regularization really matter?.* We investigated the impact of the regularization term $\mathcal{J}$ added to the training loss in the supervised approach, as discussed in Section 4.1. Table 3 shows that, when applied to Dataset 1, the use of the Frobenius or the nuclear norms does not improve the results in terms of MSE. In both cases the value of the regularization parameter is heuristically chosen. Notice that also the matrices Σ_i trained without any penalty term naturally show a good degree of sparsity (quantified by the ratio between their average rank and the size of the signal).

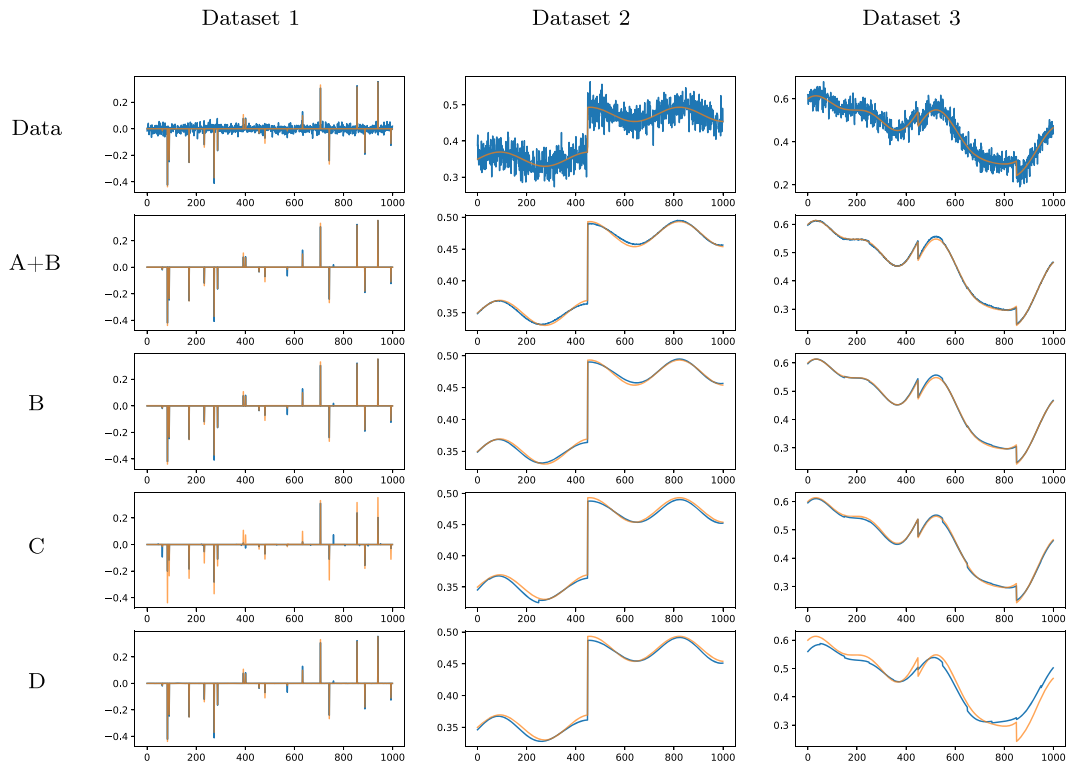

FIG. 2. Qualitative comparison for the denoising problem. In each column we show a signal of the test set from Datasets 1, 2 and 3, respectively. In each row we report the original data in foreground, and the noisy data, the reconstructions obtained with the methods supervised (initialized with the unsupervised one) (A+B), unsupervised (B), dictionary learning (C) and group dictionary learning (D), in background.

TABLE 3 *Denoising problem with* 10% *noise for Dataset* 1 *with the supervised approach (B) without penalty, with a nuclear-norm penalty, and with a Frobenius-norm penalty. We compare the values of the relative MSE and the ratio between the average rank of the learned* Σ_i *and the signal size* n

	No regularization	Nuclear norm	Frobenius norm
Relative MSE	1.97%	2.18%	2.11%
Average rank of Σ_i	0.41 n	0.22 n	0.92 n

6.4 *Deblurring*

In this section we focus on a deblurring problem with 10% of noise, namely, we consider the problem $y = q * x + \varepsilon$, where $*$ represents the discrete convolution, the noise ε is sampled from $\mathcal{N}(\mathbf{0}, \sigma^2 I)$ and σ is the 10% of the maximum of the amplitudes of the training set signals. We choose the filter q to be Gaussian, i.e. the entries of q are a finite discretization of the density of $\mathcal{N}(0, \sigma_b^2)$ in a neighbourhood of 0. For Dataset 1 we choose $\sigma_b = 1$, while for Datasets 2 and 3 we set $\sigma_b = 30$ and $\sigma_b = 20$,

TABLE 4 *Relative MSE values for the deblurring problem with* 10% *noise*

	Dataset 1	Dataset 2	Dataset 3
Unsupervised (B)	**3.68%**	$\mathbf{2.65 \times 10^{-3}}$%	$\mathbf{1.01 \times 10^{-2}}$%
Dictionary learning (C)	14.32%	6.61×10^{-3}%	1.28×10^{-2}%
Group dictionary learning (D)	13.51%	4.62×10^{-3}%	3.41×10^{-2}%
IHT with SVD bases of groups (F)	3.80%	5.54×10^{-3}%	9.48×10^{-1}%
Group LASSO with SVD bases (I)	11.48%	1.34×10^{-2}%	9.11×10^{-1}%

respectively. The latter values of σ_b are larger because the effect of the convolution on piecewise smooth signals (Datasets 2 and 3) is less visible than on Dirac deltas (Dataset 1). Since for the denoising problem our unsupervised method provides better results than the supervised one we only show the results for the deblurring problem using the unsupervised technique and we compare it with the most significant baseline algorithms. In particular, we consider the following methods: Unsupervised (B), Dictionary learning (C), Group dictionary learning (D), IHT with SVD bases of groups (F) and Group LASSO with SVD bases (I). We exclude from the comparisons the methods that do not employ clustering, except for dictionary learning, as they performed significantly worse in the denoising experiments.

In Table 4 we show the mean over 2000 signals of the test set of the relative MSE between the original signals and the different reconstructions. As for the denoising problem our unsupervised method outperforms the others.

In Fig. 3 we show a qualitative comparison between the reconstructions provided by our methods, dictionary learning and group dictionary learning. For completeness we provide the remaining qualitative comparisons in Appendix C.

7. Conclusions

We introduced an innovative approach to sparse optimization for inverse problems, leveraging an explicit formula for the MMSE estimator in the context of a mixture of degenerate Gaussian random variables. This methodology, rooted in statistical learning theory, follows a different approach to sparsity promotion compared with the deterministic optimization paradigm (e.g. Lasso) and the Bayesian inverse problem framework. Our reconstruction formula exhibits a notable connection to the self-attention mechanism underlying the transformer architecture, offering an efficient training process: this is useful whenever the mixture model, corresponding to the sparsity properties of the signals of interest, is unknown. This training can be conducted in both supervised and unsupervised modes.

To validate our approach we conducted numerical implementations (both supervised and unsupervised) and compared them with established baseline algorithms for sparse optimization, such as Lasso, IHT and their group variants. Additionally, we incorporated a dictionary learning strategy for fair comparisons. The experiments focused on three datasets of one-dimensional signals (discretized as vectors in $\mathbb{R}^n$) featuring sparse or compressible signals, addressing denoising and deblurring tasks.

Our findings indicate that the unsupervised method consistently outperforms baseline algorithms in nearly all experiments, demonstrating superior performance with lower computational costs. However, a notable limitation of our work lies in its numerical scalability, particularly concerning larger and higher dimensional datasets. Addressing this limitation will be a key focus in future research endeavours.

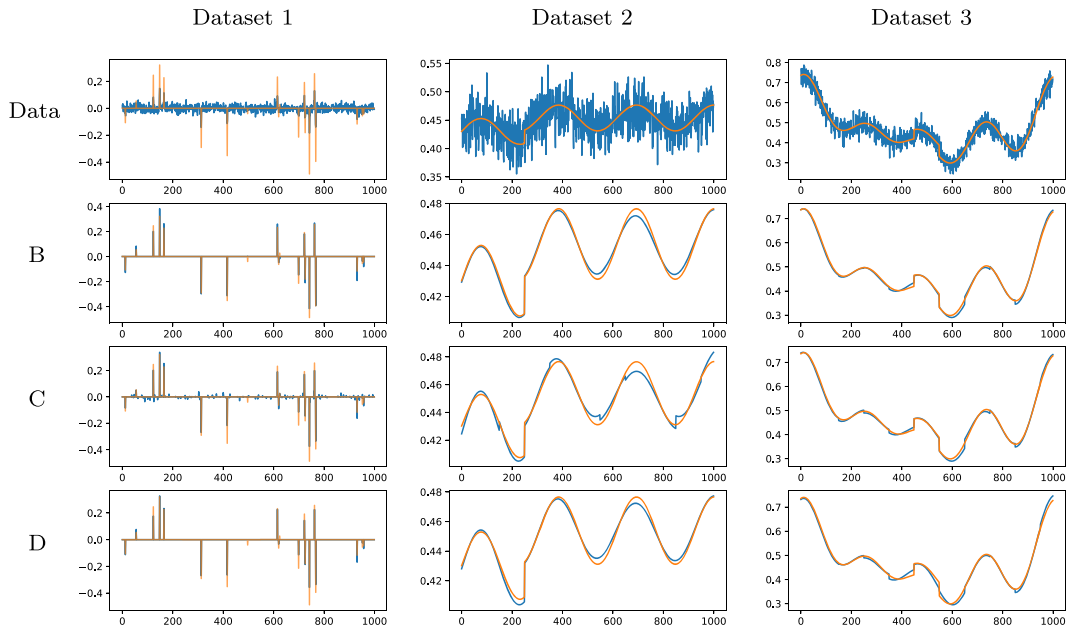

FIG. 3. Qualitative comparison for the deblurring problem. In each column we show a signal of the test set from Datasets 1, 2 and 3, respectively. In each row we report the original data in foreground and the noisy data, and the reconstructions obtained with the unsupervised (B), the dictionary learning (C) and the group dictionary learning (D) approach, in background.

Acknowledgements

It is a pleasure to thank Ernesto De Vito and Matti Lassas for stimulating discussions on some of the aspects of this work.

Funding

This material is based upon work supported by the Air Force Office of Scientific Research under award numbers FA8655-20-1-7027 and FA8655-23-1-7083; Fondazione Compagnia di San Paolo; European Union (ERC, SAMPDE, 101041040); PNRR-M4C2 - Investimento 1.3. Partenariato Esteso PE00000013 - 'FAIR-Future Artificial Intelligence Research' - Spoke 8 'Pervasive AI' (to L.R.); Ministry of Education, Universities and Research (CUP D33C23001110001).

REFERENCES

ALBERTI, G. S., DE VITO, LASSAS, M., RATTI, L. & SANTACESARIA, M. (2021) Learning the optimal Tikhonov regularizer for inverse problems. *Advances in Neural Information Processing Systems*, **34**, 25205–25216.

BECK, A. (2017) *First-Order Methods in Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

BLUMENSATH, T. (2013) Compressed sensing with nonlinear observations and related nonlinear optimization problems. *IEEE Trans. Inf. Theory*, **59**, 3466–3474.

BOCCHINFUSO, A., CALVETTI, D. & SOMERSALO, E. (2024) Bayesian sparsity and class sparsity priors for dictionary learning and coding. *J. Comput. Math. Data Sc.,* **11**, 100094.

CALVETTI, D., SOMERSALO, E. & STRANG, A. (2019) Hierachical Bayesian models and sparsity: ℓ_2-magic. *Inverse Probl.*, **35**, 035003.

Calvetti, D., Somersalo, E., Calvetti, D. & Somersalo, E. (2023) *Hierarchical Models and Bayesian Sparsity*. Cham: Springer International Publishing, pp. 183–210.

Candès, E. J., Romberg, J. & Tao, T. (2006) Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory*, **52**, 489–509.

Cucker, F. & Smale, S. (2002) On the mathematical foundations of learning. *Bull. Am. Math. Soc.*, **39**, 1–49.

Daubechies, I., Defrise, M. & De Mol (2004) An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. Pure Appl. Math.*, **57**, 1413–1457.

de Hoop, Kovachki, N. B., Nelsen, N. H. & Stuart, A. M. (2023) Convergence rates for learning linear operators from noisy data. *SIAM/ASA J. Uncertainty Quantif.*, **11**, 480–513.

Donoho, D. L. (2006) Compressed sensing. *IEEE Trans. Inf. Theory*, **52**, 1289–1306.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004) Least angle regression. *Ann. Stat.*, **32**, 407–451.

Friedman, J., Hastie, T. & Tibshirani, R. (2010) A note on the group lasso and a sparse group lasso. arXiv preprint arXiv:1001.0736.

Gribonval, R. (2011) Should penalized least squares regression be interpreted as maximum a posteriori estimation? *IEEE Trans. Signal Process.*, **59**, 2405–2410.

Guoshen, Y., Sapiro, G. & Mallat, S. (2011) Solving inverse problems with piecewise linear estimators: from Gaussian mixture models to structured sparsity. *IEEE Trans. Image Process.*, **21**, 2481–2499.

Hale, E. T., Yin, W. & Zhang, Y. (2008) Fixed-point continuation for ℓ_1-minimization: Methodology and convergence. *SIAM J. Optim.*, **19**, 1107–1130.

Horesh, L. & Haber, E. (2009) Sensitivity computation of the l_1 minimization problem and its application to dictionary design of ill-posed problems. *Inverse Probl.*, **25**, 095009.

Huang, H., Haber, E. & Horesh, L. (2012) Optimal estimation of ℓ_1-regularization prior from a regularized empirical Bayesian risk standpoint. *Inverse Probl. Imaging*, **6**, 447–464.

Kovachki, N. B., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A. M. & Anandkumar, A. (2023) Neural operator: learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, **24**, 1–97.

Kundu, A., Saikat Chatterjee, A., Murthy, S. & Sreenivas, T. V. (2008) GMM based Bayesian approach to speech enhancement in signal/transform domain. *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, Las Vegas, NV, USA, pp. 4893–4896.

Lee, H., Battle, A., Raina, R. & Ng, A. (2006) Efficient sparse coding algorithms. *Advances in neural information processing systems*, **19**, 801–808.

Lee Giles, C. & Maxwell, T. (1987) Learning, invariance, and generalization in high-order neural networks. *Appl. Opt.*, **26**, 4972–4978.

Li, S., Yin, H. & Fang, L. (2012) Group-sparse representation with dictionary learning for medical image denoising and fusion. *IEEE Trans. Biomed. Eng.*, **59**, 3450–3459.

Lu, C.-Y., Min, H., Zhao, Z.-Q., Zhu, L., Huang, D.-S. & Yan, S. (2012) Robust and efficient subspace segmentation via least squares regression. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision,* Florence, Italy, October 7–13, 2012, Proceedings, Part VII 12, pages 347–360. Springer, Berlin, Heidelberg.

Lunz, S., Öktem, O. & Schönlieb, C.-B. (2018) Adversarial regularizers in inverse problems. *Advances in neural information processing systems*, **31**, 8507–8516.

Lustig, M., Donoho, D. L., Santos, J. M. & Pauly, J. M. (2008) Compressed sensing MRI. *IEEE Signal Process. Mag.*, **25**, 72–82.

Mairal, J., Bach, F., Ponce, J. & Sapiro, G. (2009) Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*. Montreal, Quebec, Canada, pp. 689–696.

Mallat, S. (2008) *A Wavelet Tour of Signal Processing: The Sparse Way*. Academic Press, Elsevier, Burlington, MA.

Mantini, P. & Shah, S. K. (2021) Cqnn: Convolutional quadratic neural networks. *In 2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Milan, Italy, pp. 9819–9826.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011) Scikit-learn: machine learning in python. *J. Mach. Learn. Res.*, **12**, 2825–2830.

Schwab, J., Antholzer, S. & Haltmeier, M. (2019) Deep null space learning for inverse problems: convergence analysis and rates. *Inverse Probl.*, **35**, 025008.
Shiryaev, A. N. (2016) *Probability-1*, vol. **95**. New York, NY: Springer.
Tibshirani, R. (1996) Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. B*, **58**, 267–288.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017) Attention is all you need. *Advances in neural information processing systems*, **30**, 5998–6008.
Yuan, M. & Lin, Y. (2006) Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. B*, **68**, 49–67.
Zelnik-Manor, L., Rosenblum, K. & Eldar, Y. C. (2012) Dictionary optimization for block-sparse representations. *IEEE Trans. Signal Process.*, **60**, 2386–2395.

Appendix A. Proof of Theorem 2.4

The proof of Theorem 2.4 follows easily using the next two lemmas. The first lemma provides the formula of the Bayes estimator for the denoising problem with a random variable that comes from a general mixture distribution. The second one establishes an explicit formula in the case of Gaussian mixture distributions.

Lemma A1. Suppose that Assumption 2.1 holds true. Consider the statistical linear inverse problem (3), where x is sampled from a mixture of random variables, as in Definition 2.2, and $A \in \mathbb{R}^{m\times n}$. Let $Y_i = AX_i + E$, for $i = 1, \dots, L$. Then,

1. the density of Y is $p_Y(y) = \sum_{i=1}^{L} w_i p_{Y_i}(y)$, where $w_i = \mathbb{P}(I = i)$;
2. $\mathbb{E}[X|Y = y] = \sum_{i=1}^{L} \frac{w_i p_{Y_i}(y)}{p_Y(y)} \mathbb{E}[X_i|Y_i = y]$.

Proof. We prove the two parts separately.

Proof of 1. The random variable Y can be written as

$$Y = A\left(\sum_{i=1}^{L} X_i \mathbb{1}_{\{i\}}(I)\right) + E = \left(\sum_{i=1}^{L} AX_i \mathbb{1}_{\{i\}}(I)\right) + \left(\sum_{i=1}^{L} E\mathbb{1}_{\{i\}}(I)\right) = \sum_{i=1}^{L} Y_i \mathbb{1}_{\{i\}}(I).$$

Since $Y_i = AX_i + E$ and $X_i, E \perp I$ for every $i = 1, ..., L$ $Y_i \perp I$ for every $i = 1, ..., L$. Therefore, the density of Y becomes

$$p_Y(y) = \sum_{i=1}^{L} w_i p_{Y_i}(y),$$

recalling that $w_i = \mathbb{P}(I = i)$.

Proof of 2. Since X_i and E are independent their joint density is $p_{X_i,E}(x, \varepsilon) = p_{X_i}(x)p_E(\varepsilon)$. Then, using the change of variable $\Phi(X_i, E) = (X_i, AX_i + E) = (X_i, Y_i)$ we have

$$\begin{aligned} p_{X_i,Y_i}(x, y) &= p_{\Phi(X_i,E)}(\Phi(x, \varepsilon)) \\ &= p_{X_i,E}(x, \varepsilon)|J_{\Phi^{-1}}| \\ &= p_{X_i,E}(x, \varepsilon) \\ &= p_{X_i}(x)p_E(y - Ax), \end{aligned}$$

since $\Phi^{-1}(X_i, Y_i) = (X_i, Y_i - AX_i)$ and

$$J_{\Phi^{-1}} = \begin{bmatrix} I & \mathbf{0} \\ -A & I \end{bmatrix}.$$

The same argument holds using X instead of X_i:

$$p_{X,Y}(x, y) = p_X(x)p_E(y - Ax).$$

Moreover, since $X_i \perp I$ for every $i = 1, ..., L$ $p_X(x) = \sum_{i=1}^{L} w_i p_{X_i}(x)$ with $w_i = \mathbb{P}(I = i)$. Then,

$$\begin{aligned} p_{X|Y}(x|y) &= \frac{p_{X,Y}(x, y)}{p_Y(y)} \\ &= \sum_{i=1}^{L} \frac{w_i p_{X_i}(x) p_E(y - Ax)}{p_Y(y)} \\ &= \sum_{i=1}^{L} \frac{w_i p_{Y_i}(y)}{p_Y(y)} \frac{p_{X_i}(x) p_E(y - Ax)}{p_{Y_i}(y)} \\ &= \sum_{i=1}^{L} \frac{w_i p_{Y_i}(y)}{p_Y(y)} \frac{p_{X_i,Y_i}(x, y)}{p_{Y_i}(y)} \\ &= \sum_{i=1}^{L} \frac{w_i p_{Y_i}(y)}{p_Y(y)} p_{X_i|Y_i}(x|y). \end{aligned}$$

Integrating in x we obtain the result. □

LEMMA A2. Suppose that Assumption 2.1 holds true. Consider the statistical linear inverse problem (3), where x is sampled from a Gaussian mixture, the noise ε is sampled from $E \sim \mathcal{N}(\mathbf{0}, \Sigma_E)$, independent of X_i and I, and $A \in \mathbb{R}^{m \times n}$. Let $Y = AX + E$. Then,

1. the density of Y_i is

$$p_{Y_i}(y) = \frac{1}{\sqrt{(2\pi)^n |A\Sigma_i A^T + \Sigma_E|}} \exp\Big(-\frac{1}{2} \|(A\Sigma_i A^T + \Sigma_E)^{-\frac{1}{2}}(y - A\mu_i)\|_2^2 \Big);$$

2. and $\mathbb{E}[X_i | Y_i = y] = \mu_i + \Sigma_i A^T (A\Sigma_i A^T + \Sigma_E)^{-1}(y - A\mu_i)$.

Proof. We prove the two parts separately.

Proof of 1: Since $X_i \perp E$, $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ and $E \sim \mathcal{N}(\mathbf{0}, \Sigma_E)$ we have

$$Y_i = AX_i + E \sim \mathcal{N}(A\mu_i, A\Sigma_i A^T + \Sigma_E).$$

The expression for the density of Y_i immediately follows.

Proof of 2: Since $Y_i = AX_i + E$ and $X_i \perp E$ we have

$$(X_i, Y_i) \sim \mathcal{N}\left(\begin{bmatrix}\mathbb{E}[X_i]\\ \mathbb{E}[Y_i]\end{bmatrix}, \begin{bmatrix}\mathrm{Var}[X_i] & \mathrm{Cov}[X_i, Y_i]\\ \mathrm{Cov}[Y_i, X_i] & \mathrm{Var}[Y_i]\end{bmatrix}\right)$$
$$= \mathcal{N}\left(\begin{bmatrix}\mu_i\\ A\mu_i\end{bmatrix}, \begin{bmatrix}\Sigma_i & \Sigma_i A^T\\ A\Sigma_i & A\Sigma_i A^T + \Sigma_E\end{bmatrix}\right).$$

Then, using (Shiryaev, 2016, theorem 2, section 13) the conditional distribution $(X_i|Y_i = y)$ is Gaussian and its expectation is given by

$$\mathbb{E}[X_i|Y_i = y] = \mu_i + \Sigma_i A^T (A\Sigma_i A^T + \Sigma_E)^{-1}(y - A\mu_i).$$

□

Appendix B. Implementation of the baseline algorithms

We collect here some additional information regarding the baseline techniques presented in Section 5. In particular, we specify the numerical algorithms we employ to minimize the involved functionals and provide further implementation details.

B.1 *LASSO*

The minimization of the LASSO functional (17), introduced in Section 5.1, can be performed employing the Iterative Soft Thresholding Algorithm (ISTA) (Daubechies *et al.*, 2004; Hale *et al.*, 2008), which is a proximal-gradient descent method involving the computation of the proximal operator for the convex and nonsmooth term of the functional (the ℓ^1-norm), and the gradient descent step for the smooth term (the data fidelity). The proximal operator of the ℓ^1-norm is the soft thresholding operator, from which ISTA takes its name. Mathematically, the $(k+1)$th iteration is

$$\beta^{k+1} = S_{t\lambda}(\beta^k - tM^T A^T(AM\beta^k - y)), \tag{B.1}$$

where $t > 0$ is a stepsize and $S_\lambda : \mathbb{R}^n \to \mathbb{R}^n$ is the soft thresholding operator defined componentwise as

$$S_\lambda(\beta)_i = \max\{|\beta_i| - \lambda, 0\}\ \mathrm{sign}(\beta_i),$$

where i indicates the component.

For the denoising problem ($A = I$), setting $t = 1$, algorithm (B.1) achieves convergence in a single step. Therefore, the solution $\bar{\beta}$ is

$$\bar{\beta} = S_\lambda(M^T A^T y). \tag{B.2}$$

B.2 *Group LASSO*

The minimization of (18) can be performed via the iterative algorithm proposed in (Yuan & Lin, 2006, proposition 1), showing strong connections with the proximal gradient descent method, which reads

$$\beta^{k+1} = \mathrm{prox}_{t\lambda f}(\beta^k - t\tilde{M}^T A^T(A\tilde{M}\beta^k - y)),$$

where $f(\beta) = \sum_{i=1}^L \|\beta_i\|_{K_i}$, $\tilde{M} = [M_1|M_2|...|M_L]$ and $t > 0$ is a stepsize. The map $\mathrm{prox}_{t\lambda f}$ satisfies (see (Beck, 2017, theorem 6.6, chapter 6))

$$\mathrm{prox}_{t\lambda f}(\beta) = (\mathrm{prox}_{t\lambda f_i}(\beta_i))_{i=1}^L,$$

being $f_i(\beta_i) = \|\beta_i\|_{K_i}$, and the proximal operator of the ℓ^2 weighted norm is (see (Beck, 2017, lemma 6.68, chapter 6))

$$\operatorname{prox}_{t\lambda f_i}(\beta_i) = \begin{cases} \beta_i - K_i^T(K_iK_i^T)^{-1}K_i\beta_i & \|(K_iK_i^T)^{-1}K_i\beta_i\|_2 \le t\lambda \\ \beta_i - K_i^T(K_iK_i^T + \alpha^* I)^{-1}K_i\beta_i & \|(K_iK_i^T)^{-1}K_i\beta_i\|_2 > t\lambda, \end{cases}$$

where α^* is the unique positive root of the nondecreasing function $g(\alpha) := \|(K_iK_i^T + \alpha I)^{-1}K_i\beta_i\|_2^2 - (t\lambda)^2$.

The resulting algorithm is computationally rather expensive: indeed, the iterative computation of the proximal operator, involving the root-finding problem, must be applied on each vector y_j separately. To ease the computational burden, in our experiments, we instead minimize (18) through the ADAM optimization scheme implemented in the *pytorch* library, which can process multiple signals in parallel.

B.3 *IHT*

The iterative algorithm to minimize (19) consists in the alternation of a gradient descent step in the direction given by the MSE and a projection onto $\mathcal{S}$, namely the $(k+1)$th iteration is

$$\beta^{k+1} = P_{\mathcal{S}}(\beta^k - tM^TA^T(AM\beta^k - y)), \tag{B.3}$$

where $P_{\mathcal{S}}(\beta) = P_{\mathcal{S}_{\bar{i}}}(\beta)$, $\bar{i} = \arg\min_i\{\|P_{\mathcal{S}_i}(\beta) - x\|_2\}$, $P_{\mathcal{S}_i}$ is the orthogonal projection onto $\mathcal{S}_i$ and $t > 0$ is a stepsize. The projection onto coordinate hyperplanes is performed by simply setting to 0 all the nonactive coordinates. The overall projection $P_{\mathcal{S}}(\beta)$ is also very efficient, since it consists only of computing L projections onto the different subspaces $\mathcal{S}_i$ and selecting the optimal one. Moreover, if $\mathcal{S}$ is the union of all coordinate hyperplanes of dimension s $P_{\mathcal{S}}(\beta)$ simply reduces to selecting the s largest components of β.

For the denoising problem ($A = I$) the IHT algorithm (B.3) with $t = 1$ converges to the solution in a single step. Therefore, using the same definitions as before the solution is

$$\bar{\beta} = P_{\mathcal{S}_{\bar{i}}}(M^Ty).$$

B.4 *Dictionary learning*

Once a dictionary has been learned through the technique discussed in Section 5.4 we are left with the minimization of the functional $\mathcal{F}$ as in (21). We do so by means of a proximal-gradient scheme, whose iterations read as

$$x^{k+1} = \operatorname{prox}_{\lambda tG}(x^k - tA^T(Ax^k - y)), \tag{B.4}$$

where $t > 0$ is a stepsize and

$$\begin{aligned} \operatorname{prox}_G(z) &= \arg\min_{x\in\operatorname{Im}(D)}\left\{\frac{1}{2}\|x - z\|^2 + \|D^+x\|_1\right\} \\ &= D\arg\min_{\beta\in\mathbb{R}^d}\left\{\frac{1}{2}\|D\beta - z\|^2 + \|\beta\|_1\right\}. \end{aligned}$$

For the denoising problem ($A = I$) the algorithm (B.4) with $t = 1$ converges to the solution $\bar{x}$ in a single step, namely

$$\bar{x} = D\operatorname{prox}_{\lambda G}(y).$$

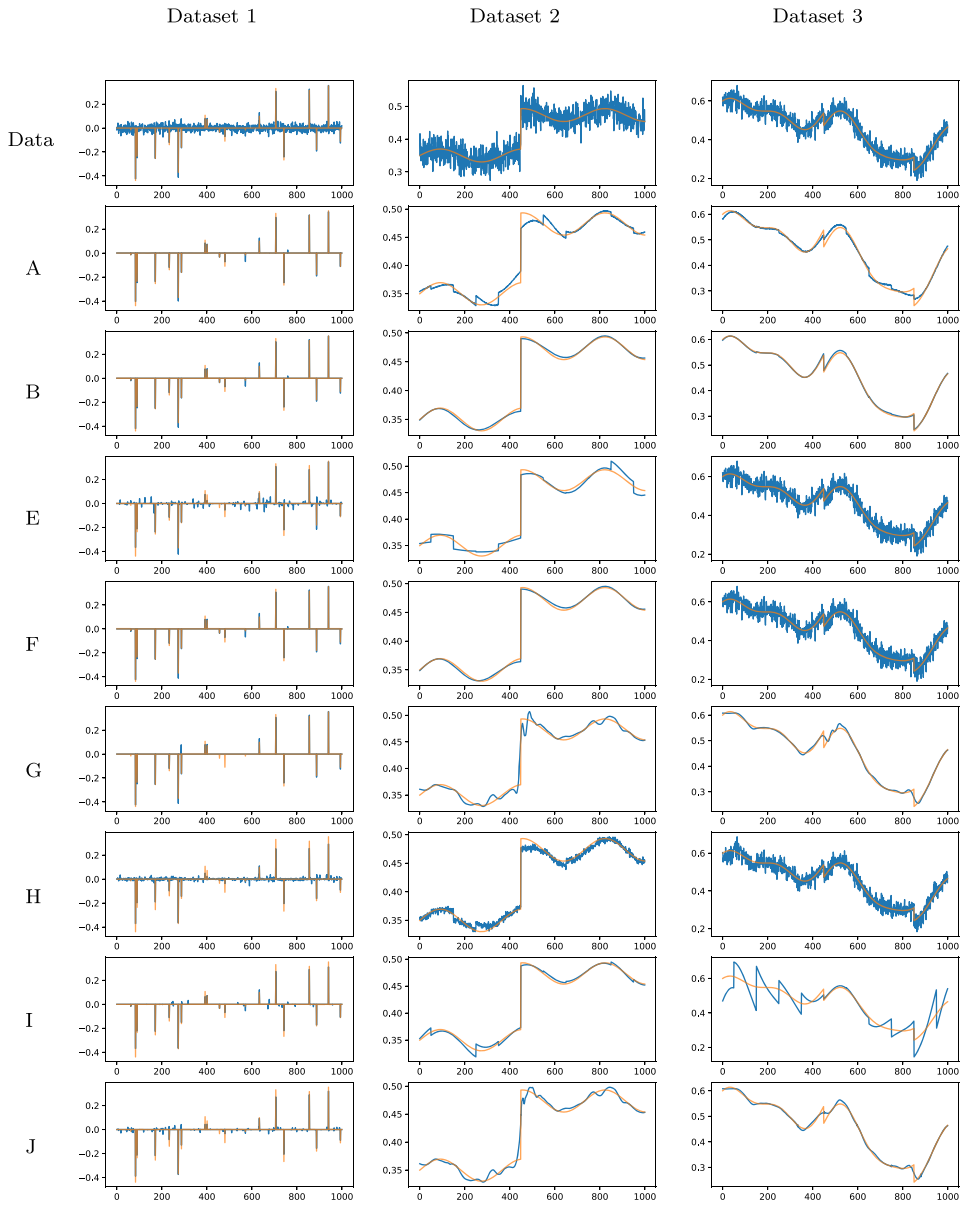

FIG. C1. Further qualitative comparisons for the denoising problem. In each column we show a signal of the test set from Datasets 1, 2 and 3, respectively. In each row we report the original data in foreground and the noisy data, and the reconstructions obtained with the randomly initialized supervised (A), unsupervised (B), IHT with SVD basis (E), IHT with SVD bases of groups (F), IHT with known basis (G), LASSO with SVD basis (H), Group LASSO with SVD bases (I) and LASSO with known basis (J) approach, in background.

Notice that, under the injectivity assumption on the matrix D, the minimization of $\mathcal{F}$ in (21) is equivalent to a LASSO problem in synthesis formulation, using D as a synthesis operator. Despite the minimizers of those two problems are the same the iterates approximating them by proximal-gradient

schemes do not coincide. For numerical reasons we prefer the formulation (21), leading to the iterations (B4). Indeed, the implementation of prox_G can be done efficiently by means of the same routines employed for dictionary learning. Specifically, we rely on the routines provided in the *scikit-learn* library (Pedregosa *et al.*, 2011), and compute prox_G through the *transform* method of the learned dictionary.

Finally, in the case of Group Dictionary Learning (22) we minimize $\mathcal{F}$ by means of a proximal-gradient scheme as in (B4), replacing prox_G by $\mathrm{prox}_{\hat{G}}$, which can be easily computed as

$$\mathrm{prox}_{\hat{G}}(z) = \mathrm{prox}_{G_I}(z), \quad I = \underset{i=1,\dots,L}{\arg\min} \left\{ \frac{1}{2}\|z - \mathrm{prox}_{G_i}(z)\|^2 + G_i\big(\mathrm{prox}_{G_i}(z)\big) \right\}.$$

Appendix C. Additional numerical results

In Fig. C1 we show a qualitative comparison for the denoising problem described in Section 6.3 between the reconstructions obtained with our supervised method (with random initialization), our unsupervised method, IHT with SVD basis, IHT with SVD bases of groups, IHT with known basis, LASSO with SVD basis, Group LASSO with SVD bases and LASSO with known basis.

In Fig. C2 we show a qualitative comparison for the deblurring problem described in Section 6.4 between the reconstructions obtained with our unsupervised method, IHT with SVD bases of groups and Group LASSO with SVD bases.

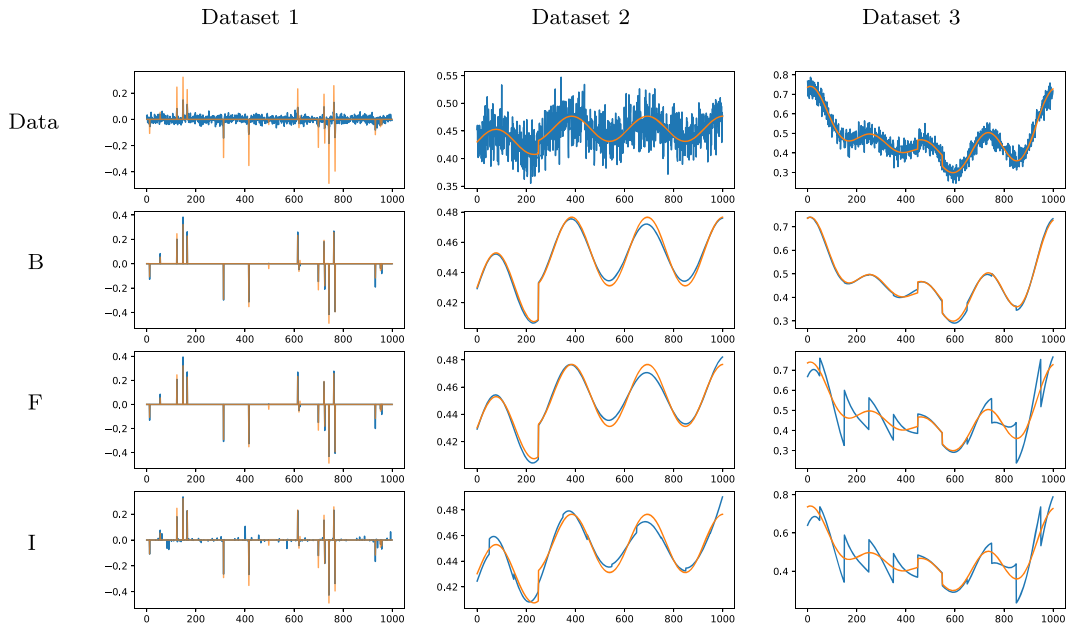

FIG. C2. Further qualitative comparisons for the deblurring problem. In each column we show a signal of the test set from Datasets 1, 2 and 3, respectively. In each row, we report the original data in foreground and the noisy data, and the reconstructions obtained with the unsupervised (B), IHT with SVD bases of groups (F) and Group LASSO with SVD bases (I) approach, in background.