entropy

MDPI

Article

Bayesian Compressive Sensing of Sparse Signals with Unknown Clustering Patterns

Mohammad Shekaramiz *, Todd K. Moon and Jacob H. Gunther

Electrical and Computer Engineering Department and Information Dynamics Laboratory, Utah State University, 4120 Old Main Hill, Logan, UT 84322-4120, USA; todd.moon@usu.edu (T.K.M.); jake.gunther@usu.edu (J.H.G.)

* Correspondence: mohammad.shekaramiz@aggiemail.usu.edu; Tel.: +1-435-797-2970

Received: 7 February 2019; Accepted: 1 March 2019; Published: 5 March 2019

Abstract: We consider the sparse recovery problem of signals with an unknown clustering pattern in the context of multiple measurement vectors (MMVs) using the compressive sensing (CS) technique. For many MMVs in practice, the solution matrix exhibits some sort of clustered sparsity pattern, or clumpy behavior, along each column, as well as joint sparsity across the columns. In this paper, we propose a new sparse Bayesian learning (SBL) method that incorporates a total variation-like prior as a measure of the overall clustering pattern in the solution. We further incorporate a parameter in this prior to account for the emphasis on the amount of clumpiness in the supports of the solution to improve the recovery performance of sparse signals with an unknown clustering pattern. This parameter does not exist in the other existing algorithms and is learned via our hierarchical SBL algorithm. While the proposed algorithm is constructed for the MMVs, it can also be applied to the single measurement vector (SMV) problems. Simulation results show the effectiveness of our algorithm compared to other algorithms for both SMV and MMVs.

Keywords: compressed sensing (CS); sparse Bayesian learning (SBL); joint sparsity; cluster structured sparsity; single measurement vector (SMV); multiple measurement vectors (MMVs)

1. Background and Introduction

Single and multiple measurement vector (SMV and MMV) problems are computational inverse problems in the compressive sensing (CS) area. CS provides the possibility of representing a sparse or compressible signal using a small set of non-adaptive linear measurements [1,2]. In linear CS, the P-dimensional signal $\mathbf{x} \in \mathbb{R}^P$ is modeled by the linear equation $\mathbf{y} = \Phi\mathbf{x}$, where $\mathbf{y} \in \mathbb{R}^M$ is the measurement vector (with $M \ll P$) and $\Phi \in \mathbb{R}^{M\times P}$ is a wide sensing matrix. The sensing matrix Φ is usually constructed from a Gaussian or Bernoulli random operator. In [3], it was shown that Φ also can be constructed from a class of circulant matrices based on deterministic sequences such as the Golay sequence [3]. In the CS context, it is further assumed that $\mathbf{x}$ is sparse under some proper basis Ψ, i.e., $\mathbf{x} = \Psi\mathbf{x}_s$, where $\mathbf{x}_s$ denotes a sparse vector. A sparse vector contains few non-zero components. Combining the two above equations, we obtain $\mathbf{y} = A\mathbf{x}_s$, where $A = \Phi\Psi$ [4]. Since A is wide, the model is underdetermined, and CS looks for a sparse (if not the most sparse) solution $\hat{\mathbf{x}}_s$ such that $\mathbf{y} = A\hat{\mathbf{x}}_s$ [5,6]. The SMV is a CS problem when A is known and the measurements are contaminated with noise $\mathbf{e}$, i.e., $\mathbf{y} = A\mathbf{x}_s + \mathbf{e}$. The case where Y and X_s are matrices is called the MMV problem, i.e., $Y = AX_s + E$. In the basic MMV model, it is assumed that all the columns of the solution matrix X_s share joint sparsity, meaning that they have the same unknown non-zero locations. Figure 1 shows an example of the MMV structure.

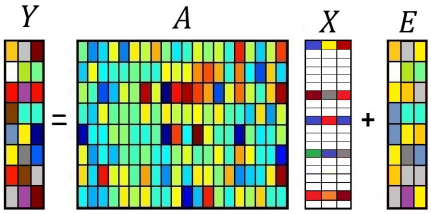


Figure 1. Example of the MMV structure.

The problem we address in this paper is for the recovery of sparse signals with an unknown clustering pattern via either SMV or MMVs. After providing an extensive survey of existing techniques in this area, we present a new hierarchical sparse Bayesian learning model and compare it to the existing algorithms. Though the formulation and modeling will be presented for the basic MMV, the proposed model is also applicable to SMV by simply considering vector cases rather than matrices in the model.

1.1. Literature Review on SMV and MMVs

Finding a sparse representation $\hat{\mathbf{x}}_s$ in the SMV problem can be achieved using greedy algorithms such as matching pursuit (MP) and orthogonal matching pursuit (OMP) [5,7] or relaxed-to-be-convex methods (such as basis pursuit de-noising (BPDN) and the in-crowd algorithm) [8,9], the class of iterative shrinkage-thresholding algorithms (ISTA), and their variations such as fast iterative shrinkage-thresholding algorithm (FISTA), NESTA (a shorthand for Nesterov's algorithm), and ISTA-NET [10–12], and sparse Bayesian learning (SBL) algorithms [13–16]. Similarly, there exist three main approaches for solving MMVs. The first approach is the extended version of the greedy-based SMV solvers such as MMV basic matching pursuit (M-BMP), MMV order recursive matching pursuit (M-ORMP), and MMV orthogonal matching pursuit (M-OMP) [17–19]. The second approach is relaxed-to-be-convex algorithms such as the joint $l_{2,0}$ approximation algorithm (JLZA) [20]. The third approach includes the SBL algorithms that are more flexible for incorporating prior knowledge on the structure of the solution compared to the greedy-based algorithms [21–23].

In some practical applications, the non-zero components of the sparse signal appear in clusters. For the MMV case, this means that in addition to the joint sparsity structure, the non-zeros also appear in clusters in each column of X_s in $Y = AX_s + E$. This feature has been referred to as the clustered structure or block-sparsity pattern in the literature [7,24,25]. Applications of clustered sparsity for the SMV cases arise in problems such as gene expression analysis [26], image reconstruction of hand-written digits [27], and audio signals using the discrete cosine transform (DCT) basis [28]. Applications of MMVs can be found in neuromagnetic imaging [17], the reconstruction stage of Xampling (compressed-sensing of analog signals) for multi-band signals [7,24], and the direction of arrival (DOA) estimation problem [29]. For example, in magnetoencephalography (MEG), the goal is to investigate the locations where most brain activities are produced. The brain activities exhibit contiguity, meaning that they occur in localized regions [25]. Therefore, the measured signal at each snapshot can be modeled as a block-sparse SMV problem.

When taking successive and almost simultaneous snapshots from the phenomena, one expects the block-sparsity structure to be preserved. Hence, it is possible to model these activities with a block-sparse MMV problem where the block partitions are unknown a priori.

During the last decade, several greedy-based algorithms have been proposed to solve clustered pattern SMVs such as reduce MMV and boost (ReMBo) algorithm [7], block-OMP [30], structured OMP (StructOMP) algorithm [31], and group LASSO [32]. However, these algorithms need prior knowledge on the block sizes or the cluster pattern.

Bayesian learning models incorporate prior knowledge on the characteristics of the underlying signal. Starting with prior knowledge, these algorithms update their belief about the underlying features of interest in an unsupervised manner using the observations. Regarding the sparse recovery of

SMV and MMVs, existing SBL algorithms can be mainly categorized into the two following approaches. The first and most common approach to impose sparsity on the solution is achieved by modeling each component of the solution with a zero-mean Gaussian prior accompanied with a Gamma distribution on the precision (inverse of variance) of the corresponding component [13,33–35]. In order to promote the clustering pattern, as well as sparsity in these models, different priors have been introduced on the variance of each component of the signal [15,28,36]. For example, in the case of clustered SMV using SBL with zero-mean Gaussian priors, Zhang and Rao incorporated the intra-block correlation structure (correlation structure in each block) [15]. In order to simplify the model, reduce the complexity, and suppress the over-fitting of the parameters in the model, they considered the same, but uncorrelated underlying covariance matrix for each possible block of the solution. The covariance matrix is updated via the expectation-maximization (EM) algorithm. In another work, Fang et al. used a zero-mean Gaussian prior where the precision on each component is statistically dependent on the precisions of the corresponding component and its two immediate neighbors [28].

The second SBL approach for the clustered sparse signal reconstruction uses a spike-and-slab prior [27,37–40]. These models have been applied to the SMV problem. Hernandez et al. proposed the generalized spike-and-slab prior, which is suitable for situations where prior information on the groups of components in the solution (that are expected to be jointly zero or jointly non-zero) is available [27]. Yu et al. made the spike and slab probabilities for each solution component depend on three possible patterns of the neighbor supports [37,40]. The patterns depend on whether the two immediate neighbors of each component are active, inactive, or only one of them is active. In [38], a Gaussian process prior was imposed on the spike-and-slab probabilities.

1.2. Idea Behind the Proposed Algorithm

In this paper, we present a new hierarchical Bayesian learning algorithm to solve the MMV problem for sparse signals with an unknown cluster pattern. We first establish a simple hierarchical Bayesian model for solving the general form of the MMV problem. In this initial model, we use the Bernoulli-Gaussian prior, which approximates the spike-and-slab prior, but in the sense that instead of employing spikes in the model, we have a binary vector that is to be learned to determine the supports of the sparse solution. Related algorithms can be found in [37,41,42]. In terms of Gaussian–Bernoulli modeling of the sparse signal, our initial model is close to the simplified form of [37,42], and in terms implementation, it uses the MCMC implementation with Gibbs sampler technique as in [37,41]. A binary matrix was used as a part of the Bayesian modeling in [41] for separating the foreground (sparse) component and the background (low-rank) component from the collection of noisy frames of a video recording. The difference between our initial model and [41] is that here, we use a binary vector to learn the supports of the solution for the *MMV problem with the joint sparsity structure*. However, this initial model only favors sparse solutions without any feature to promote the clustering pattern. The main reason for defining this initial model appears later in this paper where we modify the model not only to promote sparsity, but also to account for the clustered structure that may exist in the solution. The main contribution in this paper is related to the modified model, where we impose a prior based on the l_1-norm of the discrete gradient of the support vector $\mathbf{s}$ to promote clustering. This prior incorporates a parameter to account for the measure of contiguity, or "clumpiness," in the supports of the solution. Assigning a large value to the hyperparameter represented by this parameter encourages the overall supports of the solution to have fewer on/off transitions, that is more contiguity of clustering in the supports. Similar to the other parameters, this parameter is also to be learned via our hierarchical SBL algorithm. Previously-developed algorithms do not have this control parameter for learning the pattern via the measure of overall clumpiness over the solution.

The proposed algorithm learns the supports and the corresponding values of the active components in the solution simultaneously. This task is accomplished by defining a Bernoulli-Gaussian-inverse Gamma distribution on the solution. Based on the built-in prior modeling, the solution tends to become sparse. In contrast, existing algorithms that use Gaussian-inverse Gamma prior modeling may

need to perform some post-processing to remove the non-dominant components from the estimated solution. Existing algorithms in this area evaluate the performance based on the MSE reconstruction performance, but there are applications for which the support recovery is of equal or more importance. Examples can be found in compressive sampling and reconstruction of blind multi-narrowband signals in the continuous-to-finite (CTF) reconstruction stage, where the goal is to seek the edges of the sparsely-scattered narrowband signals to estimate the carrier frequencies [7,24]. In [7,24], the authors assumed that the number of narrowband signals is known, which yields the use of a modified OMP. The modified OMP is fed with the true sparsity level and the support block sizes of two. However, if the number of such signals is unknown, then we need an algorithm such as our proposed algorithm to learn the actual sparsity level, as well. For the problems raised in [7,24], it turns out that the supports of the solution in the CTF stage tend to clump together, which is representative of clustered pattern supports. For this problem, the supports of the solution are needed in order to figure out the locations of the actual carriers of the signals in the spectrum. Since our algorithm learns the support vector, it can naturally estimate the location of the carriers. In contrast, the other existing algorithms may tend to provide many supports, including non-dominant supports, which need to be post-processed to estimate the locations of the carriers. Another example is in [43], where the goal is to figure out the location of the sparsely-scattered multi-narrowband signals in the spectrum and then use this information to be able to fill out the empty spaces in the spectrum.

The novelty of our algorithm can be described as follows. Prior works in this area usually consider three hyperpriors commonly modeled by Gamma distributions, either on the precision of the Gaussian modeling on the solution or on the probabilities associated with Bernoulli modeling of the support vector elements, to promote clustered pattern solutions. These Gamma hyperpriors have different parameters to decide on each component of the solution based on the active/inactive status of its immediate components. By contrast, our model incorporates a total variation-like hyperprior, as a measure of the overall clustering pattern, on the support vector of the solution. Our model includes *one* control parameter in this prior to account for the emphasis on the amount of clumpiness in the support vector. This control parameter is learned in a Bayesian fashion, rather than having three different hyperpriors. Learning this parameter and the use of total variation-like prior on the support vector are novel.

Our proposed algorithm differs from [28] in two aspects. First, our model can be readily applied to either SMV or MMV problems, while the original PC-SBL algorithm proposed in [28] solves for the clustered pattern SMVs. Although it has been recently extended via generalized approximate message passing (GAMP) to solve for 2D problems [44], it needs some extra modifications to be used for the MMVs. Secondly, our model uses the Bernoulli-Gaussian prior, and it promotes the clustering pattern by adding hyperpriors on the supports of the solution, while in [28], this task is performed on the variances of the solution components. Yu et al. [37,40] used the spike-and-slab prior model and forced each mixing weight to depend on one of the three different possible active/inactive patterns of its immediate neighboring supports. This idea comes from the k-nearest neighbor approach in the clustering problems. Our approach differs from [37,40] due to the first reason we provided earlier for [28]. Tibshirani et al. presented an algorithm for SMVs referred to as the fused-lassoalgorithm, which promotes both sparsity and smoothness in the solution[26]. In this algorithm, the smoothness is promoted by using the absolute value of the difference between the estimated values of the successive components of the solution. In contrast, our proposed algorithm promotes sparsity and is able to learn the clustering pattern that may exist in the supports of the solution. The clustering pattern is learned by using the summed absolute value of the differences in the supports (our *sigma-delta* function), which distinguishes these algorithms. More specifically, we incorporate a total variation-like prior on the support vector of the solution rather than using such a prior on the solution vector itself. Finally, there are some SMV solvers that use the spike and slab prior model where the slab is modeled by a Gaussian scaled mixture model instead of just one Gaussian distribution [45]. It turns out that using this model can provide a better estimate of the underlying distribution of the non-zero elements.

However, using this model increases the number of parameters to be learned even when we have only one mixing probability parameter to learn the supports. In [45], for the purpose of reducing the complexity of the algorithm, the expectation-maximization algorithm using approximate message passing was employed. Our measure of clumpiness can also be incorporated in this model to promote the clustering pattern.

Early development of this work was presented in [46], following which significant changes have been made. We have completely changed the update rule of some of the parameters, i.e., γ_p in (15) and the controlling parameter α in (20) to learn the overall clumpiness of the supports. The convergence issue in the MCMC algorithm is addressed, and we explain how to track and monitor the convergence of the posterior distributions and make decisions on the supports based on the collected samples. Additionally, we compare the performance of our algorithm with other algorithms for both the SMV and MMVs, both on synthetic and real data.

This paper is organized as follows. In Section 2, we construct a basic hierarchical SBL model for solving the MMVs when the solution shares joint sparsity. Section 3 describes our main proposed algorithm, which extends this basic model to account for both joint sparsity and the unknown clustering pattern that may exist in the solution. In Section 4, we illustrate the performance of our proposed work compared to the other algorithms. Finally, Section 5 discusses the convergence diagnostic of the MCMC technique for the proposed algorithm. Section 6 presents conclusions.

2. Initial Model: Sparse Bayesian Learning for MMVs

As an initial model, here we construct a hierarchical SBL algorithm for the sparse recovery of basic MMVs with the joint sparsity structure that is expected to occur across the columns of the solution matrix. As discussed earlier, this model serves as the initial model, which we will modify in Section 3 for the clustered pattern sparse signals. We refer to this SBL algorithm as ordinary-SBL (O-SBL).

In this model, the supports of the solution are modeled by the binary vector $\mathbf{s}$. Therefore, the sparse solution is described by $\mathbf{s} \circ X$, where $\mathbf{s}$ and X account for the support and the solution-values, respectively, and $\circ$ denotes element-by-element multiplication (Hadamard product) applied across the columns of X. The model for the MMV problem is:

$$Y = A(\mathbf{s} \circ X) + E, \tag{1}$$

where $Y \in \mathbb{R}^{M \times N}$, $A \in \mathbb{R}^{M \times P}$, $\mathbf{s} \in \{0,1\}^{P \times 1}$, $X \in \mathbb{R}^{P \times N}$, and $E \in \mathbb{R}^{M \times N}$. The matrix Y contains N columns of observed noisy data; A denotes the known sensing matrix; $\mathbf{s}$ is an unknown binary support-learning vector; X is an unknown solution-values matrix; and E represents the measurement noise. In the product $s \circ X$, when $N > 1$, the support vector $\mathbf{s}$ deals across the columns of X. The term $\mathbf{s} \circ X$ is simply equivalent to $\text{diag}\{\mathbf{s}\} \cdot X$, where "$\cdot$" is the regular matrix product and $\text{diag}\{\cdot\}$ creates a diagonal matrix from its argument vector.

A representation of a hierarchical Bayesian graphical model of the problem used in the development of our algorithm is portrayed in Figure 2.

The shaded node Y shows the observations, and the small solid nodes represent the hyperparameters. Each unshaded node denotes a random variable (or a group of random variables) [47]. The support-learning component $\mathbf{s}$ in (1) is a binary vector, and we model the elements of $\mathbf{s}$ as Bernoulli random variables, whose probabilities are governed by the prior $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_P]^T$; that is,

$$s_p \sim \text{Bernoulli}(\gamma_p),\ \gamma_p \sim \text{Beta}(\alpha_0, \beta_0), p = 1, \ldots, P. \tag{2}$$

In order to favor the sparsity structure in $\mathbf{s}$, on the basis of experimentation, we set $\alpha_0 = \frac{1}{P}$ and $\beta_0 = 1 - \alpha_0$, as suggested in [41].

The columns of the solution-value matrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ in (1) are assumed to be drawn *i.i.d.* according to the normal-gammadistribution:

$$\mathbf{x}_n \sim \mathcal{N}(0, \tau^{-1} I_P), \tau \sim \text{Gamma}(a_0, b_0), n = 1, \ldots, N,$$

where a_0 and b_0 denote the shape and rate of the Gamma distribution, respectively. For the purpose of reducing the model complexity, we use the same precision τ for all the components of X. Moreover, due to the lack of prior knowledge on the entries of X, we experimentally set the hyper-parameters to $a_0 = b_0 = 10^{-3}$, endowing X *a priori* with a fairly high variance.

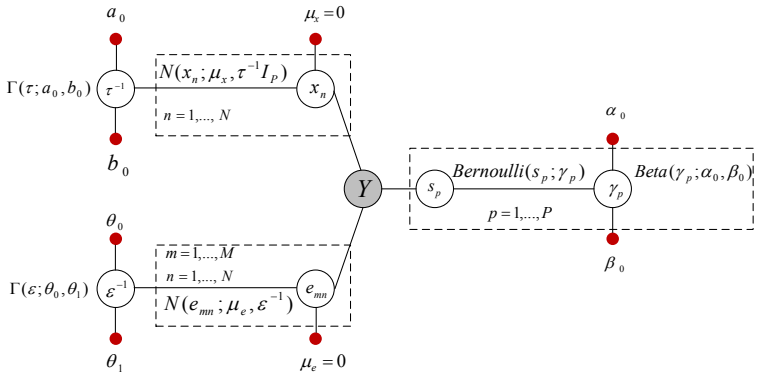


Figure 2. Graphical model of the Bayesian formulation (1).

The entries of the noise component E are assumed to be drawn *i.i.d.* from a Gaussian distribution with the precision ε^{-1}. In our model, the precision ε^{-1} is unknown and will be inferred, so that:

$$e_{mn} \sim \mathcal{N}(0, \varepsilon^{-1}),\ m = 1, \ldots, M,\ n = 1, \ldots, N,\ \varepsilon \sim \text{Gamma}(\theta_0, \theta_1). \tag{3}$$

The hyper-parameters in (3) are set to $\theta_0 = \theta_1 = 10^{-3}$. Referring to the graphical model in Figure 2, the joint probability distribution of the model can be written as:

$$p(Y, \mathbf{s}, X) \propto p(Y|\mathbf{s}, X, \varepsilon)\big(\prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{0}, \tau^{-1} I_P)\big) p(\mathbf{s}|\gamma) p(\gamma; \alpha_0, \beta_0) p(\tau; a_0, b_0) p(\varepsilon; \theta_0, \theta_1). \tag{4}$$

In the descriptions of the marginalized posterior distributions below, conditioning on $-$, as in $(s_p|-)$, is used to denote the inference on s_p conditioning upon all relevant variables (including the observations).

- $$(s_p|-) \sim \text{Bernoulli}\left(\frac{q_1}{q_0 + q_1}\right), \tag{5}$$

  where $q_1 = \gamma_p e^{-\frac{\varepsilon}{2}\left(\|\mathbf{a}_p\|_2^2 \sum_{n=1}^{N} x_{pn}^2 - 2\mathbf{a}_p^T \sum_{n=1}^{N} x_{pn} \tilde{\mathbf{y}}_n^{-p}\right)}$ and $q_0 = 1 - \gamma_p$.

  Here, $\tilde{\mathbf{y}}_n^{-p} = [\tilde{y}_{1n}^{-p}, \ldots, \tilde{y}_{mn}^{-p}]^T$, and the term $\tilde{y}_{mn}^{-p}$ is defined as:

  $$\tilde{y}_{mn}^{-p} = y_{mn} - \sum_{l \neq p}^{P} a_{ml} s_l x_{ln}.$$

  The Appendix provides more details.
- $(\gamma_p|-) \propto p(s_p|\gamma_p) p(\gamma_p; \alpha_0, \beta_0) \propto \gamma_p^{s_p} (1 - \gamma_p)^{1 - s_p} \gamma_p^{\alpha_0 - 1} (1 - \gamma_p)^{\beta_0 - 1}$

  Therefore, $(\gamma_p|-) \sim \text{Beta}(\alpha_0 + s_p\ ,\ \beta_0 + 1 - s_p)$.
- $(x_{pn}|-) \sim \mathcal{N}(\mu_{pn}, \sigma_{pn})$, where $\mu_{pn} = \varepsilon s_p \sigma_{pn} \mathbf{a}_p^T \tilde{\mathbf{y}}_n^{-p}$ and $\sigma_{pn} = (\tau + \varepsilon s_p^2 \|\mathbf{a}_p\|_2^2)^{-1}$.
- $(\tau|-) \propto p(X|0, \tau^{-1}) p(\tau; a_0, b_0) \propto \tau^{\frac{NP}{2}} e^{-\frac{\tau}{2}\|X\|_F^2} \tau^{a_0 - 1} e^{-b_0 \tau}$.

Therefore, $(\tau|-) \sim \text{Gamma}(a_0 + \frac{NP}{2}, b_0 + \frac{1}{2}\|X\|_F^2)$, where $\|.\|_F$ denotes the Frobenius norm. Finally,

- $(\varepsilon|-) \propto p(Y|\mathbf{s}, X, \varepsilon)p(\varepsilon;\theta_0,\theta_1) \propto \varepsilon^{\frac{MN}{2}} e^{-\frac{1}{2}\varepsilon\|Y-A(\mathbf{s}\circ X)\|_F^2}\varepsilon^{\theta_0-1}e^{-\varepsilon\theta_1}$.
  Thus, $(\varepsilon|-) \sim \text{Gamma}(\theta_0+\frac{MN}{2}, \theta_1+\frac{1}{2}\|Y-A(\mathbf{s}\circ X)\|_F^2)$.

In our implementation, we draw from the conditional posterior densities via Markov chain Monte Carlo (MCMC) using Gibbs sampling [41], as shown in the following pseudocode.

O-SBL Algorithm:

$\{\Theta^{(i)}\}_{i=1}^{N_{\text{collect}}} = \textbf{O-SBL}(Y, A, \Theta_0, N_{\text{burn-in}}, N_{\text{collect}})$
For Iter $= 1$ to $N_{\text{burn-in}} + N_{\text{collect}}$
% Support-learning vector component
  For $p = 1$ to P
    $\tilde{y}_{mn}^{-p} = y_{mn} - \sum_{l\neq p}^{P} a_{ml}s_l x_{ln}, \ \forall m = 1 \text{ to } M, \forall n = 1 \text{ to } N$
    $q_0 = 1 - \gamma_p$
    $q_1 = \gamma_p e^{-\frac{\varepsilon}{2}(\|\mathbf{a}_p\|_2^2 \sum_{n=1}^{N} x_{pn}^2 - 2\mathbf{a}_p^T \sum_{n=1}^{N} x_{pn}\tilde{\mathbf{y}}_n^{-p})}$
    $(s_p|-) \sim \text{Bernoulli}(\frac{q_1}{q_0+q_1})$
    % Solution-value matrix component
    For $l = 1$ to P
      $\sigma_x = (\tau + \varepsilon s_l^2 \|\mathbf{a}_l\|_2^2)^{-1}$
      $\bar{\boldsymbol{\mu}} = \varepsilon s_l \sigma_x \mathbf{a}_l$
      $\tilde{\mathbf{y}}_n^{-l} = \mathbf{y}_n - A(\mathbf{s}\circ\mathbf{x}_n) + s_l x_{ln}\mathbf{a}_l, \ \forall n = 1 \text{ to } N$
      $(x_{ln}|-) \sim \mathcal{N}(\bar{\boldsymbol{\mu}}^T\tilde{\mathbf{y}}_n^{-l}, \sigma_x), \ \forall n = 1 \text{ to } N$
    End For {l}
    $(\gamma_p|-) \sim \text{Beta}(\alpha_0 + s_p \ , \ \beta_0 + 1 - s_p)$
  End For {p}
  $(\tau|-) \sim \text{Gamma}(a_0 + \frac{NP}{2}, b_0 + \frac{1}{2}\|X\|_F^2)$
  $(\varepsilon|-) \sim \text{Gamma}(\theta_0+\frac{MN}{2}, \theta_1+\frac{1}{2}\|Y-A(\mathbf{s}\circ X)\|_F^2)$
  $\Theta^{(\text{Iter}-N_{\text{burn-in}})} \leftarrow \Theta, \quad \forall \text{Iter} > N_{\text{burn-in}}$
End For {Iter}

In the above algorithm, Θ_0 represents the set of initial values of the parameters of interest, drawn initially from the prior distributions defined in (2)–(3). We then run the O-SBL algorithm for the number of $N_{\text{burn-in}}$ iterations. Samples are not collected during the burn-in period. Then, N_{collect} more iterations are performed to collect the set of samples. For example, the estimate of the solution matrix X is computed from the sample mean, i.e., $\tilde{X} = 1/N_{\text{collect}} \sum_{n=N_{\text{burn-in}}+1}^{N_{\text{burn-in}}+N_{\text{collect}}} \hat{X}^{[n]}$, where $\hat{X}^{[n]}$ denotes the collected samples for the solution matrix obtained from the corresponding approximated posterior distribution at the n^{th} iteration. As an alternative, one may use the samples obtained from the last iteration of the collection period as the estimate of the variable of interest. For example, $\tilde{X} = \hat{X}^{[N_{\text{collect}}]}$. The MCMC convergence for the algorithm is discussed in Section 4.

3. Clustered Sparse Bayesian Learning

In this section, we modify the initial SBL model described in Section 2 to improve the support recovery performance of MMVs for the clustered sparse signals. We assume that the columns of the solution matrix are jointly sparse and that each of the vectors might have groups of clumps, i.e., groups of adjacent non-zero terms. Following [48], we measure the amount of clumpiness in the support-learning vector $\mathbf{s}$ by the absolute sum of the differences between successive elements of $\mathbf{s}$,

$$\Sigma\Delta(\mathbf{s}) = \sum_{p=2}^{P} |s_p - s_{p-1}|. \tag{6}$$

There exist fewer transitions in $\mathbf{s}$, corresponding to a smaller $\Sigma\Delta$, when the supports of the solution have a clustered pattern compared to an unstructured distribution of supports. The $\Sigma\Delta$ measure is used to establish a prior probability, which encourages clustered pattern supports by setting the prior for the support-learning vector $\mathbf{s}$ proportional to $e^{-\alpha(\Sigma\Delta)(\mathbf{s})}$ for some $\alpha > 0$. The parameter α specifies the significance of the clumpiness in the supports. Large values of α encourage more contiguity in the supports of $\mathbf{s}$. However, the measure of $\Sigma\Delta$ (total variation on the support vector) itself is not sufficient to promote *sparse* clustered pattern supports, but rather only promotes clustered pattern solutions. Here, we provide a motivational example to clarify the reasoning for this. In Figure 3, we have two signals (one sparse and the other non-sparse), where both signals are of length 100.

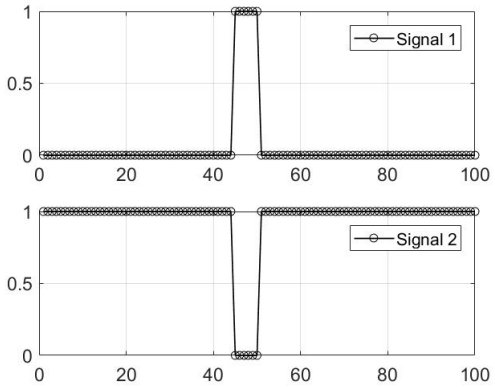


Figure 3. Example showing the effect of $\Sigma\Delta$ on the pattern of support vector $\mathbf{s}$.

According to Figure 3, both signals have the same measure of $\Sigma\Delta = 2$. As we can see in this figure, Signal 1 is sparse, while Signal 2 is non-sparse. Therefore, the measure of total variation is not sufficient to promote *sparse* clustered pattern solutions. Let us also investigate how the measure of total variation is affected by forcing a specific component in these two signals, active and inactive. The original active locations of Signal 1 are 45:50 (with MATLAB notation). Now, if we set the 44^{th} component in Signal 1 to become active, the measure of total variation does not change. Similarly, setting the 44^{th} component on Signal 2 to zero does not change the measure of total variation. This suggests that we need to further modify our model to promote sparsity, as well. For this purpose, we incorporate a binomial distribution into our prior model for $\mathbf{s}$. This distribution contains the effect of the sparsifying hyperprior γ_p and the sum over all the support components of the support vector (for both active and inactive status of s_p). Referring back to the example illustrated in Figure 3, for Signal 1, the sum over the supports of the signal (the number of active components) is six, while this measure is 94 for Signal 2. Therefore, the solution with the lower value of summation over the supports is sparser and of more interest.

We model the prior on the elements of the support-learning vector $\mathbf{s}$ as follows:

$$(s_p; \Omega_p) \sim \text{Bernoulli}(\Omega_p), \forall p = 1, 2, ..., P, \tag{7}$$

where $\Omega_p := \frac{\omega_{1,p}}{\omega_{0,p}+\omega_{1,p}}$ and:

$$\omega_{k,p} = e^{-\alpha(\Sigma\Delta)_{k,p}} \text{Binomial}(\Sigma_{k,p}, P, \gamma_p), k \in \{0, 1\}, \tag{8}$$

and where $\Sigma_{k,p}$, $k \in \{0, 1\}$, is defined as:

$$\Sigma_{k,p} := k + \sum_{i \neq p}^{P} s_p,$$

that is, it is the sum over all the elements of $\mathbf{s}$ for the case of forcing $s_p = k$. In other words, $\Sigma_{k,p}$ is the number of active elements in $\mathbf{s}$ when the p^{th} component of $\mathbf{s}$ is set to be one (active, via $k = 1$) or zero (inactive, via $k = 0$). For example, $\Sigma_{1,5} = 1 + \sum_{p \neq 5}^{P} s_p$. Furthermore, in (8), the term $(\Sigma\Delta)_{k,p}$ is the

measure of clumpiness evaluated via (6) for the case of forcing the p^{th} component of $\mathbf{s}$ equal to k. This measures how the status of s_p affects the contiguity over the supports of the solution. For example, $(\Sigma\Delta)_{0,5} = \sum_{p=2}^{P} |s_p - s_{p-1}|$, when we force $s_5 = 0$. This measures how the inactive status of s_5 affects the contiguity over the supports of the solution. The term Ω_p in the prior on the support learning vector $\mathbf{s}$ (7) can be further simplified into:

$$(s_p; \Omega_p) \sim \text{Bernoulli}(\Omega_p), \Omega_p := \frac{1}{1 + c_p e^{-\alpha(\bar{\Sigma\Delta})_p}}, \forall p, \tag{9}$$

where:

$$c_p := \frac{1 - \gamma_p}{\gamma_p} \frac{\Sigma_{1,p}}{P - \Sigma_{0,p}}, \forall p = 1, \ldots, P, \tag{10}$$

and:

$$(\bar{\Sigma\Delta})_p = (\Sigma\Delta)_{0,p} - (\Sigma\Delta)_{1,p}, \forall p = 1, \ldots, P. \tag{11}$$

Roughly speaking, this distribution favors drawing $s_p = 1$ if this draw reduces $(\Sigma\Delta)_{1,p}$. Define $\bar{c}_p := (1 - \gamma_p)/(\gamma_p)$ and $\zeta_p := \Sigma_{1,p}/(P - \Sigma_{0,p}) e^{-\alpha(\bar{\Sigma\Delta})_p}$, and thus, $\Omega_p = 1/(1 + \bar{c}_p \zeta_p)$. If we set $\zeta_p = 1$ in Ω_p, then the prior on s_p would be only governed by $\bar{c}_p$. In this case, Ω_p in (9) will be simplified into $\Omega_p = \gamma_p$, which is the same prior as we had earlier in (2). This prior only tends to promote sparsity without favoring the clustered pattern supports. By contrast, setting $\bar{c}_p = 1$ in Ω_p, for a sufficiently large value of α and $(\bar{\Sigma\Delta})_p > 0$, favors clustered pattern supports, which may lead to non-sparse solutions. Therefore, by incorporating both c_p and the exponential term in the prior on s_p, defined in (9), the supports exhibit a trade off between sparsity and clustering.

In Figure 4, we demonstrate the effect of α defined in (8) and its role in the prior (9) on the support learning vector $\mathbf{s}$. The figure shows draws of $\mathbf{s}$ according to (9) using (10) and (11).

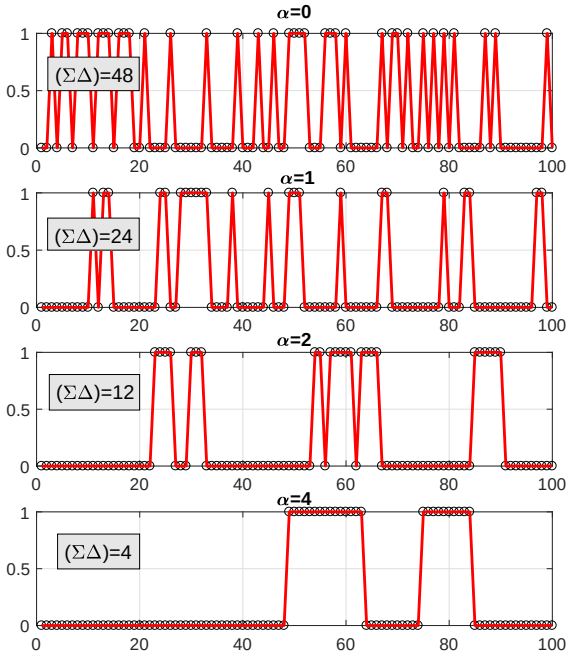


Figure 4. Example showing the effect of α on the pattern of support vector $\mathbf{s}$.

Larger values of α result in lower $\Sigma\Delta(\mathbf{s})$, meaning that the support vector $\mathbf{s}$ will tend to have fewer transitions along its components, promoting clustering.

We employ a Gamma prior $\alpha \sim \text{Gamma}(a_1, b_1)$ on α, where a_1 and b_1 are hyperparameters denoting the shape and rate of the Gamma distribution, respectively. In order to promote the clustering pattern as a prior knowledge, we set $a_1 = 2 \times 10^{-3}$ and $b_1 = 10^{-3}$, meaning that on average, we expect to have $\alpha = 2$ before incorporating the measurements. By setting these parameters to small values, as we have here, the learning process of α will not be biased by the prior, once the measurements are incorporated.

With these priors, the joint probability distribution for the complete model becomes:

$$\begin{aligned} p(Y,\mathbf{s},X) \propto p(Y|\mathbf{s},X,\varepsilon)\Big(\prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{0},\tau^{-1}I_P)\Big)p(\tau;a_0,b_0)\times \\ p(\varepsilon;\theta_0,\theta_1)\Big(\prod_{p=1}^{P}(s_p|\Omega_p)\Big)p(\gamma|\mathbf{s},\alpha_0,\beta_0)p(\alpha;a_1,b_1). \end{aligned} \tag{12}$$

The graphical model for the clustered pattern MMVs is shown in Figure 5.

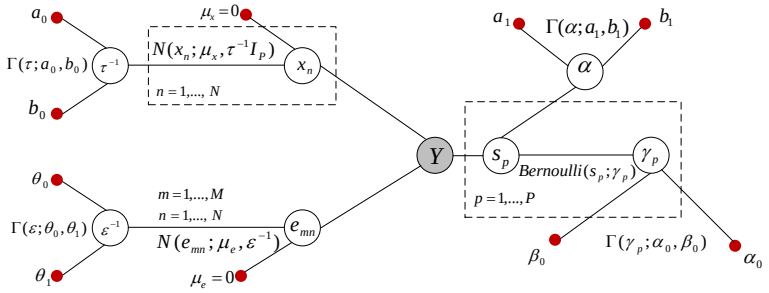


Figure 5. Graphical model of the Bayesian formulation (12).

Below, we describe the inference on the variables that are modified by (12) compared to (4). The inference on the other variables and parameters in the model is the same as those we described in Section 2.

- The posterior for s_p is given by:

$$p(s_p|-) \propto p(Y|s_p,X)p(s_p|\Omega_p) \propto \text{Bernoulli}(s_p|\Omega_p)e^{-\frac{\varepsilon}{2}\left(\|\mathbf{a}_p\|_2^2(\sum_{n=1}^{N}x_{pn}^2)s_p^2-2\mathbf{a}_p^T(\sum_{n=1}^{N}x_{pn}\tilde{\mathbf{y}}_n^{-p})s_p\right)},$$

  where Ω_p was defined in (7). Using the fact that s_p is a binary random variable, the posterior inference on s_p can be simplified to $(s_p|-) \sim \text{Bernoulli}(Q_p)$, where $Q_p = \frac{q_{1,p}}{q_{0,p}+q_{1,p}}$,

$$q_{0,p} = \frac{1}{1+(\frac{\gamma_p}{1-\gamma_p})(\frac{P-\Sigma_{0,p}}{\Sigma_{1,p}})e^{-\alpha\left((\Sigma\Delta)_{1,p}-(\Sigma\Delta)_{0,p}\right)}},$$

  and:

$$q_{1,p} = (1-q_{0,p})e^{-\frac{\varepsilon}{2}\left(\|\mathbf{a}_p\|_2^2(\sum_{n=1}^{N}x_{pn}^2)-2\mathbf{a}_p^T(\sum_{n=1}^{N}x_{pn}\tilde{\mathbf{y}}_n^{-p})\right)}.$$

  The posterior for s_p can be further simplified into:

$$(s_p|-) \sim \text{Bernoulli}\Big(\frac{1}{1+c_p\kappa_p e^{-\alpha(\bar{\Sigma\Delta})_p}}\Big), \tag{13}$$

  where c_p and $(\bar{\Sigma\Delta})_p$ were defined in (10) and (11), respectively, and:

$$\kappa_p := e^{\frac{\varepsilon}{2}\left(\|\mathbf{a}_p\|_2^2(\sum_{n=1}^{N}x_{pn}^2)-2\mathbf{a}_p^T(\sum_{n=1}^{N}x_{pn}\tilde{\mathbf{y}}_n^{-p})\right)}. \tag{14}$$

- The posterior for γ_p is given by $p(\gamma_p|-) \propto p(\gamma_p;\alpha_0,\beta_0)\prod_{k=0}^{1}p(\omega_{k,p}|\Sigma_{k,p},\gamma_p,\alpha,\mathbf{s})$. Thus, $(\gamma_p|-) \sim \text{Beta}(\alpha_1,\beta_1)$, where:

$$\alpha_1 := \alpha_0+1+2\sum_{i\neq p}^{P}s_i\ ,\ \beta_1 := \beta_0-1+2(P-\sum_{i\neq p}^{P}s_i). \tag{15}$$

- The update rule for α is determined as follows. Using the joint probability distribution of the complete model (12) and discarding the terms that are unrelated to α, we have:

$$\begin{aligned} p(\alpha|-) &\propto p(\alpha; a_1, b_1) \prod_{p=1}^{P} p(s_p|\Omega_p) \\ &\propto \alpha^{a_1-1} e^{-b_1\alpha} \prod_{p=1}^{P} \text{Bernoulli}\left(\frac{1}{1+c_p e^{-\alpha(\bar{\Sigma\Delta})_p}}\right). \end{aligned} \tag{16}$$

Due to the complicated nature of (16), we estimate α based on the current value of all the other variables and parameters of the model in the implemented MCMC approach. In other words, we compute:

$$\hat{\alpha}^{[t+1]}_{MAP|Y,X^{[t]},s^{[t]},\Theta^{[t]}} = \arg\max_{\alpha} L_{\alpha|Y,X^{[t]},s^{[t]},\Theta^{[t]}}, \tag{17}$$

where L_α is obtained by taking the logarithm of (16) and is defined as:

$$\begin{aligned} L_\alpha &= \log\{p(\alpha; a_1, b_1) \prod_{p=1}^{P} p(s_p|\Omega_p)\} \\ &\propto (a_1-1)\log\alpha - b_1\alpha + \sum_{p=1}^{P} \{s_p \log\Omega_p + (1-s_p)\log(1-\Omega_p)\}, \end{aligned} \tag{18}$$

and may be further simplified into:

$$L_\alpha \propto (a_1-1)\log\alpha - b_1\alpha - \alpha\sum_{p=1}^{P}(1-s_p)(\bar{\Sigma\Delta})_p - \sum_{p=1}^{P}\log\{1+c_p e^{-\alpha(\bar{\Sigma\Delta})_p}\}. \tag{19}$$

Taking the derivative of L_α with respect to α and equating the resulting equation to zero, we obtain:

$$\frac{a_1-1}{\alpha} - b_1 - \sum_{p=1}^{P}(1-s_p)(\bar{\Sigma\Delta})_p + \sum_{p=1}^{P}\frac{(\bar{\Sigma\Delta})_p}{1+\frac{1}{c_p}e^{\alpha(\bar{\Sigma\Delta})_p}} = 0.$$

The maximum a *posteriori* point estimate of α conditioned on the measurements and the current estimate of hidden variables and the parameters of the model can be obtained by iteratively solving:

$$\frac{a_1-1}{\alpha^{[t+1]}} - b_1 - \sum_{p=1}^{P}(1-s_p^{[t]})(\bar{\Sigma\Delta})_p^{[t]} + \sum_{p=1}^{P}\frac{(\bar{\Sigma\Delta})_p^{[t]}}{1+\frac{1}{c_p^{[t]}}e^{\alpha^{[t+1]}(\bar{\Sigma\Delta})_p^{[t]}}} = 0 \tag{20}$$

This update is computed at each iteration of the MCMC approach.

As an alternative approach, one can set α to a fixed predefined value. If under some prior knowledge, no clustering pattern is expected in the solution, one can set α to a small value close to zero. In case of expecting a highly clustered solution, we recommend setting $\alpha \gg 0$.

Remark 1. *In [46], the marginal posterior inference on γ_p was estimated by $(\gamma_p|-) \sim Beta(\alpha_0+s_p, \beta_0-s_p)$, $\forall p=1,\ldots,P$. The idea behind the above update rule was the assumption of having a directed link from the node γ_p to the node s_p in Figure 5. However, in (15), we have removed this assumption and directly found the inference based on the relationship between the variables that we have in Figure 5. Furthermore, we have provided an analytical approach for the update rule of α, rather than the empirical approach discussed in [46].*

The pseudocode below describes our algorithm, referred to as C-SBL, which works for the clustered patterns SMV or MMVs.

C-SBL Algorithm:

$\{\Theta^{(i)}\}_{i=1 \text{ to } N_{\text{collect}}} = \textbf{C-SBL}(Y, A, \Theta_0, N_{\text{burn-in}}, N_{\text{collect}})$

For Iter $= 1$ to $N_{\text{burn-in}} + N_{\text{collect}}$

% Support-learning vector component

For $p = 1$ to P

$\tilde{y}_{mn}^{-p} = y_{mn} - \sum_{l \neq p}^{P} a_{ml} s_l x_{ln}, \quad \forall m = 1 \text{ to } M, \forall n = 1 \text{ to } N$

$c_p = \frac{1-\gamma_p}{\gamma_p} \frac{\Sigma_{1,p}}{P+1-\Sigma_{1,p}}$, $(\bar{\Sigma\Delta})_p = (\Sigma\Delta)_{0,p} - (\Sigma\Delta)_{1,p}$

$k_p = e^{\frac{\varepsilon}{2}\left((\|\mathbf{a}_p\|_2^2 \sum_{n=1}^{N} x_{pn}^2) - 2\mathbf{a}_p^T (\sum_{n=1}^{N} x_{pn} \tilde{\mathbf{y}}_n^{-p}) \right)}$

$(s_p|-) \sim \text{Bernoulli}(\frac{1}{1 + c_p k_p e^{-\alpha (\bar{\Sigma\Delta})_p}})$

% Solution-value matrix component

For $l = 1$ to P

$\sigma_x = (\tau + \varepsilon s_l^2 \|\mathbf{a}_l\|_2^2)^{-1}$

$\bar{\boldsymbol{\mu}} = \varepsilon s_l \sigma_x \mathbf{a}_l$

$\tilde{\mathbf{y}}_n^{-l} = \mathbf{y}_n - A(\mathbf{s} \circ \mathbf{x}_n) + s_l x_{ln} \mathbf{a}_l, \ \forall n = 1, \ldots, N$

$(x_{ln}|-) \sim \mathcal{N}(\bar{\boldsymbol{\mu}}^T \tilde{\mathbf{y}}_n^{-l}, \sigma_x), \ \forall n = 1, \ldots, N$

End For $\{l\}$

$(\gamma_p|-) \sim \text{Beta}\big(\alpha_0 + 1 + 2\sum_{k \neq p}^{P} s_k \ , \ \beta_0 - 1 + 2(P - \sum_{k \neq p}^{P} s_k)\big)$

End For $\{p\}$

$(\tau|-) \sim \text{Gamma}(a_0 + \frac{NP}{2}, b_0 + \frac{1}{2}\|X\|_F^2)$

$(\varepsilon|-) \sim \text{Gamma}(\theta_0 + \frac{MN}{2}, \theta_1 + \frac{1}{2}\|Y - A(\mathbf{s} \circ X)\|_F^2)$

α : obtained from solving (20) for $\alpha^{[t+1]}$

$\Theta^{(\text{Iter} - N_{\text{burn-in}})} \leftarrow \Theta, \quad \forall \text{Iter} > N_{\text{burn-in}}$

End For {Iter}

Similar to the O-SBL algorithm, we perform MCMC inference in the implementation of the C-SBL using Gibbs sampling for all the variables and parameters of the model.

4. Simulation Results

Here, we compare the performance of our algorithm against other algorithms on both synthetic/simulated and real-world data for both the SMV and MMV problems.

4.1. Simulations on Synthetic Data

We first compare our proposed algorithm with other algorithms for the SMV problem, i.e., $N = 1$ defined in (1). We then consider the MMV problem for the case where the number of columns in the solution matrix X is $N = 2, 5$.

4.1.1. Performance for the SMV Problem

Each independently-generated trial is constructed as follows. We consider the solution-value vector $\mathbf{x} \in \mathbb{R}^{100}$, i.e., $P = 100$ and $N = 1$ in (1). The supports of the true solution are drawn randomly so that the support vector $\mathbf{s}$ exhibits a random clustered sparsity pattern. The total number of non-zeros in the sparse solution ($\mathbf{x}_s = \mathbf{s} \circ \mathbf{x}$) is set to 25 for all the trials. The nonzero elements of $\mathbf{x}$ at the active supports of $\mathbf{s}$ are drawn *i.i.d.* from a zero-mean Gaussian distribution with variance $\sigma_x^2 = 1$. For each trial, the entries of the sensing matrix $A \in \mathbb{R}^{M \times 100}$ are drawn *i.i.d.* from a zero-mean Gaussian distribution with variance one, and then, we normalize the columns of A. We vary the number of measurements M to show the performance as the ratio $\lambda = M/P$ changes. The elements of the noise

component are drawn *i.i.d.* from a Gaussian distribution $e_m \sim \mathcal{N}(0, \sigma^2)$. The SNR for all trials was 25 dB and is defined as SNR $= 20\log_{10}(\sigma_x/\sigma)$. The measurement $\mathbf{y}$ is computed from $\mathbf{y} = A\mathbf{x}_s + \mathbf{e}$. The data described above were generated for 200 trials.

The recovery performance of the algorithms is demonstrated using both probability of support recovery and mean squared error. The probability of correct detection of a support location (probability of detection) P_D and the probability of (erroneously) detecting a support location where there is none (probability of false alarm) P_{FA} are respectively defined as $P_D =$ (#Correct detections)/(#Possible correct detections), $P_{FA} =$ (#False detections)/ ((#Possible detections) − (#Correct detections)). A successful reconstruction is reported when all the supports of the true solution are recovered. The normalized mean-squared error is defined as:

$$\text{NMSE (dB)} := 20\log_{10}\frac{\|\mathbf{x}_s - \hat{\mathbf{x}}_s\|_2}{\|\mathbf{x}_s\|_2}, \tag{21}$$

where $\hat{\mathbf{x}}_s$ is the estimated solution.

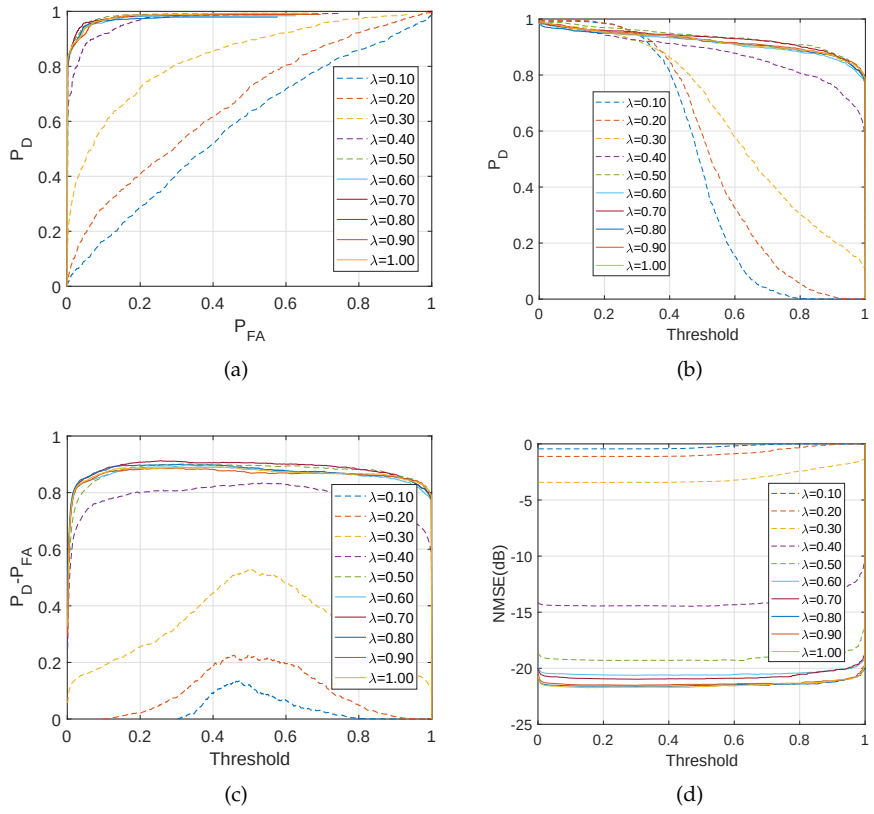


Figure 6. Aspects of performance of the proposed C-SBL algorithm for the SMV problem. (**a**) Empirical ROC; (**b**) Detection rate; (**c**) $P_D - P_{FA}$; (**d**) NMSE (dB).

Figure 6a–d demonstrates the aspects of the performance of the C-SBL algorithm. Figure 6a shows the performance of C-SBL using receiver operating characteristic (ROC) curves as the number of measurements, and equivalently the ratio $\lambda = M/P$ varies. For $\lambda > 0.4$ ($M > 40, P = 100$), C-SBL successfully finds all the supports of the true solution with generally a low false alarm rate. The algorithm exhibits high performance when the number of measurements is almost twice the number of true non-zeros in the solution (the true number of non-zeros was set to 25). In Figure 6b–d, we also illustrate the performance of C-SBL *vs.* the threshold, where the threshold is defined as follows. We average over all of the N_{collect} collected samples of the support learning vector, where each component belongs to $\{0, 1\}$. In other words, we compute $\hat{\mathbf{s}}_{ave} = 1/N_{\text{collect}} \sum_{n=N_{\text{burn-in}}+1}^{N_{\text{collect}}} \hat{\mathbf{s}}^{[n]}$. Then, those indices in the resulting vector $\hat{\mathbf{s}}_{ave}$ that contain values greater than the threshold are chosen as the estimated supports of the solution (setting the threshold to 0.5 results in the sample mean of the

collected samples). In Figure 6b, we illustrate the detection rate *vs.* the threshold as the ratio M/P changes (if we decide on the supports based on all the samples obtained in both burn-in and collected periods, then the detection rate would become zero for the threshold of one for all λ). Figure 6c shows the difference between detection rate and false alarm rate of C-SBL *vs.* the threshold. The higher values of $P_D - P_{FA}$ indicates the higher overall support recovery of the algorithm. In Figure 6c, there is a threshold of around 0.5, where $P_D - P_{FA}$ has a peak for $\lambda \leq 0.4$. For the case of $\lambda > 0.4$, C-SBL reaches its highest performance with a wide range of threshold of approximately $[0.1, 0.9]$. This verifies that estimating the support learning vector based on the sample mean (threshold of 0.5) provides a high performance for all λ. Finally, Figure 6d shows the C-SBL behavior in terms of normalized mean-squared error, defined in (21), *vs.* the threshold. In Figure 6d, the error term for each λ remains almost constant regardless of the threshold. This means that one can take a threshold that leads to a very high detection rate, even for a very low number of measurements, without any major change in terms of the error. However, according to Figure 6a and Figure 6c, different choices of the threshold result in different false alarm rates. According to Figure 6d, as the number of measurements becomes around twice the sparsity level $(\lambda \geq 0.5)$, the error becomes almost negligible.

We now compare the performance of C-SBL and O-SBL with other algorithms, specifically: CLUSS-MCMC algorithm for solving clustered structure compressive sensing using Markov chain Monte Carlo method [37], orthogonal matching pursuit (OMP) algorithm [5,49], MMV focal underdetermined system solver (MFOCUSS) [17], block-sparse Bayesian learning algorithm (BSBL) [15,23], MMV sparse Bayesian learning algorithm (MSBL) [22], basis pursuit denoising algorithm for group sparsity (BPDN-group) using the spectral projected gradient for l_1 minimization (SPGL1) solver [50], the single-task version of multi-task compressive sensing algorithm (MTCS) [34,51], and PC-SBL [28,52]. In all of the algorithms, we discarded those estimated elements in the solution with an amplitude less than 0.01 from the support set. In Figure 7a–d, the results for the O-SBL and C-SBL algorithms are based on the sample mean of the collected samples.

In Figure 7a, we demonstrate the empirical results of detection rate *vs.* the ratio M/P.

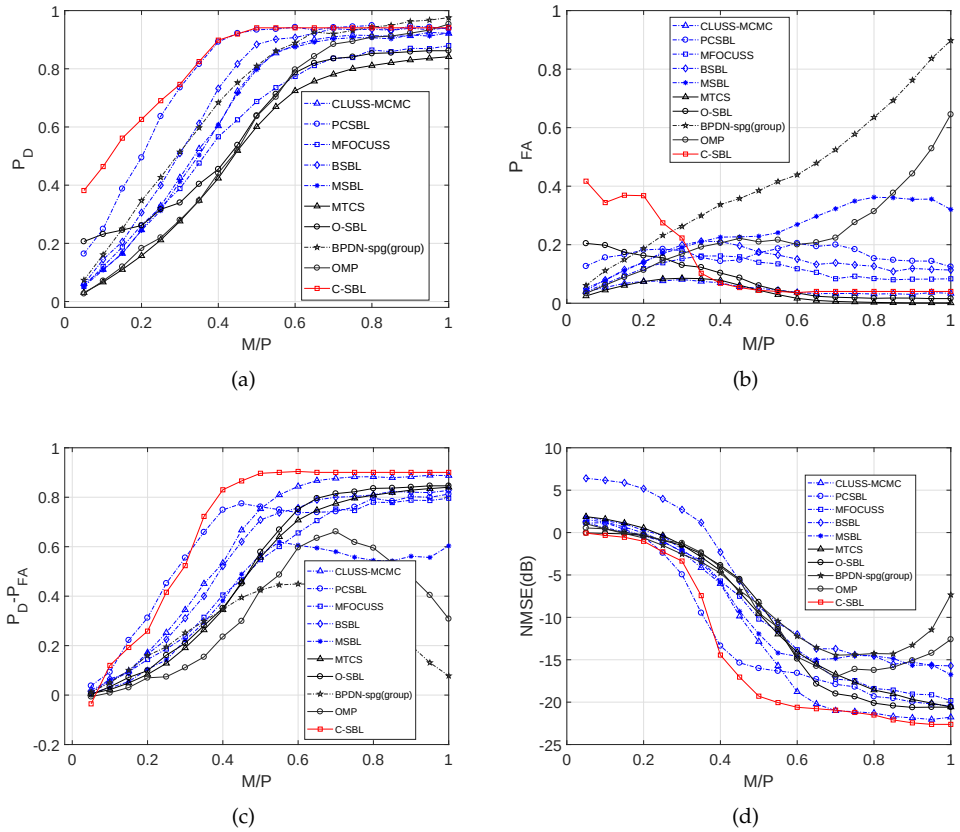


Figure 7. Comparisons of various algorithms in the SMV case. (**a**) Detection rate; (**b**) False alarm rate; (**c**) $P_D - P_{FA}$; (**d**) NMSE (dB).

Figure 7a shows that C-SBL provides the best performance in terms of detecting the true supports of the solution. In Figure 7b, the false alarm rate in support recovery is illustrated, where we see that for $M/P < 0.35$, our algorithm has a higher false alarm rate in support recovery at the cost of providing a higher detection rate within the same range for M/P. In contrast, the rates for C-SBL, O-SBL, CLUSS-MCMC, and MTCS become almost the same and have the lowest values.

Figure 7c compares the performance algorithms in terms of the trade off between the detection rate and false alarm rate in support recovery, in which C-SBL and PC-SBL show almost the same performance for $M/P < 0.35$. However, C-SBL outperforms all the other algorithms for $M/P > 0.35$. Figure 7d illustrates the comparison of NMSE between the true and estimated solution. We observe that C-SBL provides lower error among the other algorithms for a wide range of M/P.

Remark 2. *The MATLAB codes for BSBL, MSBL, and MFOCUSS were obtained from [53]. For BSBL, the noise flag was set to two (small noise). and the block-size of $h = 2$ was considered. For MSBL, we activated the option for learning the tuning parameter and initialized it by the true noise variance. For the MFOCUSS algorithm, the regularization parameter was set to 10^{-3}. Based on some initial experiments, we decided to use the default settings for CLUSS-MCMC [54] and PCSBL [52]. The parameters of the Gamma prior on the noise variance for MTCS [51] were both set to one.*

It has been very common in the literature to demonstrate the performance primarily on NMSE, so that successful recovery is reported when the NMSE becomes lower than some pre-defined value. In that sense and by referring to Figure 7d, we see that our algorithm provides the lowest error rate for a wide range of M/P. In addition, our algorithm demonstrates good performance on the ROC plot, showing high detection against the false alarm rate. Finally, in Figure 8, we show the average run-time comparison of the respective algorithms for 500 randomly-generated SMVs in the same way stated earlier. In Figure 8, the legend C-SBL(learning α) denotes the execution time of the C-SBL algorithm for the case where the algorithm learns α from (20), while in C-SBL($\alpha = 1$), we do not use (20) and instead experimentally set $\alpha = 1$.

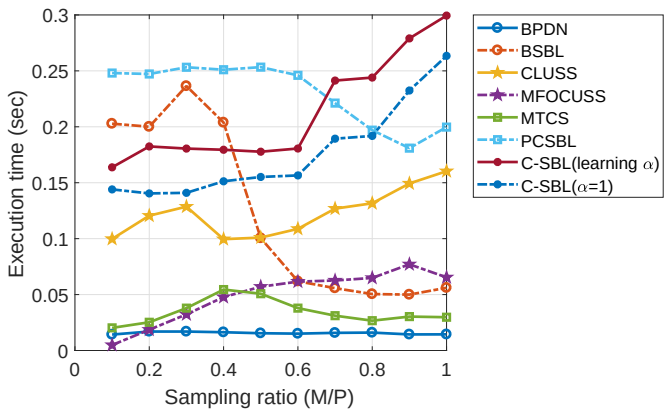


Figure 8. Execution-time comparison for the SMV case.

According to Figure 8, for low sampling ratios, which is of more interest in CS problems, the lowest execution times belong to THE PBDN, MFOCUSS, MTCS, and CLUSS-MCMC algorithms, respectively. However, as we showed in Figure 7, these algorithms do not provide good accuracy in signal reconstruction compared to the other algorithms. Furthermore, these algorithms do not account for learning the unknown clustering pattern. Comparing the execution time of the PC-SBL, B-SBL, and C-SBL algorithms for low sampling ratios, we observe that C-SBL demonstrates reasonably low execution time. Furthermore, notice that the C-SBL and CLUSS-MCMC algorithms are the only algorithms that are implemented using the MCMC method, and based on Figure 8, we observe that they both show the same behavior with respect to the change of the sampling ratio. The reason for the higher execution time of C-SBL against CLUSS-MCMC is again due to the extra computations that C-SBL requires to account for the clustering pattern.

4.1.2. Performance for the MMV Problem with $N=2$

We first demonstrate the performance of C-SBL in terms of detection rate and false alarm rate in support recovery as the ratio M/P and NMSE.

Figure 9a displays ROC curves. Comparing the results demonstrated in Figure 6a with Figure 9a, increasing the number of columns in the solution from $N = 1$ to $N = 2$ provides considerable improvement in the support recovery. Figure 9b illustrates the detection rate of C-SBL for the MMVs (with $N=2$) *vs.* different threshold values. Once $M/P \geq 0.4$ (over 40% compression and the sparsity of 25), almost full success in support recovery is attained regardless of the selected threshold value. The difference between the detection rate and false alarm rate of C-SBL *vs.* the threshold is shown in Figure 9c. There is a threshold of around 0.5, where $P_D - P_{FA}$ has a peak for $\lambda \leq 0.4$. For $\lambda > 0.4$ in the MMV case, C-SBL reaches its highest performance almost regardless of the chosen threshold value. Therefore, we make a decision on the supports based on computing the sample mean, i.e., setting the threshold equal to 0.5. Figure 9d shows the NMSE *vs.* the threshold for the clustered MMV problem using the C-SBL algorithm.

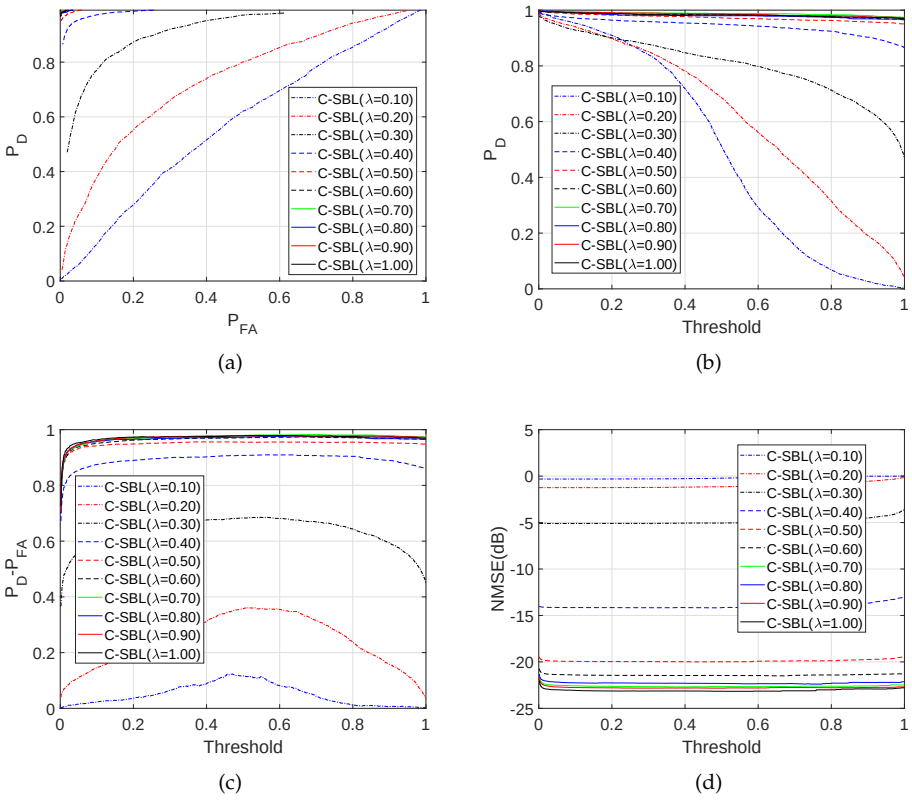


Figure 9. Aspects of the performance of C-SBL for MMV (with $N = 2$). (**a**) Empirical ROC; (**b**) Detection rate; (**c**) $P_D - P_{FA}$; (**d**) NMSE (dB).

Finally, Figure 10a–d compares the performance of C-SBL against the other algorithms. We compare the results of C-SBL with the following algorithms: MFOCUSS [17], MSBL [22], T-MSBL [55,56], and MTCS [34]. Notice that T-MSBL is devised for correlated signals, while our model does not account for this feature. Although MTCS does not promote the clustering pattern, we use it as a baseline for our comparisons. We consider two sets of simulations. Our setup for the simulations is similar to the SMV case, as was described earlier. The only difference is in generating the true solution matrices. In the first case, we generate uncorrelated columns for the solution matrices. In the legend of Figure 10a–d, the uncorrelated cases are denoted by $\rho = 0$. In the second case, the columns of the solution matrix for each trial are correlated with the correlation factor of $\rho = 0.85$.

Figure 10a illustrates the detection rate in support recovery for clustered pattern MMVs (with $N=2$), in which we observe that C-SBL has the highest performance among the other algorithms for the uncorrelated case. Furthermore, we observe that C-SBL competes with T-MSBL for the correlated case. In Figure 10b, it is clear that C-SBL provides a lower false alarm rate in terms of support recovery

compared to the MSBL and T-MSBL algorithms. The best performance belongs to MTCS, but it provided the lowest performance in terms of detection rate, as was shown in Figure 10a. For overall comparison, Figure 10c shows the simulation results in terms of the difference between the detection rate and false alarm rate in support recovery when varying the ratio M/P. According to Figure 10c, the overall performance of C-SBL is higher than the other algorithms. The NMSE comparisons are illustrated in Figure 10d, where we see that C-SBL provided the lowest error. According to Figure 10a–d, C-SBL is more successful than the compared algorithms in terms of both support recovery and estimating the non-zero values of the true solution.

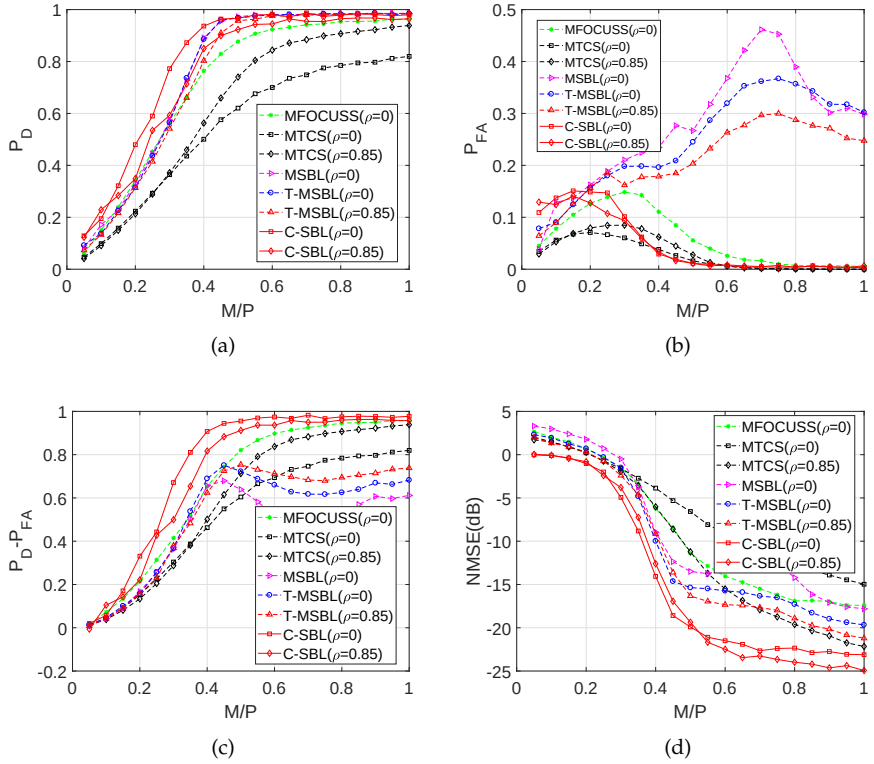


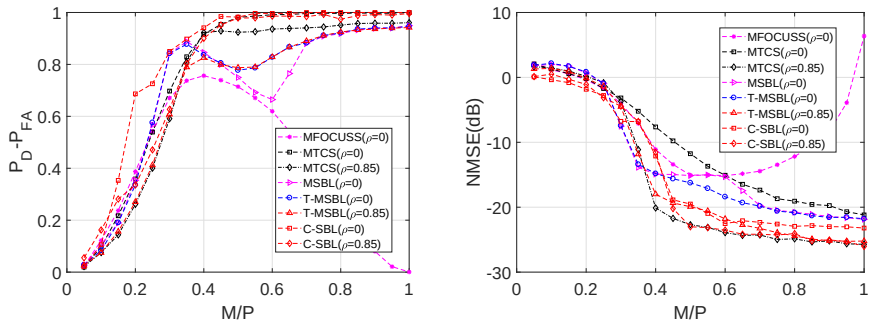
Figure 10. Comparison of various algorithms in the MMV case with $N = 2$. (**a**) Detection rate; (**b**) False alarm rate; (**c**) $P_D - P_{FA}$; (**d**) NMSE (dB).

4.1.3. Performance for the MMV Problem with $N = 5$

Here, we perform simulations on synthetically-generated data in the same way explained previously except setting $N = 5$. The burn-in and collection periods of C-SBL were set to 2000 and 1000, respectively. Figure 11 illustrates the performance comparison results for both the uncorrelated case ($\rho = 0$) and the correlated case (with $\rho = 0.85$).

According to Figure 11a, the best performance in terms of the difference between the detection rate and false alarm rate belongs to the C-SBL algorithm. The lowest performance belongs to FOCUSS, where as the sampling ratio becomes greater than 0.4, FOCUSS starts to activate more components in $\mathbf{s}$. As a result, the false alarm rate increases, and this yields the smaller $P_D - P_{FA}$. For a sampling ratio of one, all the components of $\mathbf{s}$ are active, resulting in $P_D - P_{FA} = 0$. The performance of FOCUSS would be still acceptable if the NMSE for high sampling ratios would have became very low, meaning that the wrongly-determined supports had very low amplitudes. However, according to the error curve of FOCUSS in Figure 11b, this does not happen, meaning that MFOCUSS crashed for moderately high and high sampling ratios. For sampling ratios $\lambda > 0.5$, the best performance in terms of $P_D - P_{FA}$ belongs to both C-SBL and MTCS. Notice that they were both able to provide almost full support recovery. However, for low sampling ratios like $\lambda \leq 0.4$, the best support recovery belongs to C-SBL. For example, for $\lambda = 0.2$, the C-SBL provided $P_D - P_{FA}$ of around 0.7, while the other algorithms provided values less than 0.4. This should justify the merit of the C-SBL algorithm. In summary, C-SBL demonstrates the best performance in support recovery for the uncorrelated case.

Figure 11b compares the error between the true and estimated solution. C-SBL provides the lowest error for the most possible range of λ ($\lambda \in [0.05, 0.25) \cup [0.45, 1]$) for the uncorrelated case.



(a) (b)

Figure 11. Comparison of various algorithms in the MMV case with $N = 5$. (**a**) $P_D - P_{FA}$; (**b**) NMSE (dB).

4.1.4. Interpretation of the results for the correlated case

For the correlated case, Figure 11a shows that C-SBL performed a little bit better than the two other compared algorithms in terms of support recovery, even though the C-SBL model does not account for the correlation that may exist across the columns of X. According to Figure 11b, MTCS, T-MSBL, and C-SBL have almost the same overall performance in terms of error.

4.2. Experiments on Real Data

In this section, we compare the performance of C-SBL against other algorithms on real data. Specifically, we consider the image reconstruction problem for the SMV case using the MNISTdataset. For the comparisons of the MMV case, the problem of blind sub-Nyquist sampling and reconstruction of multi-narrowband signals is considered.

Performance for the SMV Case (Experiments on MNIST Data)

Here, we evaluate the performance of the algorithms in reconstructing images of hand-written digits, using the well-known MNIST dataset [57]. MNIST consists of 70,000 gray-scale images of 28×28 hand-written digits. Experiments are conducted on a randomly-chosen set of hand-written digits from "0"–"9" from this dataset. Due to space considerations, we show only some of the results. The original images are upsampled to size 100×100 pixels, and the pixel values were normalized to be within $[0, 1]$. Then, the pixel values were subtracted from one, and those with a value of less than 0.3 were set to zero, similar to the binary pixel values in [58,59]. The threshold of 0.3 was obtained based on the average threshold of all the images using Otsu thresholding (the actual average threshold was 0.268). The corresponding matrix of pixel values of each image is then treated as the true sparse signal of interest, denoted by the true solution matrix X.

For the SMV case, we solve each column of X for each digit one at a time. The number of measurements for each column of X is set to 55 and $\mathbf{x}_n \in \mathbb{R}^{100}, \forall n = 1, \ldots, 100$, i.e., we consider a compression of 55% measurements for each vector $\mathbf{x}_n$. We randomly generated the sensing matrix A in the same way we described earlier. Then, each column of the matrix A is normalized to have a unit norm. The hand-written images of MNIST are already naturally sparse since most pixels in these images are inactive, i.e., they have a small number of non-zero pixels. The measurements for each column of the digits are computed by $\mathbf{y}_n = A\mathbf{x}_n + \mathbf{e}_n$ with SNR = 25 dB, where $\mathbf{e}$ is a Gaussian noise accounting for the measurement noise. This setting follows some of the other recent work in this area [60–64]. We feed all the algorithms with the measurement vector $\mathbf{y}$ and the same sensing matrix A. The generated measurement noise matrix is the same for the digits. Once $\hat{\mathbf{x}}_n, \forall n = 1, \ldots, 100$ is known, we collect the results and then stack them all together and reconstruct the digits. For the purpose of demonstrating the support recovery performance, in Figure 12, we illustrate how successful the algorithms were in finding the non-zero pixel locations. Since in the compared algorithms, except our proposed C-SBL, the models do not have the support-leaning vector $\mathbf{s}$, we performed the following.

In the reconstruction, we set the estimated pixel values less than 0.3 to zero, the same way as we treated the actual images. This means that we zeroed out the brightest pixels with the normalized value of lower than the threshold 0.3 and set the non-zero survival pixel values to one. However, since the proposed C-SBL algorithm already can provide the estimates on the active locations via $\mathbf{s}$, the thresholding process is not required. The first column of images in Figure 12 shows the true hand-written digits, and the other columns show the results of processing with other algorithms.

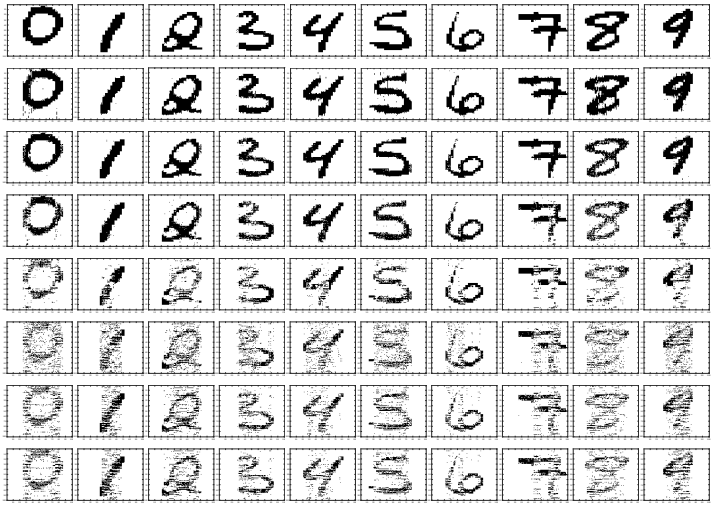


Figure 12. Results of reconstructed images for the SMV case. The first row illustrates the non-zero locations of the true hand-written digits. The other rows from top to bottom show the performance in terms of supports using the C-SBL, BSBL (h = 4), PCSBL, CLUSS-MCMC, MTCS, MFOCUSS, and BPDN algorithms, respectively.

We compare the performance of the C-SBL algorithm against other algorithms in the reconstruction of the images via both the support and pattern recovery. In Table 1, we evaluate the reconstruction based on the difference between the detection rate and false alarm rate ($P_D - P_{FA}$) in terms of support recovery. In Table 2, the performance is evaluated based on the success in pattern recovery. For this purpose, we stack all the columns of the true matrix X for each digit into a single column. We then construct the corresponding support vector, where the index of pixels with non-zero value will be replaced by "1" in the corresponding support vector. The true measure of clumpiness for each digit will then be computed via (6). We do the same procedure for computing the estimated measure of clumpiness in the reconstructed digits and provide the results in Table 2. In Table 3, the error between the true and the reconstructed images is represented.

Table 1. SMV case: comparing the reconstruction performance in terms of $P_D - P_{FA}$ for the digits. The bold face numbers show the best results in terms of support recovery.

Algorithm	Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
C-SBL	**0.9530**	**0.9954**	**0.9592**	**0.9643**	**0.9834**	**0.9690**	**0.9847**	**0.9890**	**0.9250**	**0.9794**
BSBL [15,23,53]	0.9204	0.9819	0.9341	0.8301	0.9617	0.9479	0.9116	0.9568	0.7469	0.9263
PCSBL [28,52]	0.7622	0.9544	0.8717	0.7281	0.8961	0.8270	0.8468	0.7787	0.5746	0.7046
CLUSS [37,54]	0.4265	0.6689	0.4803	0.4981	0.6421	0.5878	0.7179	0.6098	0.3123	0.4233
MTCS [34,51]	0.3030	0.4828	0.3380	0.4102	0.4779	0.4074	0.5989	0.5042	0.2740	0.3620
MFOCUSS [17,53]	0.3652	0.6510	0.4012	0.4378	0.5197	0.4961	0.5734	0.5054	0.2997	0.4324
BPDN [50]	0.3701	0.6360	0.4502	0.4161	0.5212	0.4967	0.5859	0.5370	0.3251	0.4052

Table 2. SMV case: comparing the performance in terms of learning the clustering pattern via the measure of clumpinesss ($\Sigma\Delta$) for the true and the reconstructed digits. The bold face numbers show the best results in terms of pattern recovery of the supports of the solution.

Algorithm	Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
True value	208	80	290	306	166	316	220	240	358	168
C-SBL	**244**	**78**	**266**	**284**	**176**	**324**	**196**	**241**	**340**	**172**
BSBL [15,23,53]	320	84	402	458	200	344	262	276	750	228
PCSBL [28,52]	634	126	528	636	346	632	344	532	1080	560
CLUSS [37,54]	1018	370	960	758	570	804	446	664	1152	708
MTCS [34,51]	1774	1022	1742	1200	1194	1496	760	1178	1646	1298
MFOCUSS [17,53]	1532	878	1508	1082	936	1220	684	1124	1346	1186
BPDN [50]	1478	818	1440	1054	908	1192	646	1014	1352	1172

Table 3. SMV case: comparing the performance in terms of the reconstruction error NMSE (dB) for the reconstructed digits. The bold face numbers show the best results in terms of reconstruction error.

Algorithm	Digit 0	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9
C-SBL	−8.0264	**−17.0256**	−9.2213	**−7.3822**	**−13.9387**	−10.1113	**−13.9837**	−9.6984	−3.7184	−8.3104
BSBL [15,23,53]	**−11.6553**	−16.4167	**−12.0378**	−7.2124	−13.5679	**−12.7994**	−11.3422	**−13.3647**	**−5.4846**	**−11.3947**
PCSBL [28,52]	−5.3168	−13.1082	−8.2997	−4.9542	−8.8138	−6.8723	−8.3111	−5.3168	−2.9960	−4.1409
CLUSS [37,54]	−1.5934	−4.1480	−2.1561	−2.6419	−4.1116	−3.3908	−6.1695	−3.1560	−0.9962	−1.3608
MTCS [34,51]	0.4987	−0.3990	0.1748	−0.4367	−0.9529	−0.1357	−2.6530	−0.8175	0.6790	0.4506
MFOCUSS [17,53]	−1.1638	−3.1979	−1.3832	−1.7897	−2.4664	−2.2302	−3.4462	−2.2340	−0.7605	−1.4593
BPDN [50]	−1.2096	−3.0473	−1.7805	−1.6362	−2.5115	−2.2342	−3.6447	−2.5400	−1.0469	−1.3351

According to the results shown in Table 1, the best results in support recovery belong to the C-SBL algorithm. Furthermore, Table 2 shows that C-SBL was more successful in learning the underlying clustering pattern of the digits. In terms of the reconstruction error, Table 3 shows that the C-SBL and BSBL algorithms compete with each other, with some results being better for C-SBL and some being better for BSBL.

4.3. Performance for the MMV Case (Experiments on Blind Multi-Narrowband Signal Sampling and Reconstruction

In this section, we consider the problem of blind sub-Nyquist sampling and reconstruction of multi-narrowband signals. The notion of blindness here means that the frequency support is unknown, and it occupies only a small portion of a wide spectrum [65]. In the original problem, it is assumed that the number of sparsely-scattered bands and their bandwidth are *known*, while the carrier locations are unknown at the receiver. The sub-Nyquist sampling in [65] is performed in the modulated wideband converter (MWC) stage, which multiplies the analog signal by a bank of M_{ch} periodic waveforms followed by low-pass filtering and then sampling the outputs uniformly at a low rate far less than the Nyquist rate. The periodic waveforms intentionally alias the spectrum such that a portion of each band appears in the baseband. The technique used here is referred to as Xampling for the compressive sensing of analog signals [24,65]. The MMV problem appears in the reconstruction stage of the Xampling framework, referred to as the continuous-to-finite (CTF) stage. In this stage, the low rate samples, obtained from the MWC stage, are fed to the CTF stage to estimate the support vector. The estimation problem here involves solving for the sparse solution of an underdetermined system of linear equations, which has the structure of the MMV problem. The active bands (spectrum slices) of the signal are reconstructed based on the estimated supports. For more detail, refer to [24,65–67]. In our example, the signal of interest is a multi-band signal containing two pairs of bands, i.e., $N_0 = 4$, where each band is of width $B = 50$ MHz, and it is assumed that the Nyquist rate of the multi-band signal is as high as $f_{Nyq} = 10$ GHz. The true carriers are set to $f_{c_1} = 2.4956$ MHz and $f_{c_2} = 4.4086$ MHz, which are assumed to be unknown for the simulation purposes. The number of channels are set to $M_{ch} = 70$, and the energies of the bands are set to $E_1 = 1$ and $E_2 = 2$, respectively. All the other settings of the signal model and the sampling parameters are defined the same as the implementation used in [68,69], which do not appear here due to the space consideration. In the simulations, the MMV problem that all algorithms need to solve is of the form $Y = AX$, where $P = 195$, $M = 70$, and $N = 8$. The actual sparsity level is $k = 8$, which here we assume to be unknown. Having no information on k here

means that we assume that the number of active bands in the signal is also unknown. In Figure 13, we illustrate an example of a multi-narrowband signal, its spectrum, and the signal when contaminated with noise, which is used for the simulation purposes. Furthermore, Figure 14 shows the comparison of the OMP, our proposed algorithm (C-SBL), MFOCUSS, MSBL, and MTCS in estimating the spectrum of the signal to reconstruct the signal. Notice that *the OMP algorithm is fed with the actual sparsity level* ($k = 8$), while this information is assumed *unknown* to the other algorithms. The results shown for the OMP serve as a baseline in our comparisons, but the OMP requires more information to solve the problem, so the comparison is not exactly fair.

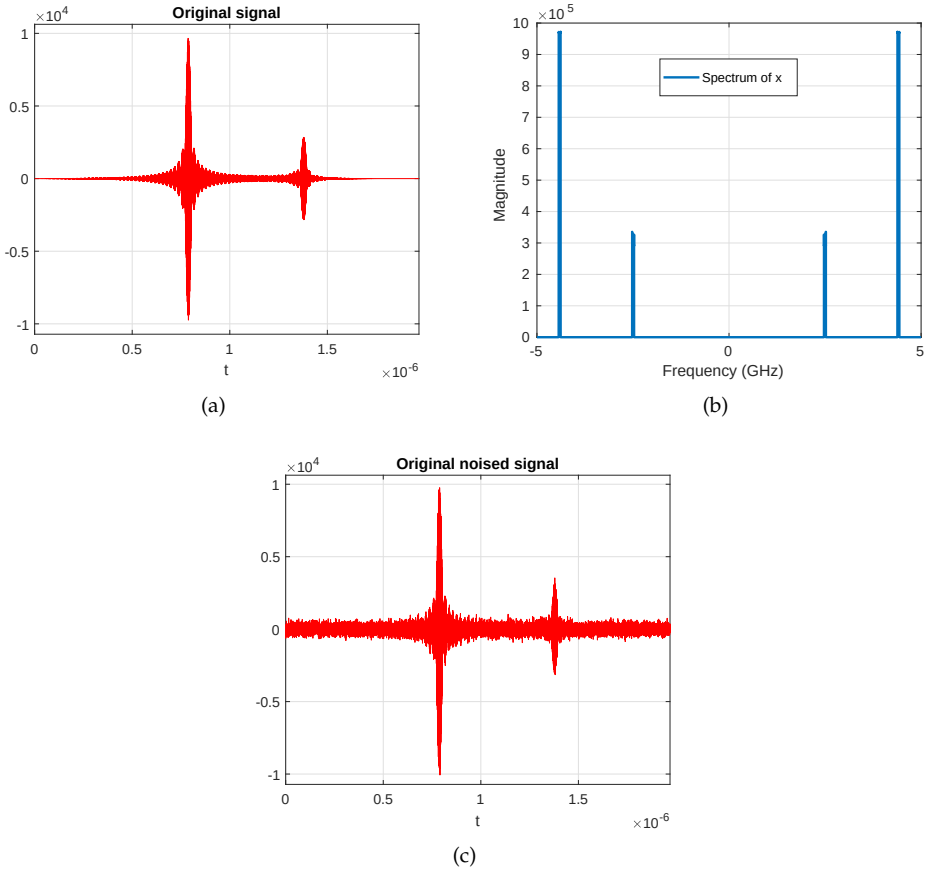


Figure 13. An example of a multi-narrowband signal for comparison purposes.(**a**) Original signal; (**b**) Spectrum of the signal; (**c**) Original noisy signal.

According to the results obtained in Figure 14, the C-SBL algorithm performed better than the alternatives in estimating the spectrum supports (closest support recovery to the OMP), which resulted in better signal reconstruction. Again, the OMP in these simulations is fed with the true sparsity level with the true support block sizes.

In Table 4, the reconstruction error in terms of NMSE (dB) is represented. According to the results, we see that the C-SBL algorithm outperformed the alternatives in reconstructing the signal.

Table 4. Comparing the reconstruction performance in terms of NMSE (dB) for the blind multi-narrowband signal. The two bold face numbers show the best results, which belong to OMP and C-SBL.

Algorithm	**OMP [5,49]**	**C-SBL**	**MTCS [34,51]**	**MFOCUSS [17,53]**	**MSBL [22,53]**
NMSE (dB)	**−17.5898**	**−14.2124**	−3.3095	−3.2028	−1.7223

The reason that the alternative algorithms did not provide low reconstruction error is due to the fact that their estimated spectrum supports of the signal became non-sparse, as illustrated in Figure 14.

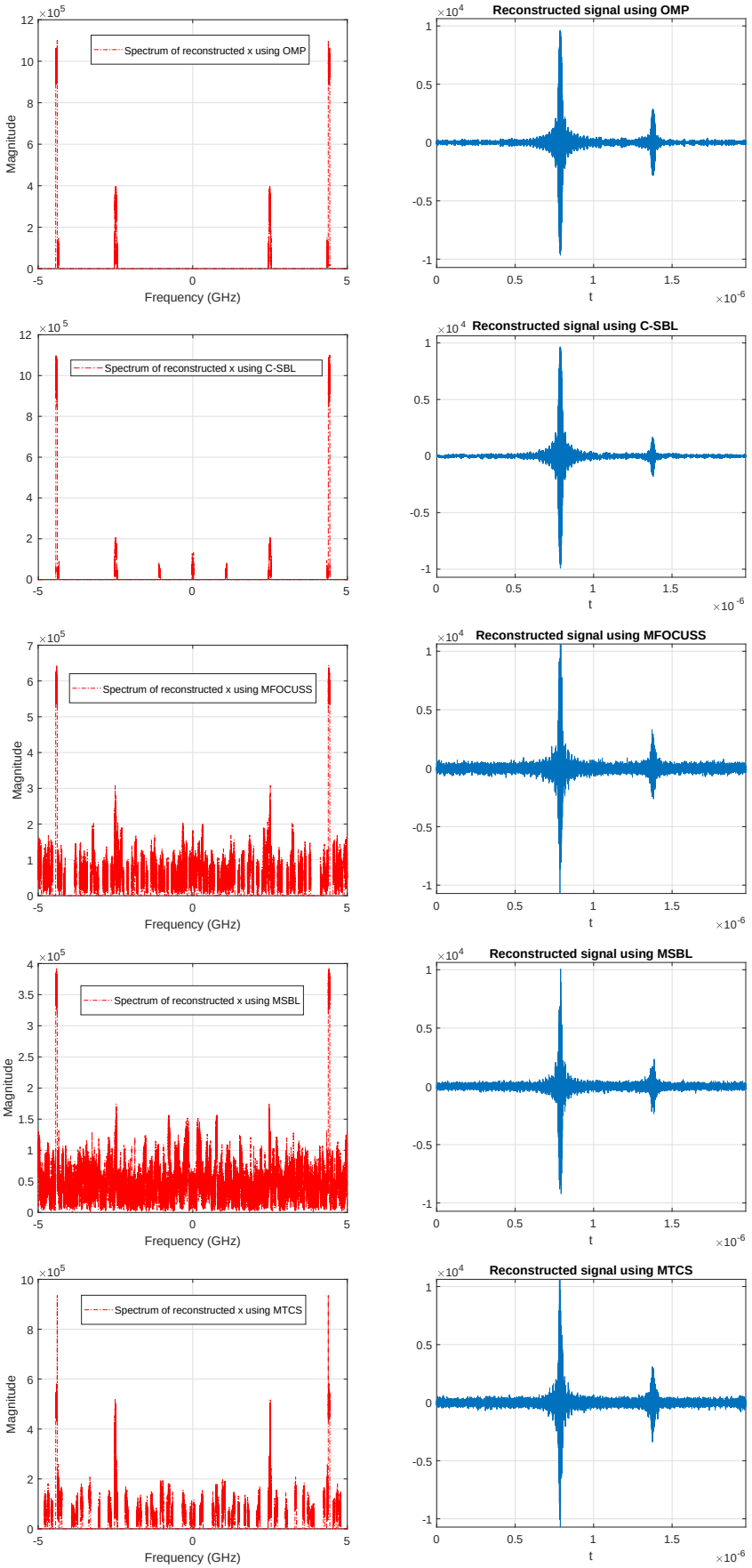


Figure 14. Results of estimated spectrum of the signal and the reconstructed signal in the CTF stage of Xampling when using the OMP, C-SBL, MFOCUSS, MSBL, and MTCS algorithms.

5. Convergence Diagnostics of the MCMC Implementation

Convergence is an important issue in order to determine the burn-in period of MCMC algorithms [37]. There is work on the convergence of iterative simulations and their inference using multiple sequences via potential scale reduction factor (PSRF) and multiple-PSRF (MPSRF) in [70,71]. For example, in [37], the convergence issue of the CLUSS-MCMC algorithm was resolved via studying the evolution of MPSRF in the collected samples for the sparse signal and its corresponding variances and the measure of PSRF for the noise variance. Following the same approach, in Figure 15, we provide some examples demonstrating the evolution of the PSRF for 20 independent chains of our proposed C-SBL algorithm. In these examples, we generated random clustered pattern sparse signals of length $N = 100$, a sparsity level of 25, and with the number of measurements $M = 20$, 50, and 90.

In Figure 15a, we demonstrate the PSRF and MPSRF for the variables and parameters of interest in our model for an example with the low sampling ratio of 0.2. According to the plot, the Gibbs sampler converges very quickly for ε, γ, and $\mathbf{x}$, i.e., the convergence measure of PSRF became close to one with few iterations. However, the convergence of the distribution on $\mathbf{s}$ is slower. The convergence of the precision on X was the slowest. According to Gelman's discussion in [72], one may prefer to set the burn-in period based on the PSRF close to 1.2. Based on this criterion, we can set the burn-in period to approximately 2000 iterations for the example made in Figure 15a. In Figure 15b, we provide another example for the case with the moderate sampling ratio of 0.5. As can be seen in this plot, the distributions of all the variables and parameters of interest converged faster than when the sampling ratio was 0.2. For this example, a burn-in period of around 600 is satisfactory. Figure 15c illustrates another example for the high sampling ratio of 0.9. In this plot, we observe that a burn-in period of around 200 suffices. Since in Figure 7 and Figures 9–11, we wanted to show the average of the overall performance of our algorithm, we first performed the following and then set the burn-in period based on the obtained experimental results. For each sampling ratio, we generated 100 random trials for solving the SMV and MMV problems. In these trials, we assessed the average PSRF measure for all the variables and parameters of interest based on Gelman's criterion in [72]. We monitored the average number of elapsed iterations until the variations on the outcomes of the estimated $\mathbf{s}$ became negligible for a fair number of iterations (this is easy to monitor since the outcome of the posterior on s_p is Bernoulli). This is equivalent to monitoring the trace plots, as suggested by Neal [72]. More specifically, we monitored the posterior distribution on the support learning vector $\mathbf{s}$, using (5) for O-SBL and (13) for C-SBL, based on the samples obtained from MCMC implementation. In Figure 16, we illustrate some examples of support learning vector $\mathbf{s}$ using the C-SBL algorithm with the number of measurements $M = 55$. Each plot shows the learning process of $\mathbf{s} \in \mathbb{R}^{100}$, represented by the samples drawn from (13), as a function of the number of iterations. Using the experimental results based on both the PSRF evaluation and monitoring the outcomes of $\mathbf{s}$, we then set fixed burn-in periods for the simulations illustrated in Section 4. In other words, since we wanted show the average performance, we preferred not to assess the PSRF of each simulated example, but rather using a fixed experimental burn-in period. Below, we provide the details on the burn-in and the collection periods of both the C-SBL and O-SBL algorithms for the simulation results illustrated in Section 4. In simulations on the synthetic data and the MNIST for the MMVs, we set the burn-in period to 500 followed by 500 iterations for the collection period. The same settings were used for the SMV case on the MNIST. In the experiments on synthetic data for the SMV, we set $N_{\text{burn-in}} = 2000$ followed by $N_{\text{collect}} = 1000$ iterations for the collection period for the sampling ratios of $M/P \leq 0.4$. For $M/P > 0.4$, we set $N_{\text{burn-in}} = 1000$ and $N_{\text{collect}} = 1000$. Thus, it might be the case that the burn-in period is required to be more than what we set. The effect of the need for a longer burn-in period can be observed in the results of Figures 7, 10, and 11 for low sampling ratios. The convergence diagnostic and the effect of burn-in period can also be detected in Figures 6 and 9. It should be clear from Figure 6c that for sampling ratios over 0.4, the average detection performance is almost independent of the threshold. The performance reveals that the approximated posterior distribution on $\mathbf{s}$ has already been stabilized. However, we see a different behavior for lower sampling ratios in Figure 6c. The posterior distribution on $\mathbf{s}$ is Bernoulli, and the variations on the supports in the iterative samples directly affect the performance in the support recovery. We see in Figure 6c that the average sample mean of the collected samples occurred around the threshold of 0.5. This can be interpreted as follows. The burn-in period may have been required to be larger than our setting, but the posterior distributions have been almost stabilized. Therefore, even for a lower burn-in period and sufficient iterations for the collection period, we could

still extract the information required for estimating the supports via computing the sample mean of the collected samples.

Computing the MSPRF for **s** needed some modifications. The estimated posterior variance of **s** is assessed based on the mixture of all the simulated sequences divided by the average of the variances within each sequence [72]. The main issue with the MPSRF for **s** occurred in our simulations when computing $\hat{R}$. In fact, this matrix became ill-conditioned, and the issue was with the fact that sequences on **s** were either zero or one. We dealt with this issue by adding random draws from a zero-mean Gaussian with the variance of 10^{-8} to the samples of **s** and then measured the MPSRF.

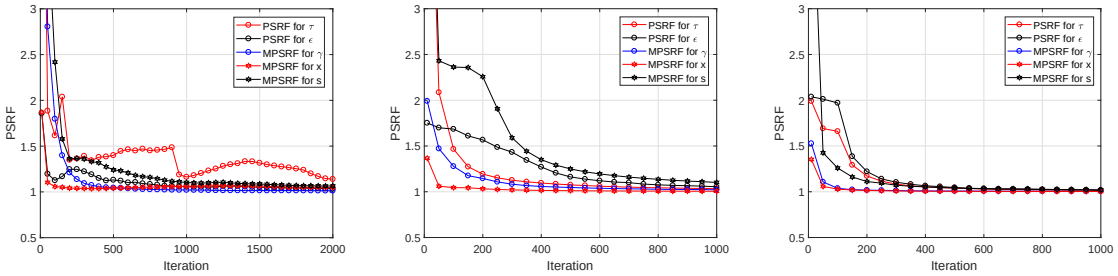


Figure 15. Examples showing the evolution of PSRF for the precision on the solution τ and the measurement noise precision ε and MPSRF for the solution vector **x**, the support vector **s**, and the mixing-coefficient vector γ for sampling ratios of 0.2, 0.5, and 0.9. (**a**) Example with $M = 20$; (**b**) Example with $M = 50$; (**c**) Example with $M = 90$.

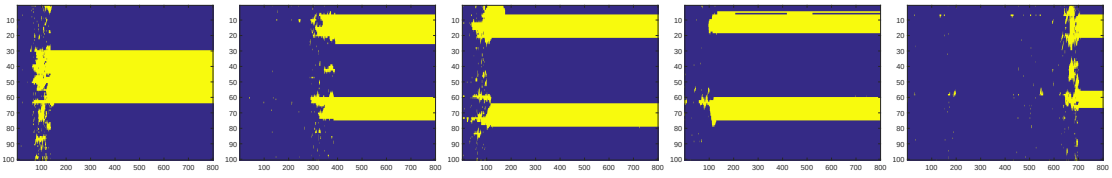


Figure 16. Examples showing samples of the support learning vector **s**. The vertical axis shows the elements of **s**, and the horizontal axis represents the iterations. (**a**) Example 1; (**b**) Example 2; (**c**) Example 3; (**d**) Example 4; (**e**) Example 5.

6. Conclusions

The O-SBL algorithm simultaneously learns both the supports and solution-value matrix for the MMVs with the joint sparsity structure. As the main contribution of this paper, the method was then extended to account for the case where the solution also exhibits an unknown clustered sparsity pattern. For this purpose, we introduced the C-SBL algorithm, which incorporates a total variation-based prior on the supports of the solution to learn the underlying clustered pattern. Based on simulations, we observed that C-SBL provides competitive performance for both the SMV and MMVs compared to the other algorithms.

Although C-SBL provides encouraging results, the MCMC implementation is computationally expensive. In future work, we will consider alternative approaches to MCMC implementation such as the variational Bayes inference technique [40,73].

Author Contributions: The contributions of the respective authors are as follows: conceptualization, M.S. and T.K.M.; methodology, M.S.; software, M.S. and T.K.M and J.H.G; validation, M.S.; formal analysis, M.S. and T.K.M. and J.H.G.; investigation, M.S. and T.K.M. and J.H.G.; resources, M.S. and T.K.M. and J.H.G.; data curation, M.S.; writing–original draft preparation, M.S.; writing–review and editing, T.K.M. and J.H.G. and M.S.; visualization, M.S. and T.K.M.; supervision, T.K.M.; project administration, T.K.M. and J.H.G.; funding acquisition, J.H.G. and T.K.M.

Funding: This work is supported by the Grant NASA NNX13 AD 39 A.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Below we describe the derivation of the inference equation for s_p provided in (5). According to the joint probability distribution (4), we have,

$$(s_p|-) \propto p(Y|s_p, X, \varepsilon)p(s_p|\gamma_p), \text{ and } (s_p|\gamma_p) \sim \text{Bernoulli}(\gamma_p), \ \forall p = 1, \ldots, P, \tag{A1}$$

and:

$$\begin{aligned} \log p(Y|\mathbf{s}, X, \varepsilon) \propto -\frac{\varepsilon}{2}\Big(&(y_{11} - \sum_{p=1}^{P} a_{1p}s_p x_{p1})^T(\star) + \cdots + \\ &(y_{M1} - \sum_{p=1}^{P} a_{Mp}s_p x_{x_{p1}})^T(\star) + \cdots + (y_{MN} - \sum_{p=1}^{P} a_{Mp}s_p x_{pN})^T(\star)\Big), \end{aligned} \tag{A2}$$

where "$\star$" in $(\kappa)^T(\star)$ denotes "κ". Define $\tilde{y}_{mn}^{-p} = y_{mn} - \sum_{l\neq p}^{P} a_{ml}s_l x_{ln}, \forall\{m, n, p\}$, and substitute into (A2) to obtain:

$$\begin{aligned} \log p(Y|s_p, X, \varepsilon) &\propto -\frac{\varepsilon}{2}\big(s_p^2 \left\|\mathbf{a}_p\right\|_2^2 \sum_{n=1}^{N} x_{pn}^2 - 2s_p(x_{p1}\sum_{m=1}^{M} a_{mp}\tilde{y}_{m1}^{-p} + \cdots + x_{pN}\sum_{m=1}^{M} a_{mp}\tilde{y}_{mN}^{-p})\big) \\ &\propto -\frac{\varepsilon}{2}\big(s_p^2 \left\|\mathbf{a}_p\right\|_2^2 \sum_{n=1}^{N} x_{pn}^2 - 2s_p(\mathbf{a}_p^T \sum_{n=1}^{N} x_{pn}\tilde{\mathbf{y}}_n^{-p})\big), \end{aligned} \tag{A3}$$

where $\tilde{\mathbf{y}}_n^{-p} := [\tilde{y}_{1n}^{-p}, \ldots, \tilde{y}_{Mn}^{-p}]^T$. Using (A1) and (A3), we have:

$$(s_p|-) \propto \gamma_p^{s_p}(1-\gamma_p)^{s_p} e^{-\frac{\varepsilon}{2}(s_p^2\left\|\mathbf{a}_p\right\|_2^2 \sum_{n=1}^{N} x_{pn}^2 - 2s_p\mathbf{a}_p^T \sum_{n=1}^{N} x_{pn}\tilde{\mathbf{y}}_n^{-p})}.$$

Since $s_p \in \{0, 1\}$, the marginalized posterior inference on this parameter can be simplified into (5).

References

1. Candes, E.J.; Romberg, J.; Tao, T. Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information. *IEEE Trans. Inf. Theory* **2006**, *52*, 489–509. [CrossRef]
2. Donoho, D.L. Compressed Sensing. *IEEE Trans. Inf. Theory* **2006**, *52*, 1289–1306. [CrossRef]
3. Li, K.; Gan, L.; Ling, C. Convolutional compressed sensing using deterministic sequences. *IEEE Trans. Signal Process.* **2013**, *61*, 740–752. [CrossRef]
4. Candes, E.J.; Wakin, M.B. An Introduction to Compressive Sampling. *IEEE Signal Process. Mag.* **2008**, *25*, 21–30. [CrossRef]
5. Elad, M. *Sparse and Redundant Representation: From Theory to Applications in Signal and Image Processing*; Springer: New York, NY, USA, 2010.
6. Baraniuk, R.G. Compressive Sensing. *IEEE Signal Process. Mag.* **2007**, *24*, 118–124. [CrossRef]
7. Mishali, M.; Eldar, Y.C. Reduce and Boost: Recovering Arbitrary Sets of Jointly Sparse Vectors. *IEEE Trans. Signal Process.* **2008**, *56*, 4692–4702. [CrossRef]
8. Chen, S.; Donoho, D. Basis Pursuit. In Proceedings of the 28th Asilomar Conference of Signals, Systems, and Computers, Pacific Grove, CA, USA, 31 October–2 November 1994; pp. 41–44.
9. Gill, P.R.; Wang, A.; Molnar, A. The in-Crowd Algorithm for Fast Basis Pursuit Denoising. *IEEE Trans. Signal Process.* **2011**, *59*, 4595–4605. [CrossRef]
10. Beck, A.; Teboulle, M. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM J. Imaging Sci.* **2009**, *2*, 183–202. [CrossRef]
11. Becker, S.; Bobin, J.; Candes, E.J. NESTA: A Fast and Accurate First Order Method for Sparse Recovery. *SIAM J. Imaging Sci.* **2009**, *4*, 1–39. [CrossRef]
12. Zhang, J.; Ghanem, B. ISTA-NET: Iterative Shrinkage-Thresholding Algorithm Inspired Deep Network for Image Compressive Sensing. *arXiv* **2017**, arXiv: 1706.07929.
13. Wipf, D.P.; Rao, B.D. Sparse Bayesian learning for basis pursuit Selection. *IEEE Trans. Signal Process.* **2004**, *52*, 2153–2164. [CrossRef]

14. Ji, S.; Xue, Y.; Carin, L. Bayesian Compressive Sensing. *IEEE Trans. Signal Process.* **2008**, *56*, 2346–2356. [CrossRef]
15. Zhang, Z.; Rao, B.D. Recovery of Block Sparse Signals Using the Framework of Block Sparse Bayesian Learning. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 3345–3348.
16. Fang, J.; Shen, Y.; Li, F.; Li, H.; Chen, Z. Support Knowledge-Aided Sparse Bayesian Learning for Compressed Sensing. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, 19–24 April 2015; pp. 3786–3790.
17. Cotter, S.F.; Rao, B.D.; Engan, K.; Delgado, K.K. Sparse Solutions to Linear Inverse Problem with Multiple Measurement Vectors. *IEEE Trans. Signal Process.* **2005**, *53*, 2477–2488. [CrossRef]
18. Chen, J.; Huo, X. Theoretical Results on Sparse Representaions of Multiple-Measurement Vectors. *IEEE Trans. Signal Process.* **2006**, *54*, 4634–4643. [CrossRef]
19. Ding, J.; Chen, L.; Gu, Y. Robustness of Orthogonal Matching Pursuit for Multiple Measurement Vectors in Noisy Scenario. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 3813–3816.
20. Hyder, M.M.; Mahata, K. A Robust Algorithm for Joint-Sparse Recovery. *IEEE Signal Process. Lett.* **2009**, *16*, 1091–1094. [CrossRef]
21. Baron, D.; Sarvotham, S.; Baraniuk, R.G. Bayesian Compressive Sensing via Belief Propagation. *IEEE Trans. Signal Process.* **2010**, *58*, 269–280. [CrossRef]
22. Wipf, D.P.; Rao, B.D. An Empirical Bayesian Strategy for Solving the Simultaneous Sparse Approximation Problem. *IEEE Trans. Signal Process.* **2007**, *55*, 3704–3716. [CrossRef]
23. Zhang, Z.; Rao, B.D. Extension of SBL Algorithm for the Recovery of Block Sparse Signals With Intra-Block Correlation. *IEEE Trans. Signal Process.* **2013**, *61*, 2009–2015. [CrossRef]
24. Mishali, M.; Eldar, Y.C. Xampling: Signal Acquisition and Processing in Union of Subspaces. *IEEE Trans. Signal Process.* **2011**, *59*, 4719–4734. [CrossRef]
25. Kwon, H.; Rao, B.D. On the Benefits Of The block-Sparsity Structure In Sparse Signal Recovery. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 3685–3688.
26. Tibshirani, R.; Saunders, M.; Rosset, S.; Zhu, J.; Knight, K. Sparsity and Smoothness via the Fused Lasso. *J. R. Stat. Soc. Ser. B* **2005**, *67*, 91–108. [CrossRef]
27. Hernandez-Lobato, D.; Hernandez-Lobato, J.M.; Dupont, P. Generalized Spike-and-Slab Priors for Bayesian Group Feature Selection Using Expectation Propagation. *J. Mach. Learn. Res.* **2013**, *14*, 1891–1945.
28. Fang, J.; Shen, Y.; Li, H.; Wang, P. Pattern-Coupled Sparse Bayesian Learning for Recovery of Block-Sparse Signals. *IEEE Trans. Signal Process.* **2015**, *63*, 360–372. [CrossRef]
29. Luo, J.A.; Zhang, X.P.; Wang, Z. Direction-of-Arrival Estimation Using Sparse Variable Projection Optimization. In Proceedings of the 2012 IEEE International Symposium on Circuits and Systems, Seoul, Korea, 20–23 May 2012; pp. 3106–3109.
30. Eldar, Y.C.; Kuppinger, P.; Bolcskei, H. Block-Sparse Signals: Uncertainty Relations and Efficient Recovery. *IEEE Trans. Signal Process.* **2010**, *58*, 3042–3054. [CrossRef]
31. Huang, J.; Zhang, T.; Metaxas, D. Learning with Structured Sparsity. *J. Mach. Learn. Res.* **2011**, *12*, 3371–3412.
32. Yuan, M.; Lin, Y. Model Selection and Estimation in Regression with Grouped Variables. *J. Mach. Learn. Soc. Ser. B* **2006**, *68*, 49–67. [CrossRef]
33. Qi, Y.; Liu, D.; Carin, L.; Dunson, D. Multi-Task Compressive Sensing with Dirichlet Process Priors. In Proceedings of the 25th international conference on Machine learning (ICML '08), Helsinki, Finland, 5–9 July 2008; pp. 768–775.
34. Ji, S.; Dunson, D.; Carin, L. Multitask Compressive Sensing. *IEEE Trans. Signal Process.* **2009**, *57*, 92–106. [CrossRef]
35. Al-Shoukairi, M.; Rao, B.D. Sparse Bayesian Learning Using Approximate Message Passing. In Proceedings of the 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 2–5 November 2014; pp. 1957–1961.
36. Shekaramiz, M.; Moon, T.K.; Gunther, J.H. On the Block-Sparse Solution of Single Measurement Vectors. In Proceedings of the 49th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 8–11 November 2015; pp. 508–512.

37. Yu, L.; Sun, H.; Barbot, J.P.; Zheng, G. Bayesian Compressive Sensing for Cluster Structured Sparse Signals. *Signal Process.* **2012**, *92*, 259–269. [CrossRef]
38. Anderson, M.R.; Winther, O.; Hansen, L.K. Bayesian Inference for Structured Spike and Slab Priors. *Adv. Neural Inf. Process. Syst.* **2014**, 1745–1753.
39. Meng, X.; Wu, S.; Kuang, L.; Huang, D.; Lu, J. AMP-NNSPL. *arXiv* **2016**, arXiv:1601.00543
40. Yu, L.; Wei, C.; Jia, J.; Sun, H. Compressive Sensing for Cluster Structured Sparse Signals: Variational Bayes Approach. *IET Signal Process.* **2016**, *10*, 770–779. [CrossRef]
41. Ding, X.; He, L.; Carin, L. Bayesian Robust Principal Component Analysis. *IEEE Trans. Image Proc.* **2011**, *20*, 3419–3430. [CrossRef] [PubMed]
42. Ziniel, J.; Schniter, P. Efficient High-Dimensional Inference in the Multiple Measurement Vector Problem. *IEEE Signal Process. Mag.* **2013**, *61*, 340–354. [CrossRef]
43. Cohen, D.; Mishra, K.V.; Eldar, Y.C. Spectrum Sharing Radar: Coexistence via Xampling. *IEEE Trans. Aerosp. Electron. Syst.* **2018**, *54*, 1279–1296. [CrossRef]
44. Fang, J.; Zhang, L.; Li, H. Two-Dimensional Pattern-Coupled Sparse Bayesian Learning via Generalized Approximate Message Passing. *IEEE Trans. Image Proc.* **2016**, *25*, 2920–2930. [CrossRef] [PubMed]
45. Vila, J.P.; Schniter, P. Expectation-Maximization Gaussian-Mixture Approximate Message Passing. *IEEE Trans. Signal Proc.* **2013**, *61*, 4658–4672. [CrossRef]
46. Shekaramiz, M.; Moon, T.K.; Gunther, J.H. Hierarchical Bayesian Approach for Jointly-Sparse Solution of Multiple-Measurement Vectors. In Proceedings of the 48th Asilomar Conference of Signals, Systems, and Computers, Pacific Grove, CA, USA, 2–5 November 2014; pp. 1962–1966.
47. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2009.
48. Grant, C.S.; Moon, T.K.; Gunther, J.H.; Stites, M.R.; Williams, G.P. Detection of Amorphously Shaped Objects Using Spatial Information Detection Enhancement (SIDE). *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2012**, *5*, 478–487. [CrossRef]
49. Pati, Y.C.; Rezaifar, R.; Krishnaprasad, P.S. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition. In Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 1–3 November 1993; pp. 40–44.
50. Friedlander, M.P.; Vanderberg, E. SPGL1 (version 1.9): A Solver for Large-Scale Sparse Reconstruction. Available online: https://www.cs.ubc.ca/~mpf/spgl1/ (accessed on 15 September 2018).
51. Ji, S. MT-CS. Available online: http://people.ee.duke.edu/~lcarin/BCS.html (accessed on 21 August 2018).
52. Fang, J.; Shen, Y. PC-SBL. Available online: http://junfang-uestc.net/swf/publication.html (accessed on 21 August 2018).
53. Zhang, Z. Materials: Matlab-Codes. Available online: https://sites.google.com/site/researchbyzhang/software (accessed on 20 August 2018).
54. Yu, L. Materials: Matlab-Codes. Available online: https://sites.google.com/site/link2yulei/publications/materials (accessed on 20 August 2018).
55. Zhang, Z.; Rao, B.D. Sparse Signal Recovery with Temporally Correlated Source Vectors Using Sparse Bayesian Learning. *IEEE J. Sel. Top. Signal Proc.* **2011**, *5*, 912–926. [CrossRef]
56. Zhang, Z.; Rao, B.D. Clarify Some Issues on the Sparse Bayesian Learning for Sparse Signal Recovery. Available online: http://dsp.ucsd.edu/~zhilin/papers/clarify.pdf (accessed on 20 September 2018).
57. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Documnet Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
58. Bora, A.; Jalal, A.; Price, E.; Dimakis, A.G. Compressed Sensing Using Generative Models. *arXiv* **2017**, arXiv: 1703.03208.
59. Tramel, E.W.; Dremeau, A.; Krzakala, F. Approximate Message Passing with Restricted Boltzmann Machine Priors. *J. Stat. Mech: Theory Exp.* **2016**, *2016*, 073401. [CrossRef]
60. Mousavi, H.S.; Monga, V.; Tran, T.D. Iterative Convex Refinement for Sparse Recovery. *IEEE Signal Process. Lett.* **2015**, *22*, 1903–1907. [CrossRef]

61. Shekaramiz, M.; Moon, T.K.; Gunther, J.H. AMP-B-SBL: An Algorithm for Clustered Sparse Signals Using Approximate Message Passing. In Proceedings of the 2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 20–22 October 2016; pp. 1–5.
62. Palangi, H.; Ward, R.; Deng, L. Distributed Compressive Sensing: A Deep Learning Approach. *IEEE Trans. Signal Process.* **2016**, *64*, 4504–4518. [CrossRef]
63. Chang, J.H.; Li, C.L.; Poczos, B.; Kumar, B.V.V.; Sankaranarayanan, A.C. One Network to Solve Them ALL-Solving Linear Inverse Problems Using Deep Projection Models. *arXive* **2017**, arXiv:1703.09912.
64. Vu, T.H.; Mousavi, H.S.; Monga, V. Adaptive Matching Pursuit for Sparse Signal Recovery. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 4331–4335.
65. Mishali, M.; Eldar, Y.C. From Theory to Practice: Sub-Nyquist Sampling of Sparse Wideband Analog Signals. *IEEE J. Sel. Top. Signal Process.* **2010**, *4*, 375–391. [CrossRef]
66. Mishali, M.; Eldar, Y.C. Blind Multiband Signal Reconstruction : Compressed Sensing for Analog Signals. *IEEE Trans. Signal Process.* **2009**, *57*, 993–1009. [CrossRef]
67. Mishali, M.; Eldar, Y.C.; Tropp, J.A. Efficient Sampling of Sparse Wideband Sparse Analog Signals. In Proceedings of the 2008 IEEE 25th Convention of Electrical and Electronics Engineers in Israel, Eilat, Israel, 3–5 December 2008; pp. 290–294.
68. Eldar, Y.C. MWC Matlab Package. Available online: http://webee.technion.ac.il/Sites/People/YoninaEldar/software_det2.php (accessed on 25 August 2018).
69. Siguoyi. MWC matlab package. Available online: https://github.com/siguoyi/MWC/find/master (accessed on 25 August 2018).
70. Gelman, A.; Rubin, D.B. Inference from Iterative Simulation Using Multiple Sequences. *Stat. Sci.* **1992**, *7*, 457–511. [CrossRef]
71. Brooks, S.P.; Gelman, A. General methods for Monitoring Convergence of Iterative Simulations. *J. Comput. Graph. Stat.* **1998**, *7*, 434–455.
72. Kass, R.E.; Carlin, B.P.; Gelman, A.; Neal, R.M. Markov Chain Monte Carlo in Practice: A Roundtable Discussion. *Am. Stat. Assoc.* **1998**, *52*, 93–100.
73. Beal, M.J. *Variational Algorithms for Approximate Bayesian Inference*; University of London: London, UK, 2003.