SPARSE SIGNAL RECOVERY BASED ON COMPRESSIVE SENSING AND EXPLORATION USING MULTIPLE MOBILE SENSORS

by

Mohammad Shekaramiz

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

Todd K. Moon, Ph.D. Major Professor	Scott E. Budge, Ph.D. Committee Member
Mark E. Fels, Ph.D. Committee Member	Jacob H. Gunther, Ph.D. Committee Member
Charles M. Swenson, Ph.D. Committee Member	Laurens H. Smith, Ph.D. Interim Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2018

ABSTRACT

Sparse Signal Recovery Based on Compressive Sensing

and Exploration Using Multiple Mobile Sensors

by

Mohammad Shekaramiz, Doctor of Philosophy

Utah State University, 2018

Major Professor: Todd K. Moon, Ph.D.
Department: Electrical and Computer Engineering

The objective of this dissertation is twofold. First, we study the reconstruction problem of sparse signals using compressive sensing (CS) technique, where the signal may exhibit unknown clustered pattern. CS provides new techniques, in a sub-Nyquist sampling sense, for signal acquisition and reconstruction. It has been found in a variety of applications such as single-pixel camera, missing pixels image recovery, medical imaging and ranging (MRI), communications, and many more. The objective in CS is to efficiently capture the important information of the underlying signal via a small number of measurements. Then, the goal becomes sparse signal recovery from such measurements. The general assumption here is that such signal is compressible or sparse but the number and location of dominating non-zeros are unknown. The focus of this dissertation is on the reconstruction stage of CS. Here, several new Bayesian modeling and algorithms for solving the inverse problem of CS are proposed. These algorithms are implemented using techniques such as Markov chain Monte Carlo (MCMC), message passing, and variational Bayesian (VB) inference. The results will be shown to outperform other state-of-the-art algorithms for most cases.

The second direction of this dissertation is on the exploration problem using multiple mobile sensors. We examine the configuration of multiple mobile sensors to explore an unknown region. The exploration problem here trades off between two desiderata: to continue taking data in a region known to be interesting with the intent of refining the measurements *vs.* taking data in unobserved areas to attempt to discover new interesting regions. Making reasonable and practical decisions to simultaneously fulfill both goals is hard and contradictory. For this problem, a new framework is proposed. The framework employs a Gaussian process regression model to predict the phenomenon at unseen locations. Then, the decision-making on the trajectories of sensors is performed via an optimization problem with a tuning parameter that balances between the two described desires. Furthermore, the decision-maker stage will be improved using an epistemic utility controller. In epistemic utility theory, as a decision-making framework, decisions are made via emphasis on avoiding wrong decisions and favoring informationally valuable decisions rather than the more restrictive (and sometimes inachievable) task of finding the best solution. As an application, we investigate a surrogate example for the exploration problem using a constellation of satellites. Here, the goal is tuning or changing the orbital planes of the constellation to find and track the most interesting features of the phenomenon. Although the proposed problem is applied to the placement of sensor-bearing satellites in this dissertation, the method is equally applicable to sensors carried by mobile robots and so on.

(236 pages)

PUBLIC ABSTRACT

Sparse Signal Recovery Based on Compressive Sensing

and Exploration Using Multiple Mobile Sensors

Mohammad Shekaramiz

The work in this dissertation is focused on two areas within the general discipline of statistical signal processing. First, several new algorithms are developed and exhaustively tested for solving the inverse problem of compressive sensing (CS). CS is a recently developed sub-sampling technique for signal acquisition and reconstruction which is more efficient than the traditional Nyquist sampling method. It provides the possibility of compressed data acquisition approaches to directly acquire just the important information of the signal of interest. Many natural signals are sparse or compressible in some domain such as pixel domain of images, time, frequency and so forth. The notion of compressibility or sparsity here means that many coefficients of the signal of interest are either zero or of low amplitude, in some domain, whereas some are dominating coefficients. Therefore, we may not need to take many direct or indirect samples from the signal or phenomenon to be able to capture the important information of the signal. As a simple example, one can think of a system of linear equations with N unknowns. Traditional methods suggest solving N linearly independent equations to solve for the unknowns. However, if many of the variables are known to be zero or of low amplitude, then intuitively speaking, there will be no need to have N equations. Unfortunately, in many real-world problems, the number of non-zero (effective) variables are unknown. In these cases, CS is capable of solving for the unknowns in an efficient way. In other words, it enables us to collect the important information of the sparse signal with low number of measurements. Then, considering the fact that the signal is sparse, extracting the important information of the signal is the challenge that needs to be addressed. Since most of the existing recovery algorithms in this area need some prior knowledge or parameter tuning, their application to real-world problems to achieve a good performance is difficult. In this dissertation, several new CS algorithms are proposed for the recovery of sparse signals. The proposed algorithms mostly do not require any prior knowledge on the signal or its structure. In fact, these algorithms can learn the underlying structure of the signal based on the collected measurements and successfully reconstruct the signal, with high probability. The other merit of the proposed algorithms is that they are generally flexible in incorporating any prior knowledge on the noise, sparisty level, and so on.

The second part of this study is devoted to deployment of mobile sensors in circumstances that the number of sensors to sample the entire region is inadequate. Therefore, where to deploy the sensors, to both explore new regions while refining knowledge in aleady visited areas is of high importance. Here, a new framework is proposed to decide on the trajectories of sensors as they collect the measurements. The proposed framework has two main stages. The first stage performs interpolation/extrapolation to estimate the phenomenon of interest at unseen loactions, and the second stage decides on the informative trajectory based on the collected and estimated data. This framework can be applied to various problems such as tuning the constellation of sensor-bearing satellites, robotics, or any type of adaptive sensor placement/configuration problem. Depending on the problem, some modifications on the constraints in the framework may be needed. As an application side of this work, the proposed framework is applied to a surrogate problem related to the constellation adjustment of sensor-bearing satellites.

To my parents
Mostafa and Tooran

To my siblings
Elham and Mehdi

ACKNOWLEDGMENTS

There are many people whom I want to mention their names here. Those who have helped me in numerous ways not only on my path through the completion of my Ph. D. but also in the other aspects of my life. First, and foremost, I want to thank my supervisor, Dr. Todd K. Moon, for giving me valuable guidance throughout the course of my study. His deep insights and positive manner have always been helpful and encouraging. He has been very patient, great encourager and supporter, and sometimes though as it should be. Next, special thanks go to my committee members, Dr. Gunther, Dr. Budge, Dr. Swenson, and Dr. Fels for their support and help, particularly for their patience in reading my dissertation draft. Using this space, I should specially thank Dr. Gunther for his consistent support and encouragement, as well. Also, special thanks to Dr. Cripps. I really enjoyed being a teaching assistant for two of his classes.

I also would like to express my gratitude to the Utah State University, as I enjoyed studying over there and I learned a lot.

In addition, I want to thank Dr. Hadi Malek, Mehdi Maher, Dr. Leila Esfahani, Dr. Leila Ahmadi, Dr. Soodeh Dadras, Mahyar Aboutalebi, Irene Garousi Nejad, Masoud Oskoui, Manijeh Nourayi, Banafsheh Nourayi, Mr. Nourayi, Amir Hossein Farzaneh, Amir Behbahanian, Sattar Dorafshan, Vahid Kouhdaragh, Afshin Abadi, Amir Manzourolajdad, Mohsen Jamal, Aboulfazl Javan, Reza Safa, Amir Mirzayinia and his wife Soraya, David Neal, Andrew Pound, Mehedi Hassan, Ali Al-Hashimi, Waled Al-Dulaimi, and many more for their friendship and encouragement.

Also, I want to thank Tricia Brandenburg, Heidi Harper, Kathy Phippen, and Diane Buist from the ECE department.

Last, but not least, I would like to take this space to express my gratitude to my mom, dad, and siblings for their consistent help, support, and in particular, moral support in my journey including my academic career and other aspects of my life.

Mohammad Shekaramiz

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

AMP	Approximate message passing
BCS	Bayesian compressive sensing
BG	Bernoulli-Gaussian
BGiG	Bernoulli Gaussian-inverse Gamma
BP	Basis pursuit
BPDN	Basis pursuit denoising
CS	Compressive sensing
C-SBL	Clustered-SBL
DCT	Discrete cosine transform
DOA	Direction of arrival
EM	Expectation-maximization
GiG	Gaussian-inverse Gamma
GP	Gaussian process
GPR	Gaussian process regression
HSI	Hyper-spectral imaging
LS	Least squares
MAP	Maximum a *posteriori*
MCMC	Markov chain Monte Carlo
MEG	Magnetoencephalography
ML	Maximim likelihood
MMV	Multiple measurement vector
MP	Matching pursuit
MPSRF	Multiple PSRF
MRI	Magnetic resonance imaging
NMSE	Normalized mean-squared error
NP-HARD	Non-deterministic polynomial-time hard
OMP	Orthogonal matching pursuit
O-SBL	Ordinary-SBL
PoI	Phenomenon of interest
PSNR	Peak-signal-to-noise ratio
PSRF	Potential reduced scale factor
RIC	Restricted isometry constant
RIP	Restricted isometry property
ROC	Receiver operating curve
SBL	Sparse Bayesian learning
SMV	Single measurement vector
TEC	Total electron content
VB	Variational Bayes
XAMPLING	Compressive sensing sampling

CHAPTER 1

INTRODUCTION

This dissertation is categorized into two research topics. In the first research direction, the reconstruction problem of sparse or compressible signals using compressive sensing (CS) technique is investigated. Compressibility or sparsity here means that many coefficients of the signal of interest are either zero or of low amplitude, in some domain, whereas some are dominating coefficients. A vector $\mathbf{x}_s \in \mathbb{R}^N$ is K-sparse, if at most $K \ll N$ elements of this vector are non-zero. Examples of sparsifying domain for natural signals or images can be discrete cosine transform (DCT), wavelet, Fourier, or time domain.

CS provides new sub-Nyquist sampling techniques for signal acquisition and reconstruction. It has been used in a variety of applications such as single-pixel camera, super-resolution imaging, missing pixels image recovery, medical imaging, face recognition, sparse channel estimation, blind multi-narrowband signals, audio signals, spectrum sharing RADAR, separating foreground and background in a video recording [1–16]. For example, in [1], the reconstruction problem of blind multi-narrowband signals from its sub-Nyquist pointwise samples was considered under the case where the band locations are unknown. The proposed approach in [1] was named Xampling i.e., compressive sensing of analog signals. They showed that when having N narrowband signals of maximum bandwidth of B, the sampling rate of $4NB$ suffices for the signal recovery, which in most cases is far less than the Nyquist rate. Figure Fig. 1.1 illustrates an example of multi-narrowband singals [1].

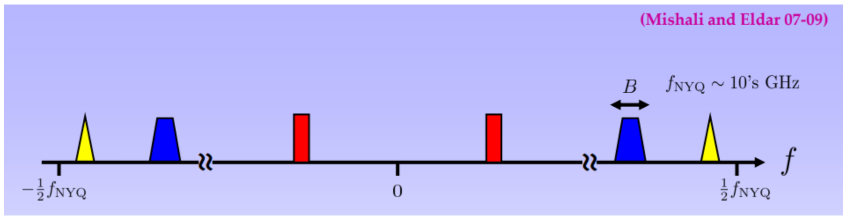

(Mishali and Eldar 07-09)
B
$f_{\mathrm{NYQ}} \sim$ 10's GHz
f
$-\frac{1}{2}f_{\mathrm{NYQ}}$
0
$\frac{1}{2}f_{\mathrm{NYQ}}$

Fig. 1.1: An example of a multi-narrowband signal [1].

The objective of using CS techniques is to efficiently capture the important information of the underlying signal from a small number of linear measurements. The main feature of using CS is as follows. Since it is assumed that the signal of interest is sparse, it may not be required to take many direct or indirect samples from the signal to be able to capture the important information of the signal. Mathematically speaking, the signal acquisition in CS is performed by constructing a sensing matrix, corresponding to the sensing device, which should be carefully designed. More specifically, this matrix should meet the restricted isometry property (RIP). Details on the RIP property can be found in [17]. In CS, the signal of interest $\mathbf{x} \in \mathbb{R}^N$ is assumed K-sparse in some basis Φ, i.e., $\mathbf{x} = \Phi \mathbf{x}_s$. The relationship between the measurements and the underlying sparse signal is then modeled as $\mathbf{y} = \Psi \mathbf{x}$ or equivalently, $\mathbf{y} = \Psi \Phi \mathbf{x}_s$, where $K < M \ll N$ and $M = O(K \log \frac{N}{K})$. Examples of Ψ can be matrices with elements generated randomly from Gaussian or Bernoulli distributions. The CS model can be also represented as $\mathbf{y} = A\mathbf{x}_s$, where $A_{M\times N} := \Psi\Phi$.

Once the measurements are collected, the goal becomes reconstructing the signal under the assumption that such signal is compressible or sparse but the number and location of dominating non-zeros are unknown. In other words, the objective is then to reconstruct $\mathbf{x}_s$ from $\mathbf{y}$ and A, which yields to solving for $\hat{\mathbf{x}}_s$ in the following underdetermined system of linear equations

$$(\text{P0}): \min_{\hat{\mathbf{x}}_s \in \mathbb{R}^N} \|\mathbf{x}_s\|_0 \text{ s.t. } \mathbf{y} = A\mathbf{x}_s. \tag{1.1}$$

Unfortunately, the optimization problem (P0) in (1.1) is non-convex resulting in a combinatorial search, which is an NP-hard problem. The common approach then becomes replacing l_1-norm instead of l_0-norm in (1.1). It has been shown that if the signal is sparse enough, then the l_1-minimization problem is a good representation of l_0-minimization problem in (1.1). The CS problem using l_1-minimization is defined as

$$(\text{P1}): \min_{\hat{\mathbf{x}}_s \in \mathbb{R}^N} \|\mathbf{x}_s\|_1 \text{ s.t. } \mathbf{y} = A\mathbf{x}_s, \text{ where } \|\mathbf{x}_s\|_1 := \sum_{n=1}^{N} |x_n|, \tag{1.2}$$

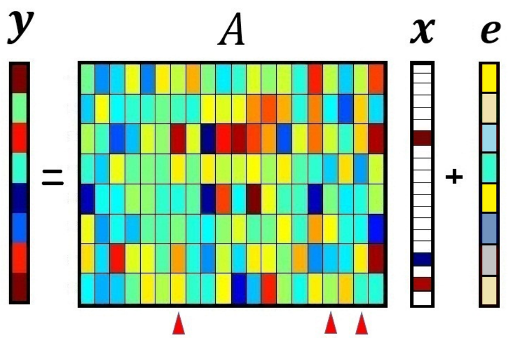


Fig. 1.2: An example of the SMV problem.

which is a convex optimization problem. However, in most practical applications, the measurements are usually contaminated with noise and the signal itself may contain many components of low amplitude. In both cases, the CS model can be redefined as $\mathbf{y} = A\mathbf{x}_s + \mathbf{e}$, where $\mathbf{e}$ accounts for the noise. This problem has been referred to the single measurement vector (SMV) problem. In Fig. 1.2, an example of an SMV problem is illustrated.

The focus of this dessertation is on the reconstruction stage of CS. Although there exist many recovery algorithms for solving the inverse problem of CS, most of these algorithms need some prior knowledge about the number of non-zeros in the signal, the noise level, have tuning parameters, or need some knowledge on the structure of the non-zeros in the signal. Usually, such knowledge is not available to us in real-world applications, which makes these algorithms difficult to work with in order to get a reasonable reconstruction performance.

In this dissertation, several new CS algorithms based on Bayesian modeling are proposed. The implementation of these algorithms have been carried out using different techniques such as Markov chain Monte Carlo (MCMC), variational Bayes (VB) inference, and message passing. Each of these techniques have their own cons and pros mainly by balancing between the execution time against the reconstruction perfomance. The reason for the interest in using Bayesian modelings is their flexibility for incorporating prior knowledge about the structure of the solution compared to the other existing algorithms such as greedy-based algorithms [18–20]. The proposed algorithms mostly do not require any prior knowledge on the signal or its structure. In fact, these algorithms can learn the underlying structure of the signal based on the collected measurements and yet successfully reconstruct the signal, with high probability. Also, if some prior knowledge was available, it can be fed into these algorithms.

In Chapter 2, a new algorithm is proposed for the recovery of jointly-sparse signals with unknown clustered patterns for the multiple measurement vector (MMV) problem. For the MMVs, the solution matrix, which is a collection of sparse vectors, is expected to exhibit some sort of clustered sparsity pattern along the rows of each column as well as joint sparsity across the columns. The notion of joint sparsity means that the columns of the solution matrix share a common support. The proposed algorithm employs a sparse Bayesian learning (SBL) model to encourage the joint sparsity structure across the columns of the solution. A parameter in the model is also incorporated to account for the amount of clumpiness in the supports of the solution in order to improve the recovery performance of sparse signals with an unknown cluster pattern. This parameter does not exist in the other existing algorithms and is learned via the proposed hierarchical SBL algorithm. While the algorithm is constructed for the MMV problems, it can also be applied to the SMV problems. The simulation results will show that the proposed algorithm is effective compared to other algorithms for both the SMV and MMV problems.

Chapter 3 considers finding the solution of the SMV problem with an unknown block-sparsity structure. For this purpose, a new sparse Bayesian learning (SBL) algorithm simplified via the approximate message passing (AMP) framework is proposed. The AMP framework reduces the computational load of the proposed SBL algorithm and as a result makes it faster. Furthermore, in terms of the mean-squared error between the true and the reconstructed solution, the algorithm demonstrates an encouraging improvement compared to the other algorithms.

Chapter 4 considers the recovery of sparse signals with unknown clustering pattern in the case of having partial erroneous prior knowledge on the supports of the signal is considered. In this case, a modified sparse Bayesian learning model is proposed to incorporate prior knowledge and simultaneously learn the unknown clustering pattern. For this purpose, we add one more layer to the support-aided sparse Bayesian learning algorithm (SA-SBL).

This layer adds a prior on the shape parameters of Gamma distributions, those modeled to account for the precision of the solution elements. The shape parameters are made to depend on the total variations on the estimated supports of the solution. The simulation results show that the proposed algorithm is able to modify its erroneous prior knowledge on the supports of the solution and learn the clustering pattern of the true signal by filtering out the incorrect supports from the estimated support set.

In Chapter 5, the performance of sparse signal recovery from a set of compressively sensed noisy measurements using variational Bayesian (VB) inference is investigated. The framework considered here is an ordinary sparse Bayesian learning where no specific structure other than sparsity is assumed on the underlying signal of interest. Two models on the signal are considered, the issues of each model are studied, and the reconstruction performances are compared. In the first model, the sparse signal is considered as the combination of a solution vector, where each component is modeled as a Gaussian distribution with the same precision, and the support vector, where each component is modeled by a Bernoulli distribution. In the second model, the components of the solution are modeled by Gaussian distributions but with different precisions. The issues of each modeling using variational Bayes inference for only promoting the sparsity have not been investigated.

The second direction of this disseration is on the exploration problem using multiple mobile sensors. More specifically, the configuration of multiple mobile sensors to explore an unknown region is investigated. The exploration problem here trades off between two desiderata: to continue taking data in a region known to be interesting with the intent of refining the measurements *vs.* taking data in unobserved areas to attempt to discover new interesting regions. Making reasonable and practical decisions to simultaneously fulfill both goals is hard and contradictory. In this dissertation, a new framework is proposed for such an exploration problem. The proposed framework consists of two main stages. In the first stage, a Gaussian process regression model is employed to predict the phenomena at unseen locations. A Gaussian process (GP) model is useful for modeling spatiotemporal data. GPs are powerful supervised learning tools for regression problems known as Gaussian process regression (GPR) models [21, 22], and are also referred to as Kriging, in the geostatistics literature [23–25]. GPR models are used here as a spatiotemporal interpolator/extrapolator tool to predict the behavior of the phenomenon of interest (PoI) at unsampled data points. GPs have already applied in sensor placement problems [26–31]. For example, a typical sensor placement technique is to use the variances associated with the maximum a *posteriori* (MAP) estimates of GP as a measure representing the amount of uncertainty in the region. This results in placing the sensors at locations with the highest variance (entropy) [27,28], to reduce the overall entropy. This characterization of the quality of sensor placements seems to be naive due to the following reasons. The sensor placement via the measure of variance usually forces the sensors to be placed at the borders of the region under study because there is no measurement outside of the region and as a result the borders tend to have very few measurements in their neighborhood. However, if we continue to perform sensor placement successively using the entropy of the region, there is a high chance ending up with some sort of uniform sampling (equally spaced sensor placements) in the region. This is because the placements tend to occur at the locations far away from the visited locations of the sensors. These criteria only fulfill the exploration goals without taking into account refining measurements of the interesting features about the underlying phenomenon. Also, the available budget on sensors may not allow us to have widely scattered sensor placements in real world applications. The proposed framework is able to tackle with this issue in its second stage referred to as a decision-maker stage.

After collecting the initial measurements over the region, the GPR model predicts the behavior of the PoI at the unseen locations. Then, the decision-maker stage comes into place to decide on the next set of trajectories based on the information obtained from the GPR stage. There is a tradeoff between repeating measurements in the same (or nearby) locations in order to refine information about that region where it is known that interesting things are happening *vs.* making measurements in new locations (exploration) with the possibility of discovering additional interesting information. There is not sufficient information to obtain an optimal solution *a priori*, since the available information is local (the result of previous measurements) and may change over time. In order to deal with the above issues, this work sets up the decision making based on epistemic utility theory [32–35], which forms the second stage of the proposed framework. In epistemic utility theory, as a decision-making framework, decisions are made via emphasis on avoiding wrong decisions and favoring informationally valuable decisions rather than the more restrictive (and sometimes inachievable) task of finding the best solution. As an application, a surrogate example for the exploration problem using a constellation of satellites is investigated, where the goal is tuning or changing the orbital planes of the constellation to find and track the most interesting features of the phenomenon.

The organization of this dissertation is as follows. In Chapter 2, a basic hierarchical SBL model is first constructed to solve the MMVs when the solution shares joint sparsity. Then an algorithm is described which extends this basic model to account for both joint sparsity and the unknown clustering pattern that may exist in the solution. The performance of the new method is then compared to the other algorithms. Also, the convergence diagnostic of the MCMC technique for the proposed algorithm is discussed.

In chapter 3, another new SBL algorithm is proposed for the block-sparse SMV problem based on the message passing and the approximate message passing algorithm proposed in [36] . The new algorithm uses the measure of clumpiness over the supports of the solution (Sigma-Delta) proposed in [37,38]. The performance of the proposed algorithm is compared against some of other existing state-of-the-art algorithms in terms of the normalized mean-squared error between the true and the reconstructed solution.

Chapter 4 deals with the recovery of sparse signals with unknown clustering pattern in the case of having partial erroneous prior knowledge on the supports of the signal. In this case, a new modified sparse Bayesian learning model is proposed to incorporate prior knowledge and simultaneously learn the unknown clustering pattern. For this purpose, one more layer is incorporated into the support-aided sparse Bayesian learning (SA-SBL) algorithm. This layer adds a prior on the shape parameters of Gamma distributions, those modeled to account for the precision of the solution elements. The shape parameters are made depend on the total variations on the estimated supports of the solution. Based on the simulation results, it will be shown that the proposed algorithm is able to modify its erroneous prior knowledge on the supports of the solution and learn the clustering pattern of the true signal by filtering out the incorrect supports from the estimated support set.

Chapter 5 begins with presenting a brief background on VB inference. Then, Bernoullis-Gausssians-inverse Gammas modeling and Gaussians-inverse Gammas modeling for CS using VB are described, accompanied with some motivational examples illustrating the issues with each mdeling and algorithm. Also, a discussion on how to improve the performance for each algorithm is provided. Then, a new algorithm based on VB inference for GiG modeling and a modified version of OMP algorithm is proposed. Finally, the simulation results and comparison of the current work with some other state-of-the-art algorithms is presented.

In chapter 6, some background of the GPR model is first presented. Also, decision making based on the epistemic utility controller is described. Then, these tools are applied to the problem of trajectory determination of mobile sensors. Finally, an example illustrating the effectiveness of the proposed approach is presented.

Chapter 7 represents the in-progess work which has not been accomplished yet. Also, some intuition on the direction for the future work of the study in this dissertation is provided. Finally, Chapter 8 summarizes this dissertation.

CHAPTER 2

BAYESIAN COMPRESSIVE SENSING OF SPARSE SIGNALS WITH UNKNOWN CLUSTERING PATTERNS

This chapter proposes a new algorithm for the recovery of jointly-sparse signals with unknown clustered patterns for the multiple measurement vector (MMV) problem . For the MMVs with this structure, the solution matrix, which is a collection of sparse vectors, is expected to exhibit some sort of clustered sparsity pattern along the rows of each column as well as joint sparsity across the columns. The notion of joint sparsity here means that the columns of the solution matrix share a common support. The proposed algorithm employs a sparse Bayesian learning (SBL) model to encourage the joint sparsity structure across the columns of the solution. Also, a new parameter is incorporated in the model to account for the amount of clumpiness in the supports of the solution in order to improve the recovery performance of sparse signals with an unknown cluster pattern. The notion of clumpiness here simply means that how the supports are scattered and clustered. This parameter does not exist in the other existing algorithms and is learned via the proposed hierarchical SBL algorithm. While the proposed algorithm is constructed for the MMV problems, it can also be applied to the single measurement vector (SMV) problems. Simulation results show the effectiveness of the proposed algorithm compared to other algorithms for both the SMV and MMV problems. A discussion on the number of MCMC iterations is also provided.

2.1 Background and Introduction

Single- and multiple measurement vector (SMV and MMV) problems are computational inverse problems in the compressive sensing (CS) area. The core idea behind compressive sensing is the possibility of representing a sparse or compressible signal from a small set of non-adaptive linear measurements [17, 39]. In linear CS, the P-dimensional signal $\boldsymbol{x} \in \mathbb{R}^P$ is modeled by the linear equation $\boldsymbol{y} = \Phi\boldsymbol{x}$, where $\boldsymbol{y} \in \mathbb{R}^M$ is the measurement vector (with

$M \ll P$) and $\Phi \in \mathbb{R}^{M\times P}$ is a wide sensing matrix. In the CS context, it is further assumed that $\boldsymbol{x}$ is sparse under some proper basis Ψ, i.e., $\boldsymbol{x} = \Psi\boldsymbol{x}_s$, where $\boldsymbol{x}_s$ denotes a sparse vector. A sparse vector contains few non-zero components. Combining the two above equations, we obtain $\boldsymbol{y} = A\boldsymbol{x}_s$, where $A = \Phi\Psi$ [40]. Since the sensing matrix A is wide, the model is underdetermined and CS instead looks for a sparse (if not the most sparse) solution $\hat{\boldsymbol{x}}_s$ such that $\boldsymbol{y} = A\hat{\boldsymbol{x}}_s$ [41, 42]. The SMV is a CS problem when the sensing matrix A is known and the measurements are contaminated with noise $\boldsymbol{e}$, i.e., $\boldsymbol{y} = A\boldsymbol{x}_s + \boldsymbol{e}$. 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. Fig. 2.1 shows an example of the MMV structure.

The problem that we address in this chapter is for the recovery of sparse signals with an unknown cluster pattern via either SMV or MMVs. For this purpose, we provide a new hierarchical sparse Bayesian learning model. 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. While the case of MMVs where the supports change slowly over time (small changes in the location of supports in successive columns of the solution matrix) is not the primary focus of this work, we provide some examples to show that our algorithm still can be used for such MMV problems, as well. Due to the applicability of our algorithm to either the SMV or MMV problems, below we provide a literature review on both.

2.1.1 Literature Review on SMV and MMVs

Finding a sparse representation $\hat{\boldsymbol{x}}_s$ for the SMV problem is achieved practically using greedy algorithms such as matching pursuit (MP) and orthogonal matching pursuit (OMP) [41, 43], or relaxed-to-be-convex methods (such as basis pursuit de-noising (BPDN) and In-Crowd algorithm) [44, 45], the class of Iterative Shrinkage-Thresholding Algorithms (ISTA) and their variations such as FISTA, NESTA, ISTA-NET [46–48], or sparse Bayesian learning (SBL) approaches [49–52]. Similarly, there exist three main approaches for solving

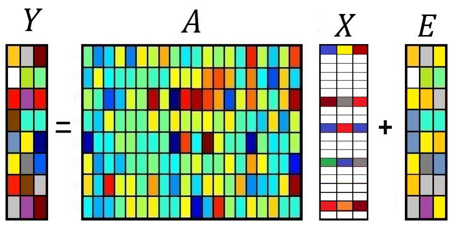


Fig. 2.1: Example of the MMV structure.

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) [53–55]. The second approach is relaxed-to-be-convex algorithms such as joint $l_{2,0}$ approximation algorithm (JLZA) [56]. The third approach is sparse Bayesian learning (SBL) algorithms which are more flexible for incorporating prior knowledge on the structure of the solution compared to the greedy-based algorithms [18–20].

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 the solution matrix X_s in $Y = AX_s + E$. This feature has been referred to as clustered structure or block-sparsity pattern in the literature [43, 57, 58]. Applications of clustered sparsity for the SMV cases arise in problems such as gene expression analysis [59], image reconstruction of hand-written digits [60], and audio signals using the discrete cosine transform (DCT) basis [15]. Applications of MMVs can be found in neuromagnetic imaging [53], the reconstruction stage of Xampling (compressed-sensing of analog signals) for multi-band signals [43, 57], and the direction of arrival (DOA) estimation problem [61]. 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 [58]. 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*. As another application, Mishali and Eldar [43] studied the infinite measurement vectors (IMVs) problem. The IMV structure arises in the recovery problems involving analog signals, such as multiband signals whose spectrum is sparse. In such problems, it is assumed that the number of carrier frequencies and the corresponding maximum bandwidths are known while the band location information is unavailable [1, 57]. In [43] it was proved that the IMV problem can be reduced to an MMV, where the supports appear in pairs.

During the last decade, many greedy-based algorithms have been proposed to solve clustered pattern SMVs such as ReMBo [43], block-OMP [62], StructOMP [63], and group LASSO [64]. However, these algorithms are application-based and they 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 based on the observations. Regarding the sparse recovery of SMV and MMVs, existing SBL algorithms can be 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 [49, 65, 66]. In order to promote the clustered pattern as well as sparsity in these models, different priors have been introduced on the variance of each component of the signal [15, 51, 67]. For example in case of clustered SMV using SBL with zero-mean Gaussian priors, Zhang and Rao incorporated intra-block correlation structure (correlation structure in each block) [51]. 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 recent work, Fang *et al.* used a zero mean Gaussian prior where the precision (inverse variance) on each component is statistically dependent on the precisions of the corresponding component and its two immediate neighbors [15]. In [65], a Dirichlet process prior was imposed on the precisions of the solution components to promote the clustering pattern.

The second SBL approach for the clustered sparse signal reconstruction is to use a spike-and-slab prior [60,68–70]. 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 [60]. Yu *et al.*, made the spike and slab probabilities for each solution component depend on three possible patterns of the neighbor supports [68]. The patterns depend on whether the immediate supports of each component are all active, inactive, or only one of them is active. In [69] a Gaussian process prior was imposed on the spike-and-slab probabilities.

2.1.2 Idea Behind the Proposed Algorithm

Here, we present a new hierarchical Bayesian learning algorithm to solve the MMV problem for sparse signals with an unknown cluster pattern. For the clustered sparse signals in the SMV, the joint sparsity structure does not exist. But, by replacing the measurement and solution matrices with vectors, the proposed model turns into a model for clustered pattern SMV problem. Our model falls within the second sparse Bayesian modeling category mentioned earlier i.e., the spike-and-slab-like prior. We first establish a simple hierarchical Bayesian model for solving the general form of the MMV problem. In this initial model, we use Bernoulli-Gaussian prior, which approximates the spike-and-slab prior, but in a 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 [14,68,71]. In terms of Gaussian-Bernoulli modeling of the sparse signal, our initial model is close to the simplified form of [68,71] and in terms implementation, it uses MCMC implementation with Gibbs sampler techniques as in [14, 68]. A binary matrix was used as a part of the Bayesian modeling in [14] 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 [14] 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 the sparse solutions without any feature to promote the clustering pattern. The main reason for defining this model appears later in this chapter when we modify the model to promote not only sparsity but also to account for the clustered structure that may exist in the solution. The main contribution in this work is related to the modified model, where a total variation-based prior on the support vector of the solution is imposed to promote the clustering patter. This prior is essentially the one-norm of the discrete gradient of the support vector $\boldsymbol{s}$. This hyperprior incorporates a parameter to account for the measure of contiguity, or “clumpiness,” in the supports of the solution. This parameter acts like a knob that determines the overall amount of clumpiness in the supports of the solution. Assigning a large value to the hyperparameter represented by this knob encourages the overall supports of the solution to have fewer on/off transitions, that is, more contiguity of clustering in the supports. Setting the parameter to a small value allows for having many transitions along the components of the support learning vector. 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.

Our proposed algorithm differs from [15] in three aspects. First, our model can be readily applied to either SMV or MMV problems while the original PC-SBL algorithm proposed in [15] solves for the clustered pattern SMVs. Although it has been recently extended via generalized approximate message passing (GAMP) to solve for the 2-d problems [72], it needs some extra modifications to be used for the MMVs. Secondly, our model uses Bernoulli-Gaussian prior and it promotes the clustering pattern by adding hyperpriors on the supports of the solution, while in [15] this task is performed on the variances of the solution components. Finally, our model makes decisions on each support based on both the status of the nearest components and the effect of the overall number of transitions in the supports rather than only the immediate nearest neighbors. Therefore, our algorithm has more control on determining the supports. Yu *et al.* [68] 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 [68] due to the first and third reason we provided earlier for [15]. Tibshirani *et al.* present an algorithm for SMVs referred to as fused-lasso algorithm [59]. The fused-lasso algorithm promotes both sparsity and smoothness in the solution. In this algorithm, the smoothness is promoted by incorporating 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 using such a prior on the solution vector itself. Recently, Fang *et al.* [52], proposed the SA-SBL algorithm, which is useful for the cases where a mixed erroneous set including a portion of the true and false supports is available. Our proposed algorithm does not assume such prior information.

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 [73]. 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 [73], for the purpose of reducing the complexity of the algorithm, the expectation-maximization algorithm using approximate message passing is employed. Our measure of clumpiness can also be incorporated in this model to promote the clustering pattern as well as sparsity, simultaneously.

Part of the current work was presented in [37].

This chapter is organized as follows. In Section 2.2, we construct a basic hierarchical SBL model for solving the MMVs when the solution shares joint sparsity. Section 2.3 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 2.4, we illustrate the performance of our proposed work compared to the other algorithms. Finally, Section 2.5 discusses the convergence diagnostic of the MCMC technique for the proposed algorithm. Section 2.6 presents conclusions.

2.2 Initial Model: Sparse Bayesian Learning for MMVs

As an initial model, here we construct a hierarchical sparse Bayesian learning (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. The model can also be applied to the SMV problem by simply removing the joint sparsity structure and replacing the measurement and solution matrices with vectors. As discussed earlier, this model serves as the initial model which we will modify in Section 2.3 for the clustered pattern sparse signals. We refer to this SBL algorithm as ordinary SBL (O-SBL).

The supports of the solution are modeled by the binary vector $\boldsymbol{s}$. Therefore, the sparse solution is described by $\boldsymbol{s} \circ X$, where $\boldsymbol{s}$ 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. Therefore, the model for the MMV problem is

$$Y = A(\boldsymbol{s} \circ X) + E, \tag{2.1}$$

where $Y \in \mathbb{R}^{M\times N}$, $A \in \mathbb{R}^{M\times P}$, $\boldsymbol{s} \in \{0,1\}^{P\times 1}$, $X \in \mathbb{R}^{P\times N}$, and $E \in \mathbb{R}^{M\times N}$. The matrix Y contains N columns of observed noisy data, A denotes the known sensing matrix, $\boldsymbol{s}$ is an unknown binary support-learning vector, X is an unknown solution-values matrix, and E represents the noise in the measurements. In the product $s \circ X$, when $N > 1$, the support vector $\boldsymbol{s}$ deals across the columns of X. In other words, the term $\boldsymbol{s} \circ X$ is simply equivalent to diag$\{\boldsymbol{s}\} \cdot X$, where '$\cdot$' is the regular matrix product and 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 Fig. 2.2. The shaded node Y shows the

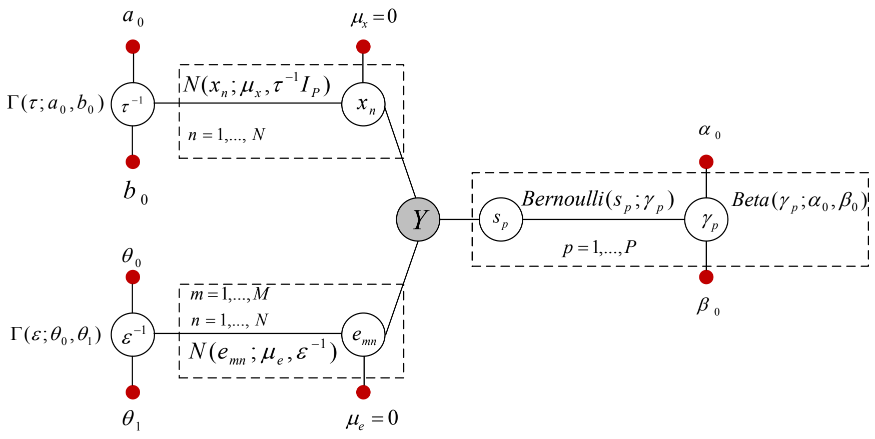


Fig. 2.2: Graphical model of the Bayesian formulation (2.1).

observations and the small solid nodes represent the hyperparameters. Each unshaded node denotes a random variable (or a group of random variables) [74]. The support-learning component $\boldsymbol{s}$ in (2.1) is a binary vector and we model the elements of $\boldsymbol{s}$ as Bernoulli random variables, whose probabilities are governed by the prior $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_P]^T$; that is,

$$s_p \sim \text{Bernoulli}(\gamma_p),\ \gamma_p \sim \text{Beta}(\alpha_0, \beta_0), p = 1, \ldots, P. \tag{2.2}$$

In order to favor the sparsity structure in $\boldsymbol{s}$, on the basis of experimentation we set $\alpha_0 = \frac{10}{P}$ and $\beta_0 = 1 - \alpha_0$, as suggested in [14].

The columns of the solution-value matrix $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ in (2.1) are assumed to be drawn i.i.d. according to the normal-gamma distribution

$$\boldsymbol{x}_n \sim \mathcal{N}(0, \tau^{-1} I_P), \tau \sim \text{Gamma}(a_0, b_0), n = 1, \ldots, N, \tag{2.3}$$

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 in (2.3) to $a_0 = b_0 = 10^{-3}$, endowing X *a priori* with a fairly high variance.

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 learned via inference, so that

$$\begin{aligned} & e_{mn} \sim \mathcal{N}(0, \varepsilon^{-1}),\ m = 1, \dots, M,\ n = 1, \dots, N, \\ & \varepsilon \sim \text{Gamma}(\theta_0, \theta_1). \end{aligned} \tag{2.4}$$

The hyper-parameters in (2.4) are set to $\theta_0 = \theta_1 = 10^{-3}$. This setting may vary under the desired precision or prior knowledge on the noise variance.

Referring to the graphical model in Fig. 2.2, the joint probability distribution of model can be written as

$$p(Y, \boldsymbol{s}, X) \propto p(Y|\boldsymbol{s}, X, \varepsilon)\Big(\prod_{n=1}^{N} p(\boldsymbol{x}_n|\mathbf{0}, \tau^{-1}I_P)\Big)p(\boldsymbol{s}|\boldsymbol{\gamma})p(\boldsymbol{\gamma}; \alpha_0, \beta_0)p(\tau; a_0, b_0)p(\varepsilon; \theta_0, \theta_1). \tag{2.5}$$

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

- $$(s_p|-) \sim \text{Bernoulli}(\frac{q_1}{q_0 + q_1}), \tag{2.6}$$

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

In the above equation, $\tilde{\boldsymbol{y}}_n^{-p} = [\tilde{y}_{1n}^{-p}, ..., \tilde{y}_{mn}^{-p}]^T$ and the term $\tilde{y}_{mn}^{-p}$ is defined as $\tilde{y}_{mn}^{-p} = y_{mn} - \sum_{l \neq p}^{P} a_{ml}s_l x_{ln}$. For more detail, please see the Appendix 2.7.1.

- $(\gamma_p|-) \propto p(s_p|\gamma_p)p(\gamma_p; \alpha_0, \beta_0)$

  $$\propto \gamma_p^{s_p}(1 - \gamma_p)^{1-s_p}\gamma_p^{\alpha_0 - 1}(1 - \gamma_p)^{\beta_0 - 1}$$

  Therefore, $(\gamma_p|-) \sim \text{Beta}(\alpha_0 + s_p\ ,\ \beta_0 + 1 - s_p)$.

- $(x_{pn}|-) \sim \mathcal{N}(\mu_{x_{pn}}, \Sigma_{x_{pn}})$, where $\mu_{x_{pn}} = \varepsilon s_p \Sigma_{x_{pn}} \mathbf{a}_p^T \tilde{\mathbf{y}}_n^{-p}$ and $\Sigma_{x_{pn}} = (\tau + \varepsilon s_p^2\, \|\mathbf{a}_p\|_2^2)^{-1}$.

- $(\tau|-) \propto p(X|0, \tau^{-1})p(\tau; a_0, b_0)$

$$\propto \tau^{\frac{NP}{2}} e^{-\frac{\tau}{2}\|X\|_F^2} \tau^{a_0-1} e^{-b_0\tau}$$

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,

- $(\varepsilon|-) \propto p(Y|\boldsymbol{s}, X, \varepsilon)p(\varepsilon; \theta_0, \theta_1)$

$$\propto \varepsilon^{\frac{MN}{2}} e^{-\frac{1}{2}\varepsilon\|Y-A(\boldsymbol{s}\circ X)\|_F^2} \varepsilon^{\theta_0-1} e^{-\varepsilon\theta_1}$$

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

One can derive the inference on the other parameters by following the same approach as described in the Appendix.

In the implementation, the above posterior densities are approximated via Markov chain Monte Carlo (MCMC) using Gibbs sampling. The idea behind this approach is to approximate each posterior density by a collection of samples drawn iteratively from the conditional posterior distribution of the corresponding random variable given the most recent estimated values of all the other parameters [14]. The pseudocode description the O-SBL algorithm is represented below.

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

In the above algorithm, the term Θ_0 contains the set of initial values of the parameters of interest. The initialization of the parameters is performed by drawing random samples from the corresponding prior distributions defined in (2.2) to (2.4). We then run the O-SBL algorithm for the number of $N_{\text{burn-in}}$ iterations. $N_{\text{burn-in}}$ is set experimentally based on the number of iterations required for reaching the stable Markov chain. We do not collect any samples during the burn-in period. Once the algorithm iterates for $N_{\text{burn-in}}$ times, we perform N_{collect} more iterations 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 nth iteration. As an alternative, one may use the samples obtained at 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 convergence issue of the MCMC for the algorithm will be discussed in Section 2.4.

2.3 Clustered-Sparse Bayesian Learning

In this section, we modify the initial SBL model described in Section 2.2 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 each of the vectors might have groups of clumps i.e., groups of adjacent non-zero terms. Following [75], we measure the amount of clumpiness in the support-learning vector $\boldsymbol{s}$ by the absolute sum of the differences between successive elements of $\boldsymbol{s}$,

$$\Sigma\Delta(\boldsymbol{s}) = \sum_{p=2}^{P} |s_p - s_{p-1}|. \tag{2.7}$$

There exist fewer transitions in $\boldsymbol{s}$, corresponding to a smaller $\Sigma\Delta$, when the supports of the solution have a clustered pattern compared to unstructured distribution of supports. In Fig. 2.3, we illustrate some examples for the binary vector $\boldsymbol{s}$ followed by the corresponding $\Sigma\Delta$ value. For example, a constant vector $\boldsymbol{s}$ ($\boldsymbol{s} = \boldsymbol{0}$ or $\boldsymbol{s} = \boldsymbol{1}$) has a $\Sigma\Delta$ equal to 0. In Fig. 2.3, the indices with active supports are shown by shaded cells. The $\Sigma\Delta$ measure is used to establish a prior probability which encourages sparsity. This task is accomplished by setting the prior for the support-learning vector $\boldsymbol{s}$ proportional to $e^{-\alpha(\Sigma\Delta)(\boldsymbol{s})}$ 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 $\boldsymbol{s}$, while small values of α result in

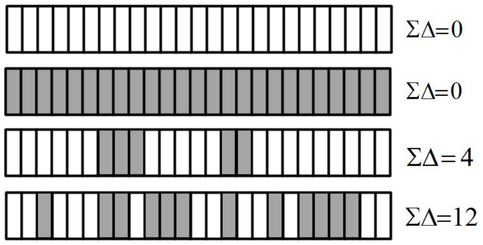


Fig. 2.3: Examples of the vector $\boldsymbol{s}$ and the corresponding $(\Sigma\Delta)$ value.

accepting many transitions along the elements of $\boldsymbol{s}$. We model the prior on the elements of the support-learning vector $\boldsymbol{s}$ as follows

$$(s_p; \Omega_p) \sim \text{Bernoulli}(\Omega_p), \forall p = 1, 2, ..., P, \tag{2.8}$$

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

$$\omega_{k,p} = e^{-\alpha(\Sigma\Delta)_{k,p}} \text{Binomial}(\Sigma_{k,p}, P, \gamma_p), k \in \{0, 1\}, \tag{2.9}$$

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 $\boldsymbol{s}$ in the case when we force $s_p = k$. In other words, $\Sigma_{k,p}$ is the number of active elements in $\boldsymbol{s}$ for the case where the pth component of $\boldsymbol{s}$ is set to be 1 (active, via $k=1$) or 0 (inactive, via $k=0$). For example, $\Sigma_{1,5} = 1 + \sum_{p\neq 5}^{P} s_p$. Also, in (2.9) the term $(\Sigma\Delta)_{k,p}$ is the measure of clumpiness evaluated via (2.7) for the case of forcing the pth component of $\boldsymbol{s}$ 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 $\boldsymbol{s}$ (2.8) can be further simplified into

$$(s_p;\Omega_p) \sim \text{Bernoulli}(\Omega_p), \Omega_p := \frac{1}{1+c_p e^{-\alpha(\bar{\Sigma\Delta})_p}}, \forall p, \tag{2.10}$$

where

$$c_p := \frac{1-\gamma_p}{\gamma_p}\frac{\Sigma_{1,p}}{P-\Sigma_{0,p}}, \forall p = 1, \dots, P, \tag{2.11}$$

and

$$(\bar{\Sigma\Delta})_p = (\Sigma\Delta)_{0,p} - (\Sigma\Delta)_{1,p}, \forall p = 1, \dots, P. \tag{2.12}$$

Roughly speaking, this distribution favors drawing $s_p = 1$ if this draw reduces $(\Sigma\Delta)_{1,p}$. Let's 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 (2.10) will be simplified into $\Omega_p = \gamma_p$, which is the same prior as we had earlier in (2.2). 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 (2.10), the supports exhibit a trade off between sparsity and clustering.

In Fig. 2.4, we demonstrate the effect of α defined in (2.9) and its role in the prior (2.10) on the support learning vector $\boldsymbol{s}$. The figure shows draws of $\boldsymbol{s}$ according to (2.10) using (2.11) and (2.12). Larger values of α result in lower $\Sigma\Delta(\boldsymbol{s})$, meaning that the support vector $\boldsymbol{s}$ will tend to have fewer transitions along its components. In other words, larger values of α promote clustering pattern in $\boldsymbol{s}$ and vice versa when α is estimated.

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. We set $a_1 = 2 \times 10^{-3}$ and $b_1 = 10^{-3}$ in our simulations. With these priors, the joint probability

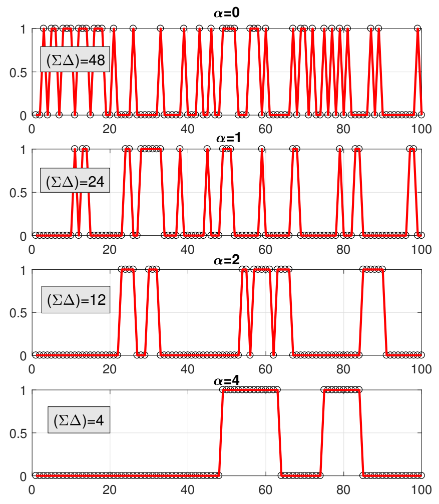


Fig. 2.4: Example showing the effect of α on the pattern of support vector $\boldsymbol{s}$.

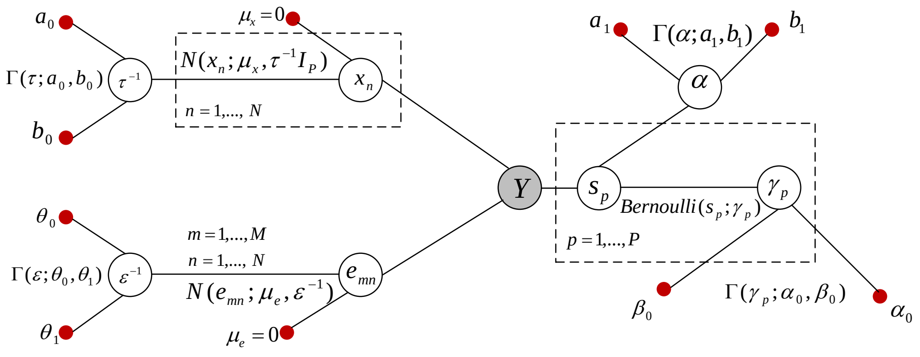


Fig. 2.5: Graphical model of the Bayesian formulation (2.13).

distribution for the complete model becomes

$$
\begin{aligned}
p(Y, \boldsymbol{s}, X) \propto p(Y|\boldsymbol{s}, X, \varepsilon)\Big(\prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{0}, \tau^{-1} I_P)\Big) p(\tau; a_0, b_0) \times \\
p(\varepsilon; \theta_0, \theta_1)\Big(\prod_{p=1}^{P}(s_p|\Omega_p)\Big) p(\boldsymbol{\gamma}|\boldsymbol{s}, \alpha_0, \beta_0) p(\alpha; a_1, b_1).
\end{aligned}
\tag{2.13}
$$

The modified graphical model for the clustered pattern MMVs is shown in Fig. 2.5. Below, we describe the inference on the variables which are modified by (2.13) compared to (2.5). The inference on the other variables and parameters in the model is the same as those we described in Section 2.2.

Posterior for s_p

The posterior for s_p is given by

$$\begin{aligned} p(s_p|-) &\propto p(Y|s_p, X)p(s_p|\Omega_p) \\ &\propto \mathrm{Bernoulli}(s_p|\Omega_p)e^{-\frac{\varepsilon}{2}(\|\boldsymbol{a}_p\|_2^2(\sum_{n=1}^N x_{pn}^2)s_p^2 - 2\boldsymbol{a}_p^T(\sum_{n=1}^N x_{pn}\tilde{\boldsymbol{y}}_n{}^{-p})s_p)}, \end{aligned}$$

where Ω_p was defined in (2.8). Using the fact that s_p is a binary random variable, the posterior inference on s_p can be simplified to

$(s_p|-) \sim \mathrm{Bernoulli}(Q_p)$, 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

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

The posterior for s_p can be further simplified into

$$(s_p|-) \sim \mathrm{Bernoulli}\Big(\frac{1}{1 + c_p\kappa_p e^{-\alpha(\Sigma\bar{\Delta})_p}}\Big), \tag{2.14}$$

where c_p and $(\Sigma\bar{\Delta})_p$ were defined in (2.11) and (2.12), respectively, and

$$\kappa_p := e^{\frac{\varepsilon}{2}\left(\|\boldsymbol{a}_p\|_2^2(\sum_{n=1}^N x_{pn}^2) - 2\boldsymbol{a}_p^T(\sum_{n=1}^N x_{pn}\tilde{\boldsymbol{y}}_n{}^{-p})\right)}. \tag{2.15}$$

Posterior for γ_p

The posterior for γ_p is given by

$$p(\gamma_p|-) \propto p(\gamma_p; \alpha_0, \beta_0)\prod_{k=0}^{1} p(\omega_{k,p}|\Sigma_{k,p}, \gamma_p, \alpha, \boldsymbol{s}).$$

Thus $(\gamma_p|-) \sim \mathrm{Beta}(\alpha_1, \beta_1)$, where

$$\alpha_1 := \alpha_0 + 1 + 2\sum_{i\neq p}^{P} s_i \quad \beta_1 := \beta_0 - 1 + 2(P - \sum_{i\neq p}^{P} s_i). \tag{2.16}$$

Update Rule for α

Using the joint probability distribution of the complete model (2.13) and discarding the terms that are unrelated to α, we have

$$\begin{aligned} p(\alpha|-) &\propto p(\alpha; a_1, b_1) \prod_{p=1}^{P} p(s_p|\Omega_p) \\ &\propto \alpha^{a_1-1} e^{-b_1\alpha} \prod_{p=1}^{P} \text{Bernoulli}(\frac{1}{1 + c_p e^{-\alpha(\Sigma\bar{\Delta})_p}}). \end{aligned} \tag{2.17}$$

Due to the complicated nature of (2.17), 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

$$\hat{\alpha}^{[t+1]}_{MAP|Y,X^{[t]},s^{[t]},\Theta^{[t]}} = \arg\max_{\alpha} L_{\alpha|Y,X^{[t]},s^{[t]},\Theta^{[t]}}, \tag{2.18}$$

where L_α is obtained by taking the logarithm of (2.17), and is defined as

$$\begin{aligned} L_\alpha &= \log\{p(\alpha; a_1, b_1) \prod_{p=1}^{P} p(s_p|\Omega_p)\} \\ &\propto (a_1 - 1)\log\alpha - b_1\alpha + \sum_{p=1}^{P} \{s_p \log\Omega_p + (1 - s_p)\log(1 - \Omega_p)\}, \end{aligned} \tag{2.19}$$

and may be further simplified into

$$L_\alpha \propto (a_1 - 1)\log\alpha - b_1\alpha - \alpha\sum_{p=1}^{P}(1 - s_p)(\Sigma\bar{\Delta})_p - \sum_{p=1}^{P}\log\{1 + c_p e^{-\alpha(\Sigma\bar{\Delta})_p}\}. \tag{2.20}$$

Taking 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

$$\frac{a_1-1}{\alpha^{[t+1]}}-b_1-\sum_{p=1}^{P}(1-s_p^{[t]})(\bar{\Sigma\Delta})_p^{[t]}+\sum_{p=1}^{P}\frac{(\bar{\Sigma\Delta})_p^{[t]}}{1+\frac{1}{c_p^{[t]}}e^{\alpha^{[t+1]}(\bar{\Sigma\Delta})_p^{[t]}}}=0 \tag{2.21}$$

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 highly clustered solution, we recommend to set $\alpha \gg 0$.

Remark 2.1: In [37], we estimated the marginal posterior inference on γ_p by $(\gamma_p|-)\sim \text{Beta}(\alpha_0+s_p, \beta_0-s_p)$, $\forall p=1,\ldots,P$. 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 Fig. 2.5. However, in (2.16) we have removed this assumption and directly found the inference based on the relationship between the variables that we have in Fig. 2.6. Also, we have provided an analytical approach for the update rule of α, rather than the empirical approach discussed in [37].

Below we provide a pseudocode description of our algorithm, referred to as C-SBL, which will work for the clustered patterns of either SMV or MMV data.

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 $P$
        $\tilde{y}_{mn}^{-p} = y_{mn} - \sum_{l \neq p}^{P} a_{ml} s_l x_{ln}, \quad \forall m = 1 \text{ to } M, \forall n = 1 \text{ to } N$
        $c_p = \frac{1-\gamma_p}{\gamma_p} \frac{\Sigma_{1,p}}{P+1-\Sigma_{1,p}}, \; (\bar{\Sigma\Delta})_p = (\Sigma\Delta)_{0,p} - (\Sigma\Delta)_{1,p}$
        $k_p = e^{\frac{\varepsilon}{2}\left((\|\boldsymbol{a}_p\|_2^2 \sum_{n=1}^{N} x_{pn}^2) - 2\boldsymbol{a}_p^T (\sum_{n=1}^{N} x_{pn} \tilde{\boldsymbol{y}}_n^{-p})\right)}$
        $(s_p|-) \sim \text{Bernoulli}(\frac{1}{1+c_p k_p e^{-\alpha(\bar{\Sigma\Delta})_p}})$
        % Solution-value matrix component
        For $l = 1$ to $P$
            $\Sigma_x = (\tau + \varepsilon s_l^2 \|\mathbf{a}_l\|_2^2)^{-1}$
            $\bar{\boldsymbol{\mu}} = \varepsilon s_l \Sigma_x \mathbf{a}_l$
            $\tilde{\boldsymbol{y}}_n^{-l} = \boldsymbol{y}_n - A(\boldsymbol{s} \circ \boldsymbol{x}_n) + s_l x_{l,n} \boldsymbol{a}_l, \; \forall n = 1, \ldots, N$
            $(x_{l,n}|-) \sim \mathcal{N}(\bar{\boldsymbol{\mu}}^T \tilde{\boldsymbol{y}}_n^{-l}, \Sigma_x), \; \forall n = 1, \ldots, N$
        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 $\{p\}$
        $(\tau|-) \sim \text{Gamma}(a_0 + \frac{NP}{2}, b_0 + \frac{1}{2}\|X\|_F^2)$
        $(\varepsilon|-) \sim \text{Gamma}(\theta_0 + \frac{MN}{2}, \theta_1 + \frac{1}{2}\|Y - A(\boldsymbol{s} \circ X)\|_F^2)$
        $\alpha$ : obtained from solving (2.21) 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. The convergence diagnostic of the MCMC technique to decide on the number of burn-in iterations for the variables of interest will be discussed in Section 2.4.

2.4 Simulation Results

We compare our algorithm with other algorithms on both synthetic/simulated and real-world data for SMV and MMVs.

2.4.1 Simulations on Synthetic Data

We first compare the proposed algorithm with other algorithms for the SMV problem i.e., $N = 1$. We then consider the MMV problem (2.1) for the case where the number of columns in the solution matrix X is $N = 2, 5$. In this case, we compare the performance of our algorithm with other existing algorithms that are devised for solving the MMV problem.

Performance for the SMV Problem

Each trial, generated independently, is constructed as follows. We consider the solution vector $\boldsymbol{x} \in \mathbb{R}^{100}$, i.e., $P = 100$ and $N = 1$. The supports of the true solution are drawn randomly so that the support vector $\boldsymbol{s}$ exhibits a random clustered sparsity pattern. The total number of non-zeros in the solution is set to 25 for all the trials. The nonzero elements of the solution vector $\boldsymbol{x}$ at the active supports of $\boldsymbol{s}$ 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 1, and then we normalize A with respect to its columns. 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 is 25 dB and is defined as SNR$= 20 \log_{10}(\sigma_x/\sigma)$. Finally, the measurement $\boldsymbol{y}$ is computed from

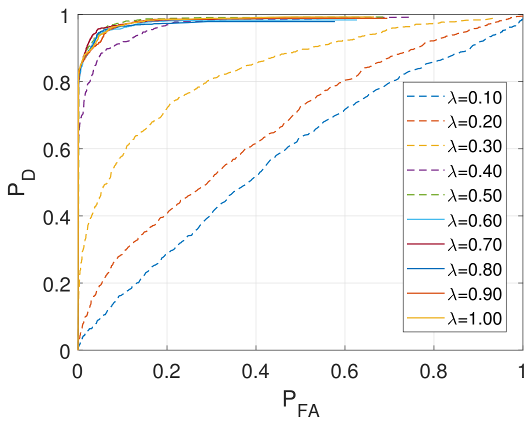
$\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$. The data described above were generated for 200 trials.

We demonstrate the recovery performance of the algorithms using both probabilities of support recovery and also 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}, respectively defined as $P_D = (\#\text{Correct detections})/(\#\text{Possible correct detections})$, $P_{FA} = (\#\text{Falsedetections})/((\#\text{Possible detections}) - (\#\text{Correct detections}))$. A successful reconstruction is reported when all the supports of the true solution are recovered. This essentially evaluates the performance of the algorithm in terms of detection. We also evaluate the performance using normalized mean-squared error defined as

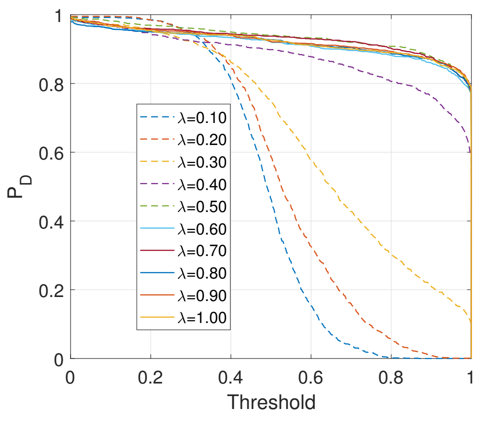
$$\text{NMSE (dB)} := 20 \log_{10} \frac{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2}{\|\boldsymbol{x}\|_2}, \tag{2.22}$$

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

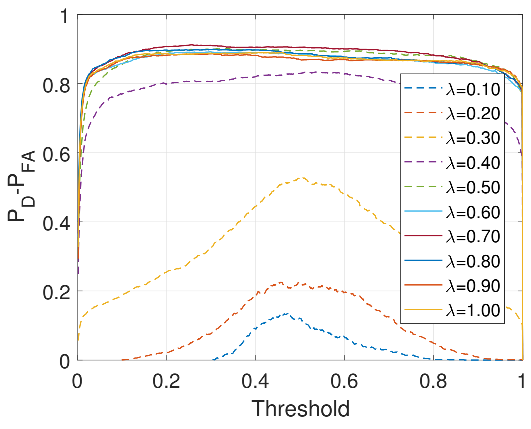
Figures 2.6(a) – (d) demonstrate aspects of performance of the C-SBL algorithm. Details on setting the $N_{\text{burn-in}}$ period are in Section 2.5. In Fig. 2.6(a), we demonstrate the performance of C-SBL using receiver operating characteristic (ROC) curves as the number of measurements, and equivalently the ratio $\lambda = M/P$, varies. In Fig. 2.6(a), we observe that for $\lambda > 0.4$ $(M > 40, P = 100)$, C-SBL successfully finds all the supports of the true solution with reasonably low false alarm rate. In other words, the algorithm produces very high performance when the number of measurements becomes almost twice the number of true non-zeros in the solution (the true number of non-zeros was set to 25). Fig. 2.6(a) does not show how the decisions on the supports for a desired performance can be made using the collected samples. Therefore, in Fig. 2.6(b) – (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 $\{0, 1\}$. Then, those indices in the resulting vector 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 Fig. 2.6(b), we illustrate the



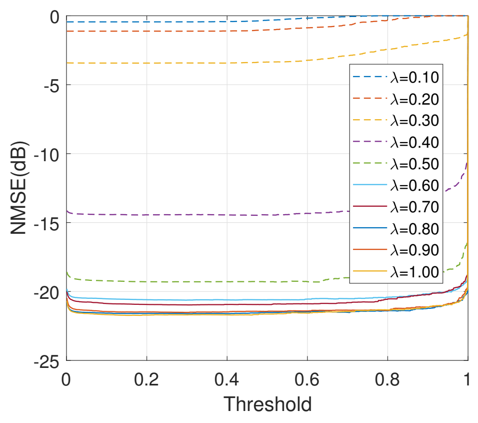
(a) Empirical ROC.



(b) Detection rate P_D.



(c) Support recovery $P_D - P_{FA}$.

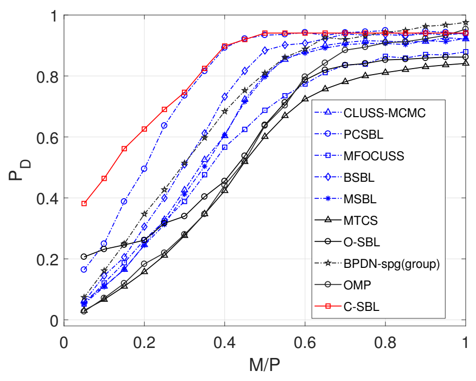


(d) NMSE (dB).

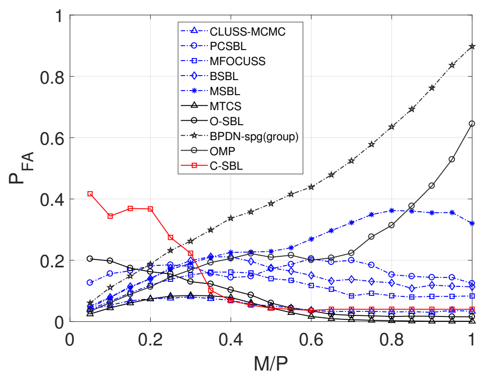
Fig. 2.6: Aspects of performance of the C-SBL algorithm for the SMV problem

detection rate *vs.* the threshold as the ratio M/P changes. Notice that 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 1 for all λ. As another way of evaluating the performance, we also show the difference between detection rate and false alarm rate of C-SBL *vs.* the threshold in Fig. 2.6(c). This figure essentially combines information about the ROC, threshold change, and the effect of varying the ratio λ on the performance of C-SBL. The higher $P_D - P_{FA}$ becomes, the higher performance the algorithm possesses. This is due to the fact that a reasonable performance requires high detectin rate and low false alarm rate in support recovery, simultaneously. In Fig. 2.6(c), we see that 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 within 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, Fig. 2.6(d) shows the C-SBL behavior in terms of normalized mean-squared error, defined in (2.22), *vs.* the threshold. We see in Fig. 2.6(d) that the error term for each λ remains almost constant regardless of the threshold. This means that one can take a threshold which 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 Fig. 2.6(a) and Fig. 2.6(c), different choices of the threshold will result in a different false alarm rate. Also, we see in Fig. 2.6(d) that once the number of measurements becomes around twice the sparsity level ($\lambda \geq 0.5$), the error becomes almost negligible.

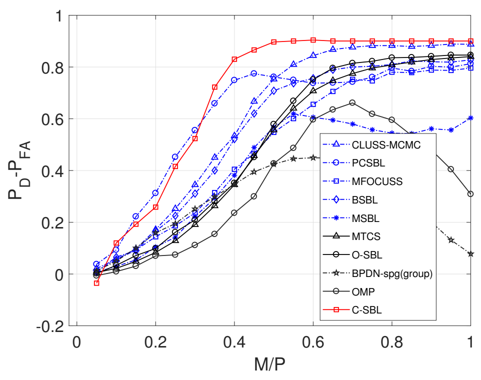
We now compare the performance of C-SBL and O-SBL with other algorithms, specifically: CLUSS-MCMC [68], OMP [41,76], MFOCUSS [53], BSBL [20,51], MSBL [19], BPDN-group using SPGL1 solver [77], single-task version of MTCS [66, 78], and PC-SBL [15, 79]. For the simulation purposes, we generated the elements of the true solution, the sensing matrix, and the measurement noise in the same way as described earlier in this section. The performance is then evaluated via averaging over 200 trials. In all of the algorithms, we discarded those estimated elements in the solution with the amplitude less than 0.01 from



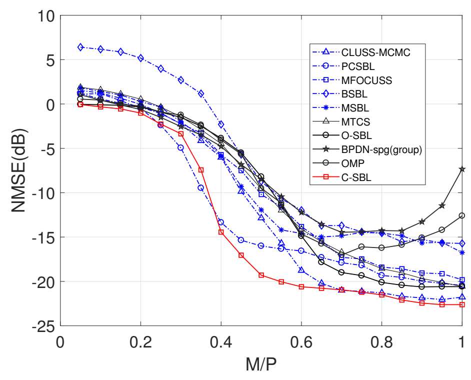
(a) Detection rate P_D.



(b) False alarm P_{FA}.



(c) Support recovery $P_D - P_{FA}$.



(d) NMSE (dB).

Fig. 2.7: Comparisons of various algorithms in the SMV case

the support set. In Fig. 2.7(a) — (d), the results for the O-SBL and C-SBL algorithms are based on the sample mean of the collected samples i.e., threshold of 0.5, as was demonstrated in Fig. 2.6(a) — (d).

In Fig. 2.7(a), we demonstrate the empirical results of detection rate *vs.* the ratio M/P. We see in Fig. 2.7(a) that C-SBL provides the best performance in terms of detecting the true supports of the solution. In Fig. 2.7(b) the empirical results in terms of 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 higher detection rate within the same range for M/P. However, one can set a larger value of threshold for $M/P < 0.35$ in order to get a much lower false alarm rate in support recovery at the cost of getting lower detection rate (See Fig. 2.6(a)—(b)). In contrast, the rate for C-SBL, O-SBL, CLUSS-MCMC, and MTCS become almost the same and with the lowest values.

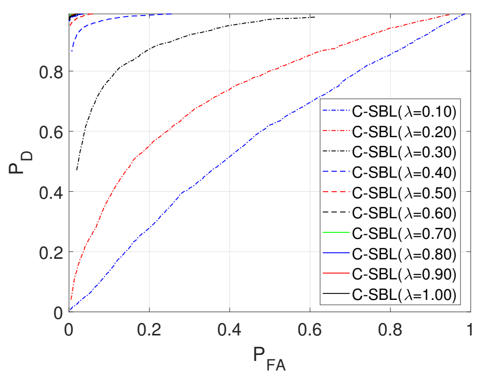
Fig. 2.7(c) shows the performance comparison of the algorithms in terms of the trade off between the detection rate and false alarm rate in support recovery, in which PC-SBL shows slightly better performance for $M/P < 0.35$. However, C-SBL outperforms all the other algorithms for $M/P > 0.35$. Moreover, we can definitely see the effect of incorporating our measure of clumpiness in the supports by comparing the performance of C-SBL with O-SBL. Finally, Fig. 2.7(d) 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 2.2: The codes for BSBL, MSBL, and MFOCUSS were obtained from [80]. For BSBL, the noise flag was set to 2 (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 [81] and PCSBL [79]. The parameters of Gamma prior on the noise variance for MTCS [78] were both set to 1.

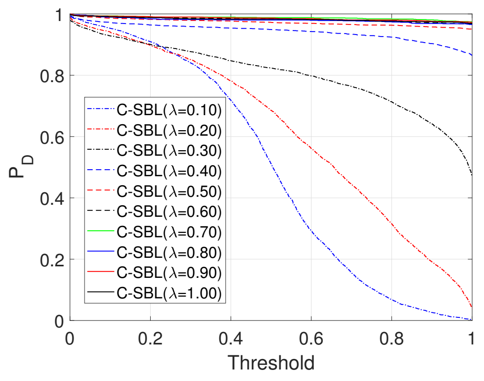
Remark 2.3: It has been very common in the literature to demonstrate the performance primarily on NMSE, so that the successful recovery is reported when the NMSE becomes lower than some pre-defined value. In that sense and by referring to Fig. 2.7(d), we see that our algorithm provides the highest success rate for a wide range of M/P. In addition, our algorithm demonstrates good performance on an ROC plot, showing high detection against the false alarm rate.

Performance for the MMV Problem with $N = 2$

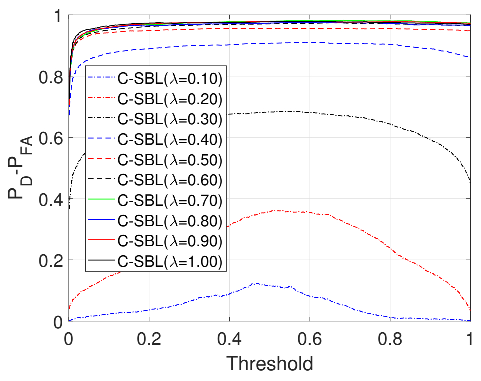
We first demonstrate the performance of C-SBL in terms of detection rate and false alarm rate in support recovery as the ratio M/P varies. Also the performance in terms of NMSE is considered. Similar to the SMV case, we generate 200 independent trials and then average over all the obtained results. In Fig. 2.8, we depict the overall performance of the C-SBL algorithm.



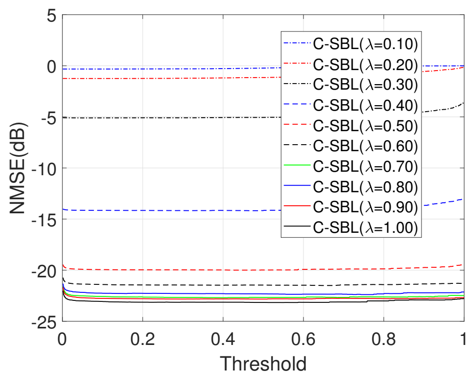
(a) Empirical ROC.



(b) Detection rate P_D.



(c) Support recovery $P_D - P_{FA}$.



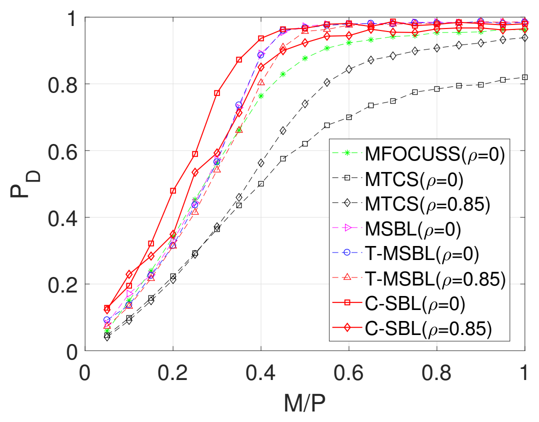
(d) NMSE (dB).

Fig. 2.8: Aspects of performance of C-SBL for MMV (with $N = 2$)

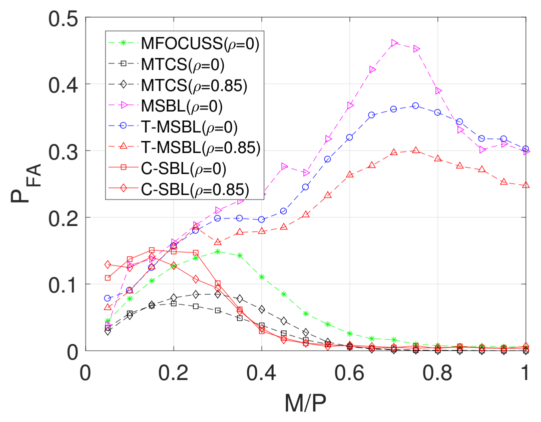
Fig. 2.8(a) displays ROC curves. Comparing the results demonstrated in Fig. 2.6(a) with Fig. 2.8(a), we see that increasing the number of columns in the solution from $N=1$ to $N=2$ provides considerable improvement in the support recovery, as expected. Fig. 2.8(b) illustrates the detection rate of C-SBL for the MMVs (with $N=2$) *vs.* different threshold values. According to Fig. 2.8(b), we observe that that 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 detection rate and false alarm rate of C-SBL *vs.* the threshold is shown in Fig. 2.8(c), in which we see that 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 to 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. In Fig. 2.8(d), we illustrate the behavior of the NMSE *vs.* the threshold for the clustered MMV problem using the C-SBL algorithm.

Finally, Figures 2.9(a) – (d) compare the performance of C-SBL against the other algorithms. We compare the results of C-SBL with the following algorithms: MFOCUSS [53], MSBL [19], T-MSBL [82,83], and MTCS [66]. Notice that T-MSBL is devised for correlated signals while our model does not account for this feature. We should emphasize that MTCS does not promote clustering pattern in the solution. However, it serves as a baseline for our comparisons. For the purpose of providing fair comparisons, here we consider two set 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 Figures 2.9(a) – (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 $\rho = 0.85$.

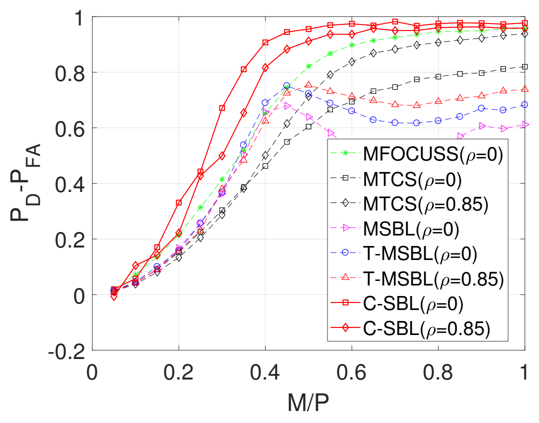
Fig. 2.9(a), illustrates the detection rate in support recovery for clustered pattern MMVs (with $N=2$), in which we observe that C-SBL has the highest performance among the other algorithms for the uncorrelated case. Also, we observe that C-SBL competes with T-MSBL for the correlated case. In Fig. 2.9(b), it is clear that C-SBL provides lower false



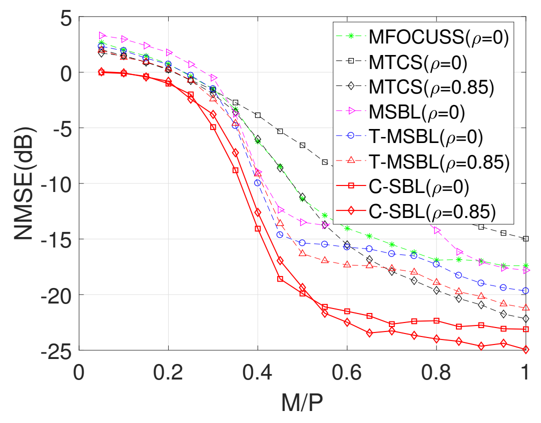
(a) Detection rate P_D.



(b) False alarm rate P_{FA}.



(c) Support recovery $P_D - P_{FA}$.



(d) NMSE (dB).

Fig. 2.9: Comparison of various algorithms in the MMV case with $N = 2$

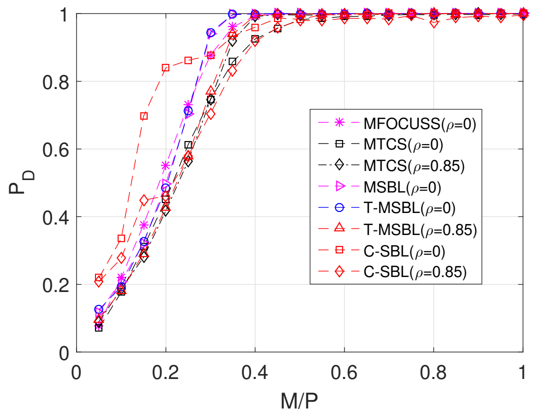
alarm rate in terms of support recovery compared to the MSBL and T-MSBL algorithms. The best performance belongs to MTCS, but it provided the lowest performance in terms of detection rate as was shown in Fig. 2.9(a). For overall comparison, Fig. 2.9(c) shows the simulation results in terms of the difference between the detection rate and false alarm rate in support recovery when varying the ratio M/P. According to Fig. 2.9(c), the overall performance of C-SBL is higher than the other algorithms. The NMSE comparisons are illustrated in Fig. 2.9(d), where we see that C-SBL provided the lowest error. According to Fig. 2.9(a)—(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.

Performance for the MMV Problem with $N=5$

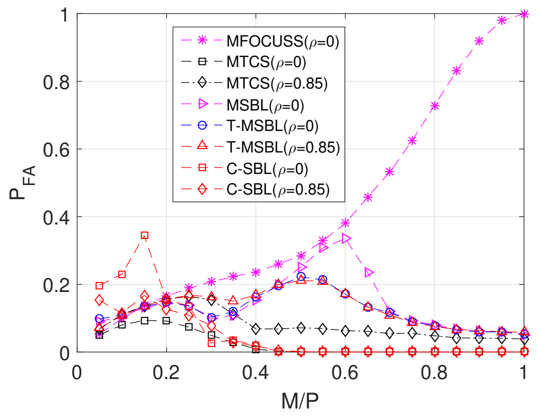
Here we perform simulations on synthetically generated data in the same way explained previously except setting $N = 5$. The burn-in and collection periods of C-SBL were set to 2000 and 1000, respectively. Fig. 2.10 illustrates the performance comparison results for both the uncorrelated case ($\rho = 0$) and the correlated case (with $\rho = 0.85$).

Interpretation of the results for the uncorrelated case

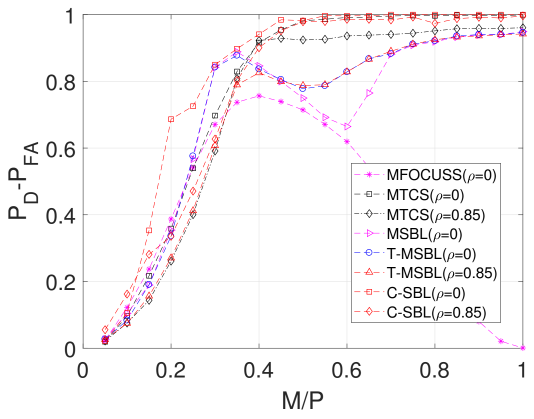
According to Fig. 2.10(a), the best performance in terms of the difference between the detection rate and false alarm rate among the other algorithms (with $\rho=0$) belongs to the C-SBL algorithm. The lowest performance belongs to FOCUSS, where as the sampling ratio becomes greater than 0.4, FOCUSS starts to activate more components in $\boldsymbol{s}$. As a result, the false alarm rate increases and this yield to the smaller P_D-P_{FA}. For instance, for sampling ratio of 1, all the components of $\boldsymbol{s}$ are active, resulting in $P_D-P_{FA}=0$. Notice that 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 were having very low amplitudes. But, according to the error curve of FOCUSS in Fig. 2.10(b), this has not happened, 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



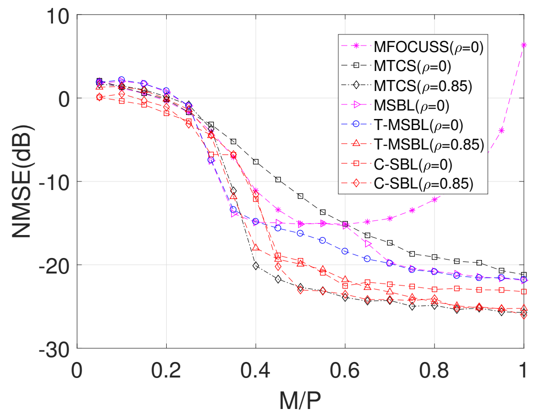
(a) Detection rate P_D.



(b) False alarm rate P_{FA}.



(a) Support recovery $P_D - P_{FA}$.



(b) NMSE (dB).

Fig. 2.10: Comparison of various algorithms in the MMV case with $N = 5$

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. As the conclusion, C-SBL demonstrates the best performance in support recovery for the uncorrelated case.

The comparison of the error between the true and estimated solution is shown in Fig. 2.10(b). According to this figure, C-SBL provides the lowest error for most possible range of λ ($\lambda \in [0.05, 0.25) \cup [0.45, 1]$) for the uncorrelated case.

Interpretation of the results for the correlated case

For the correlated case, Fig. 2.10(a) shows that C-SBL performed a little bit better than the two other compared algorithms in terms of support recovery. It worth reemphasizing that the proposed C-SBL model does not account for the correlation that may exist across the columns of X. According to Fig. 2.10(b), MTCS, T-MSBL, and C-SBL have almost the same overall performance in terms of error.

Remark 2.4: Our main interest driving the current work is to improve the performance in learning the true underlying supports of the sparse signals. The merit of our algorithm is that it provides two sets of information about the solution, one on the supports ($\boldsymbol{s}$) and the other on the non-zero values (X). Therefore, it allows making the decision based on either the combination of X and $\boldsymbol{s}$, or based on $\boldsymbol{s}$ and then solve for X_{np} in the reduced dimension determined by the active components in $\boldsymbol{s}$.

2.4.2 Experiments on Real Data

We further evaluate the performance of the algorithms in reconstructing images of hand-written digits. The data set that we use is obtained from the well-known MNIST data set [84]. MNIST consists of 70,000 gray-scale images of 28×28 of hand-written digits. Experiments are conducted on a randomly chosen set of hand-written digits from ‘0’ to ‘9’ from this data set, where only some of the obtained results are shown here. The original images are scaled up to become images of size 100×100 pixels. The pixel values were normalized to be within $[0, 1]$. Then, the pixel values were subtracted from 1, and those with value of less than 0.3 were set to zero. This setting is more general than some other works on MNIST where binary pixel values were considered [85, 86]. The corresponding matrix representing the pixel values of each image is then treated as the true sparse signal of interest, denoted by the true solution matrix X.

Performance for the SMV Case

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 $\boldsymbol{x}_n \in \mathbb{R}^{100}, \forall n = 1, \ldots, 100$, i.e., we consider a compression of 55% measurements for each vector $\boldsymbol{x}_n$. We randomly generate 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. Notice that 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 $\boldsymbol{y}_n = A\boldsymbol{x}_n + \boldsymbol{e}_n$ with SNR=25 dB, where $\boldsymbol{e}$ is a Gaussian noise accounting for the measurement noise. The above setting for the image reconstruction problem using the compressed sensing and solving the inverse SMV problem via random Gaussian projections and Gaussian noise in the measurements follows some of the other recent work in this area [87–91]. We feed all the algorithms with the measurement vector $\boldsymbol{y}$ and the same sensing matrix A. Also, the generated measurement noise matrix is the same for the digits. Once $\hat{\boldsymbol{x}}_n, \forall n = 1, \ldots, 100$ is solved, we collect the results and then stack them all together and reconstruct the digits. For the purpose of demonstrating the support recovery performance, in Fig. 2.11 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 $\boldsymbol{s}$, we performed the following. In the reconstruction, we set the estimated pixel values less than 0.3 to zero. 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 1. However, since the proposed C-SBL

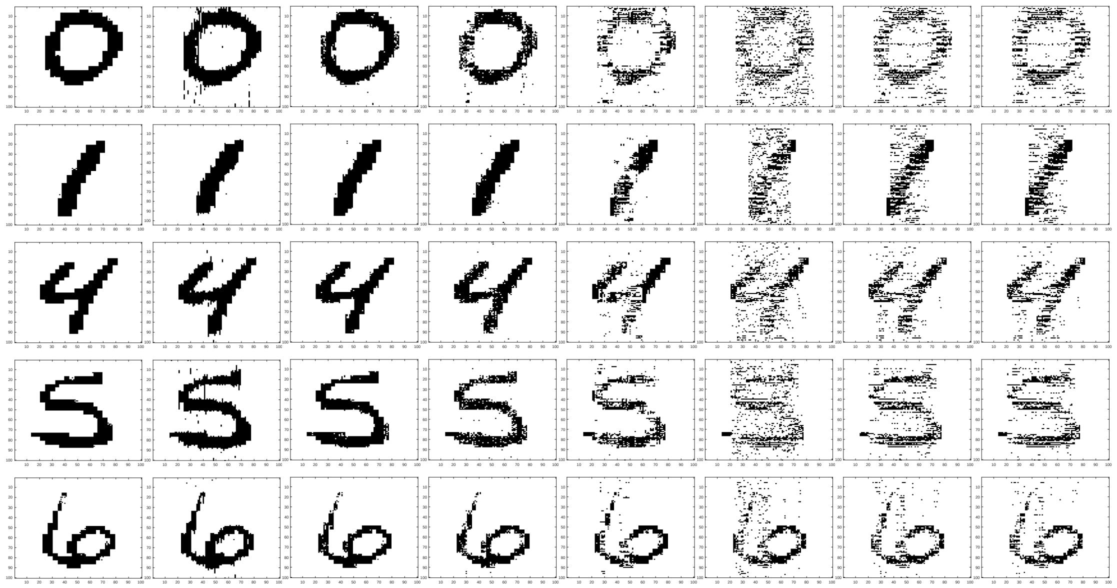


Fig. 2.11: Results of reconstructed images for the SMV case. The first column on the left illustrates the non-zero locations of the true hand-written digits. The other columns from left to right show the performance in terms of supports when using C-SBL, BSBL (h=4), PCSBL, CLUSSMCMC, MTCS, MFOCUSS, and BPDN algorithms, respectively.

algorithm already can provide the estimates on the active locations via $\boldsymbol{s}$, the thresholding process is not required.

The first column of images in Fig. 2.11 shows the true hand-written digits, and the other columns show the results of processing with different algorithms.

We study the performance of the C-SBL algorithm against the other algorithms in the reconstruction of the images via both the support and pattern recovery. In Table 2.1 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 2.2, 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 (2.7). We do the same procedure for computing the estimated measure of clumpiness in the reconstructed digits and provide the results in Table 2.2. According to the results shown in Table 2.1, the best results in support recovery belongs to C-SBL algorithm.

Table 2.1: SMV case: Comparing the reconstruction performance in terms of $P_D - P_{FA}$ for digits **0,1,4,5,6**.

Algorithm	**Digit 0**	**Digit 1**	**Digit 4**	**Digit 5**	**Digit 6**
C-SBL	**0.9530**	**0.9954**	**0.9834**	**0.9690**	**0.9847**
BSBL	0.9204	0.9819	0.9617	0.9479	0.9116
PCSBL	0.7622	0.9544	0.8961	0.8270	0.8468
CLUSS	0.4265	0.6689	0.6421	0.5878	0.7179
MTCS	0.3030	0.4828	0.4779	0.4074	0.5989
MFOCUSS	0.3652	0.6510	0.5197	0.4961	0.5734
BPDN	0.3701	0.6360	0.5212	0.4967	0.5859

Table 2.2: SMV case: Comparing the performance in terms of learning the clustering pattern via the measure of clumpinesss ($\Sigma\Delta$) for the reconstructed digits **0,1,4,5,6**. The true ($\Sigma\Delta$) for these digits are **208,80,166,316,** and **220**, respectively.

Algorithm	**Digit 0**	**Digit 1**	**Digit 4**	**Digit 5**	**Digit 6**
C-SBL	**244**	**78**	**176**	**324**	**196**
BSBL	320	84	200	344	262
PCSBL	634	126	346	632	344
CLUSS	1018	370	570	804	446
MTCS	1774	1022	1194	1496	760
MFOCUSS	1532	878	936	1220	684
BPDN	1478	818	908	1192	646

Also, Table 2.2 shows that C-SBL was more successful in learning the underlying clustering pattern of the digits.

Performance for the MMV Case

In this set of simulations we consider the same reconstruction problem of hand-written digits. The sensing matrix and noise are generated the same as before. We end up with an MMV problem where $X \in \mathbb{R}^{100\times100}$ and the measurement matrix is $Y \in \mathbb{R}^{55\times100}$. Looking at the non-zero locations (shown by black pixels) in the first column of Fig. 2.12, one can think of this problem as an MMV problem where the support set changes slowly across the columns of the true solution. In Fig. 2.12 from left to right, we illustrate the supports of the true images, and the reconstructed images for C-SBL, TMSBL, MSBL, MTCS, and MFOUCSS, respectively. Notice that both C-SBL and MSBL assume that the supports of the true solution are the same for all the columns of each solution. Therefore, instead of solving an MMV with $X \in \mathbb{R}^{100\times100}$ for C-SBL and MSBL, we solve 50 MMVs where of size $X_n \in \mathbb{R}^{100\times2}, n = 1, \ldots, 50$. Since neither C-SBL nor MSBL for support change structure, the false alarm rate for such algorithms may be increased. This means that we assume that every two columns of the solution we seek have the same support set. Therefore in C-SBL and MSBL, instead of solving one big MMV problem simultaneously, we treat the MMV as a collection of small MMVs. For the other algorithms, we show the results for the case where the full MMV (with $N = 100$) was solved.

According to the reconstructed images obtained in Fig. 2.12, we observe that C-SBL provided the best results. In Tables 2.3–2.4 we also show the results in terms of support recovery and estimated measure of clumpiness for the true digits.

2.5 Convergence Diagnostic of MCMC Implementation

Convergence is an important issue in order to determine the burn-in period of MCMC algorithms [68]. 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) [92, 93]. The term PSRF measures whether the approximated distribution for a

Table 2.3: MMV case: Comparing the reconstruction performance in terms of $P_D - P_{FA}$ for digits **0,1,4,5,6**.

Algorithm	**Digit 0**	**Digit 1**	**Digit 4**	**Digit 5**	**Digit 6**
C-SBL	**0.9174**	**0.9822**	**0.9827**	**0.9762**	**0.9528**
TMSBL	0.3439	0.5571	0.5218	0.4431	0.6147
MSBL	0.3688	0.6241	0.5515	0.4913	0.6497
MTCS	0.1776	0.2642	0.3192	0.3730	0.3795
MFOCUSS	0.3778	0.6517	0.5490	0.5282	0.6202

Table 2.4: MMV case: Comparing the performance in terms of learning the clustering pattern via the measure of clumpinesss ($\Sigma\Delta$) for the reconstructed digits **0,1,4,5,6**. The true ($\Sigma\Delta$) for these digits are **208,80,166,316,** and **220**, respectively.

Algorithm	**Digit 0**	**Digit 1**	**Digit 4**	**Digit 5**	**Digit 6**
C-SBL	**274**	**80**	**176**	**324**	**200**
TMSBL	1810	1054	1166	1568	752
MSBL	1630	796	1034	1360	650
MTCS	1228	688	834	1022	576
MFOCUSS	1522	806	936	1210	636

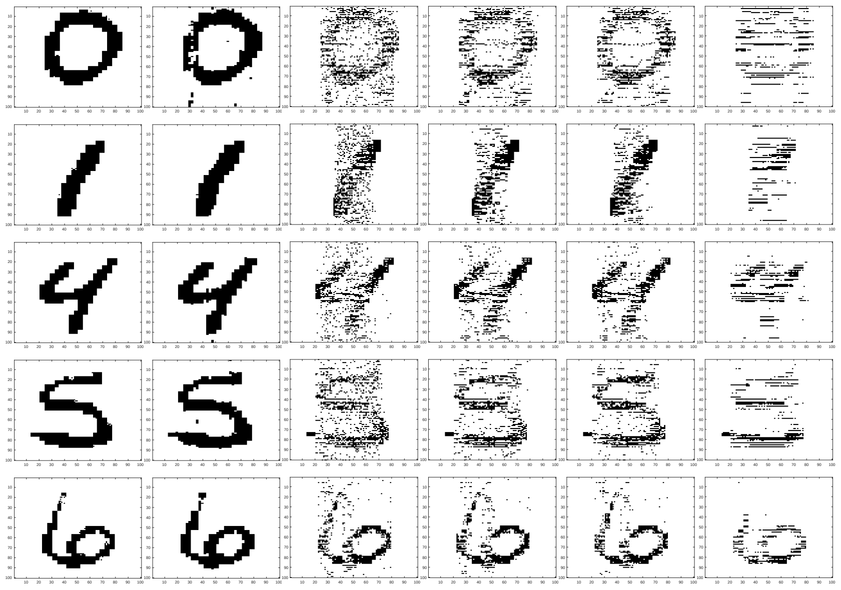
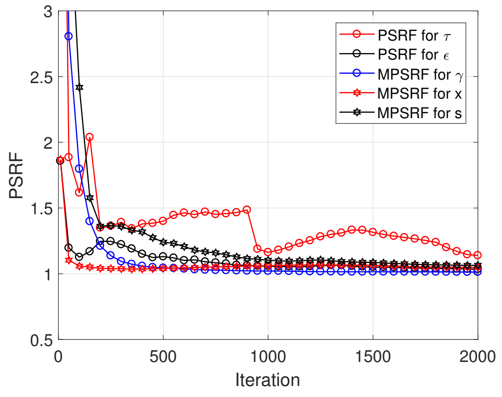


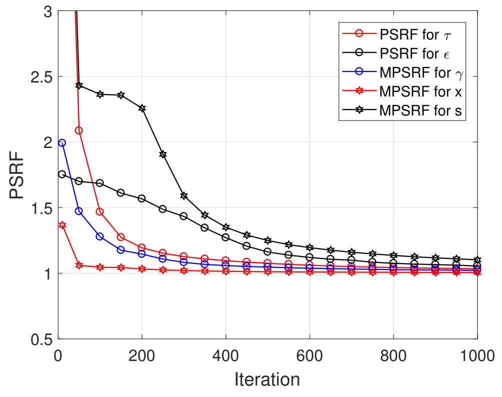
Fig. 2.12: Results of reconstructed images for the MMV case. The first column on the left illustrates the non-zero locations of the true hand-written digits. The other columns from left to right show the performance in terms of support locations when using C-SBL, TMSBL, MSBL, MTCS, and MFOCUSS algorithms, respectively.

variable of interest, represented by C chains, has converged to the target posterior distribution. The closer the two distributions are, the closer PSRF will become to 1. For example in [68], the convergence issue of CLUSSMCMC 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 Fig. 2.13, 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 $N = 100$, sparsity level of 25, and with the number of measurements $M =$ 20, 50, and 90.

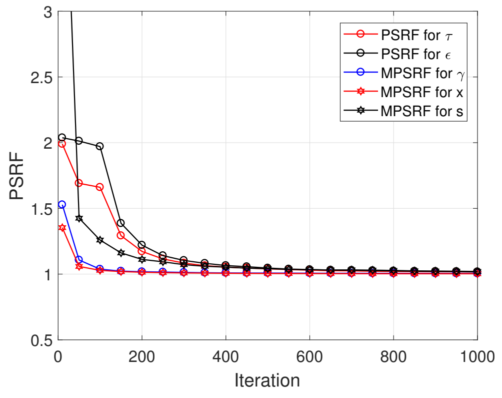
In Fig. 2.13(a), we demonstrate the measure of PSRF and MPSRF for the variables and parameters of interest in our model for an example with the low sampling ratio of 0.2. According to this plot, the Gibbs sampler converges very quickly for ε, γ, and $\boldsymbol{x}$, i.e., the convergence measure of PSRF became close to 1 with few iterations. However, the convergence of the distribution on $\boldsymbol{s}$ is slower. The convergence of the precision on X was the slowest. According to Gelman's discussion in [94], one may prefer to set the burn-in



(a) Example with $M = 20$



(b) Example with $M = 50$



(c) Example with $M = 90$

Fig. 2.13: 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 $\boldsymbol{x}$, the support vector $\boldsymbol{s}$, and the mixing-coefficient vector $\boldsymbol{\gamma}$ for sampling ratios of 0.2, 0.5, and 0.9.

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 Fig. 2.13(a). In Fig. 2.13(b), 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 is 0.2. For this example, a burn-in period of around 600 is satisfactory. Fig. 2.13(c) 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 Figs. 2.7 – 2.10 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 [94]. Also, we monitored the average number of elapsed iterations until the variations on the outcomes of the estimated $\boldsymbol{s}$ 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 [94]. More specifically, we monitored the posterior distribution on the support learning vector $\boldsymbol{s}$, using (2.6) for O-SBL and (2.14) for C-SBL, based on the samples obtained from MCMC implementation. In Fig. 2.14, we illustrate some examples of support learning vector $\boldsymbol{s}$ using the C-SBL algorithm with the number of measurements $M = 55$. Each plot shows the learning process of $\boldsymbol{s} \in \mathbb{R}^{100}$, represented by the samples drawn from (2.14), as a function of the number of iterations. Using the experimental results based on both the PSRF evaluation and monitoring the outcomes of $\boldsymbol{s}$, we then set fixed burn-in periods for the simulations illustrated in Section 2.4. 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 2.4. 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 where the burn-in period is required to be more than what we set. The effect of the need for longer burn-in period can be observed in the results of Fig. 2.7, Fig. 2.9, and Fig. 2.10 for low sampling ratios. The convergence diagnostic and the effect of burn-in period can also be detected in Fig. 2.6 and Fig. 2.8. It should be clear from Fig. 2.6(c) 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 $\boldsymbol{s}$ has already been stabilized. However, we see a different behavior for lower sampling ratios in Fig. 2.6(c). The posterior distribution on $\boldsymbol{s}$ is Bernoulli and the variations on the supports in the iterative samples directly affects the performance in the support recovery. We see in Fig. 2.6(c) that on average the 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 the computing the sample mean of the collected samples.

It worth noting that computing the MSPRF for $\boldsymbol{s}$ needed some modifications. The estimated posterior variance of $\boldsymbol{s}$ is assessed based on the mixture of all the simulated sequences divided by the average of the variances within each sequence [94]. The main issue with the MPSRF for $\boldsymbol{s}$ 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 $\boldsymbol{s}$ were either zero or 1. We dealt with this issue by adding random draws from a zero-mean Gaussian with the variance of 10^{-8} to the samples of $\boldsymbol{s}$ and then measured the MPSRF.

2.6 Summary

The sparse recovery of signals via SMV and MMV models was considered. The O-SBL

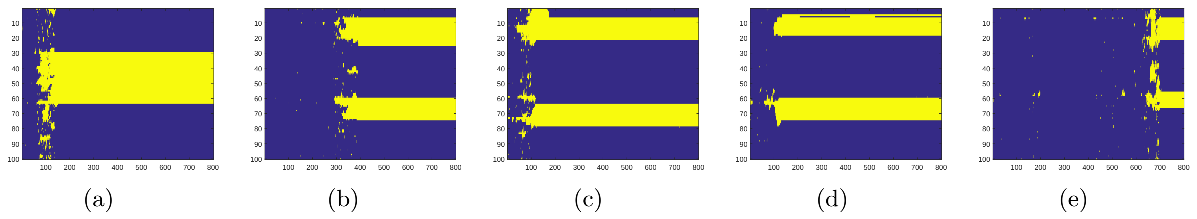


Fig. 2.14: Examples showing samples of the support learning vector $\boldsymbol{s}$. The vertical axis shows the elements of $\boldsymbol{s}$ and the horizontal axis represents the iterations.

algorithm simultaneously learns both the supports and solution-value matrix for the MMVs with the joint sparsity structure. 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 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, it is known that MCMC implementation is computationally expensive. In future work we will consider the alternative approaches to MCMC implementation such as variational Bayes inference method.

2.7 Appendix

In this section we provide the details on the update rules for O-SBL and C-SBL algorithm.

2.7.1 Details on the Update Rules of O-SBL

In this section, we provide the inference on the full posterior distributions for the model parameters to obtain approximations for the probability distributions of $\mathbf{s}$ and X for O-SBL algorithm. According to the graphical model of the O-SBL algorithm illustrated in Fig. 2.2, and the equations (2.1–2.6), the marginalized posterior distributions for the variables and parameters of the model can be obtained from the below descriptions.

- Posterior of the support-learning component:

Below we describe the derivation of the inference equation for s_p provided in (2.6). According to the joint probability distribution (2.5), we have, $\forall p = 1, \dots, P$,

$$p(\mathbf{s}|-) \sim p(Y|\mathbf{s}, X)p(\mathbf{s}|\gamma_p), \tag{2.23}$$

where

$$p(s_p|\gamma_p) \sim \text{Bernoulli}(\gamma_p), \ \ \forall p = 1, 2, ..., P. \tag{2.24}$$

Therefore,

$$P(s_p|\gamma_p) \propto \gamma_p^{s_p}(1-\gamma_p)^{1-s_p}, \ \ \forall p = 1, 2, ..., P$$

and

$$p(Y|\mathbf{s}, X, \boldsymbol{\mu}_{x_s}, \Sigma_{x_s}) \propto \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{s}, \mathbf{x}_n, \mu_{x_s}, \Sigma_{x_s}).$$

By expanding the above likelihood, we then have

$$\log\{p(Y|\mathbf{s}, X, \boldsymbol{\mu}_{x_s}, \Sigma_{x_s})\} \propto \varepsilon\big((y_{11} - \sum_{k=1}^{P} a_{1k}s_p x_{k1})^T(\star) + ... + (y_{m1} - \sum_{k=1}^{P} a_{mk}s_p x_{k1})^T(\star)) + ...$$

$$+((y_{1n} - \sum_{k=1}^{P} a_{1k}s_p x_{kn})^T(\star) + .. + (y_{mn} - \sum_{k=1}^{p} a_{mk}s_p x_{kn})^T(\star)\big),$$

where $(B)^T A(B)$ is denoted by $(B)^T A(\star)$. Equivalently,

$$\log\{p(Y|s_p, X, \boldsymbol{\mu}_{x_s}, \Sigma_{x_s})\} \propto$$

$$\varepsilon\big((a_{1k}s_p x_{k1} - (y_{11} - \sum_{l\neq k}^{p} a_{1l}s_l x_{l1}))^T(\star) + ... + (a_{mk}s_p x_{k1} - (y_{m1} - \sum_{l\neq k}^{P} a_{ml}s_l x_{l1}))^T(\star)) + ...$$

$$((a_{1k}s_p x_{kn} - (y_{1n} - \sum_{l\neq k}^{P} a_{1l}s_l x_{ln}))^T(\star) + ... + (a_{mk}s_p x_{kn} - (y_{mn} - \sum_{l\neq k}^{p} a_{ml}s_l x_{ln}))^T(\star)\big).$$

Let's define

$$\tilde{y}_{ji}^{-k} \overset{\text{def}}{=} y_{ji} - \sum_{l \neq k}^{p} a_{jl} s_l x_{li}, \ \ \forall j = 1, 2, ..., M, \ \ \forall i = 1, 2, ..., N, \ \ \forall k = 1, 2, ..., P. \tag{2.25}$$

Therefore,

$$\log\{p(Y|s_p, X, \boldsymbol{\mu}_{x_s}, \Sigma_{x_s})\} \propto$$
$$\varepsilon\big(((x_{k1}^2 \sum_{i=1}^{M} a_{ik}^2)s_p^2 - 2(x_{k1}\sum_{i=1}^{M} a_{ik}\tilde{y}_{i1}^{-k})s_p) + ... + ((x_{kn}^2 \sum_{i=1}^{m} a_{ik}^2)s_p^2 - 2(x_{kn}\sum_{i=1}^{M} a_{ik}\tilde{y}_{in}^{-k})s_p)\big),$$

Resulting in

$$\log\{p(Y|s_p, X, \boldsymbol{\mu}_{x_s}, \Sigma_{x_s})\} \propto \varepsilon\big((\|\mathbf{a}_k\|_2^2 \sum_{i=1}^{N} x_{ki}^2)s_p^2 - 2(x_{k1}\sum_{i=1}^{M} a_{ik}\tilde{y}_{i1}^{-k} + ... + x_{kn}\sum_{i=1}^{M} a_{ik}\tilde{y}_{in}^{-k})s_p\big). \tag{2.26}$$

Let's also define

$$\mathbf{a}_k^T \tilde{\mathbf{y}}_1^{-k} \overset{\text{def}}{=} \sum_{i=1}^{M} a_{ik}\tilde{y}_{i1}^{-k}, \tag{2.27}$$

where $\tilde{\mathbf{y}}_1^{-k} = [\tilde{y}_{11}^{-k}, ..., \tilde{y}_{m1}^{-k}]^T$.

Substitution of equation (2.27) into equation (2.26) yields

$$\log\{p(Y|s_p, X, \boldsymbol{\mu}_{x_s}, \Sigma_{x_s})\} \propto \varepsilon((\|\mathbf{a}_k\|_2^2 \sum_{i=1}^{N} x_{ki}^2)s_p^2 - 2(\mathbf{a}_k^T(\sum_{i=1}^{N} x_{ki}\tilde{\mathbf{y}}_i^{-k}))s_p).$$

Finally, we obtain the following results

$$p(s_p|-) \sim \text{Bernoulli}(\frac{q_1}{q_0 + q_1}), \ \ \forall p = 1, 2, ..., P, \tag{2.28}$$

where

$$q_0 \overset{\text{def}}{=} 1 - \gamma_p \tag{2.29}$$

and

$$q_1 \overset{\text{def}}{=} \gamma_p e^{-\frac{\varepsilon}{2}((\|\mathbf{a}_k\|_2^2 \sum_{i=1}^{N} x_{ki}^2) - 2(\mathbf{a}_k^T(\sum_{i=1}^{N} x_{ki}\tilde{\mathbf{y}}_i^{-k})))}. \tag{2.30}$$

- Posterior of γ_p:

The posterior γ_p can be obtained from the relationship below.

$$p(\gamma_p|-) \propto p(s_p|\gamma_p)p(\gamma_p|\alpha_0, \beta_0), \ \ \forall p = 1, 2, ..., P, \tag{2.31}$$

where

$$p(\gamma_p|\alpha_0, \beta_0) \propto (\gamma_p)^{\alpha_0 - 1}(1-\gamma_p)^{\beta_0 - 1}, \tag{2.32}$$

which yields to

$$p(\gamma_p|-) \propto (\gamma_p)^{(\alpha_0 + s_p)-1}(1-\gamma_p)^{(\beta_0 + 1 - s_p)-1}. \tag{2.33}$$

Thus

$$p(\gamma_p|-) \sim \text{Beta}(\alpha_0 + s_p \ , \ \beta_0 + 1 - s_p), \ \ \forall p = 1, 2, ..., P. \tag{2.34}$$

- Posterior of the solution-value matrix X:

Below the equation for finding the posterior x_{LK} is represented.

$$p(x_{LK}|-) \propto p(Y|\mathbf{s}, x_{LK}, \mu_{x_{LK}}, \Sigma_{x_{LK}})p(x_{LK}|X, \mu_{x_{LK}}, \tau^{-1}), \tag{2.35}$$

where

$$p(x_{LK}|X, \mu_{x_{LK}}, \tau^{-1}) \propto e^{\frac{-\tau}{2}x_{LK}^2} \ \ \forall L = 1, 2, ..., P, \ \ \forall K = 1, 2, ..., N \tag{2.36}$$

and

$$\log\{p(Y|\mathbf{s}, x_{LK}, \mu_{x_{LK}}, \Sigma_{x_{LK}})\} \propto -\frac{\varepsilon}{2}\big((y_{1k} - \sum_{l=1}^{P} a_{1l}s_l x_{lk})^T(\star) + ... + (y_{mk} - \sum_{l=1}^{p} a_{ml}s_l x_{lk})^T(\star)\big), \tag{2.37}$$

which

$$\begin{gathered}\log\{p(x_{LK}|-)\} \propto \\ -\frac{1}{2}\big((\tau + \varepsilon s_L^2 \sum_{i=1}^{M} a_{iL}^2)x_{LK}^2 - 2(\varepsilon s_L \sum_{i=1}^{M} a_{iL}\tilde{y}_{iK}^{-L})x_{LK}\big), \forall L = 1, ..., P, \forall K = 1, ..., N.\end{gathered} \tag{2.38}$$

Therefore,

$$p(x_{LK}|-) \sim \mathcal{N}(\mu_{x_{LK}}, \Sigma_{x_{LK}}), \ \ \forall L = 1, 2, ..., P, \ \ \forall K = 1, 2, ..., N, \tag{2.39}$$

where

$$\mu_{x_{LK}} \overset{\text{def}}{=} \varepsilon s_L \Sigma_{x_{LK}} \mathbf{a}_L^T \tilde{\mathbf{y}}_K^{-L}$$

and

$$\Sigma_{x_{LK}} \overset{\text{def}}{=} \big(\tau + \varepsilon(s_L \left\|\mathbf{a}_L\right\|_2)^2\big)^{-1}.$$

- Posterior of τ

$$p(\tau|-) \propto \big(\prod_{i=1}^{P} p(\mathbf{x}_i|\boldsymbol{\mu}_{\mathbf{x}_i}, T^{-1})\big)p(\tau|a_0, b_0), \tag{2.40}$$

where $T = \tau I_P$. Also,

$$p(\tau|a_0, b_0) \propto \tau^{a_0-1} e^{-b_0\tau} I_{[0,\infty)}(\tau) \tag{2.41}$$

and

$$\begin{aligned} \prod_{i=1}^{P} p(\mathbf{x}_i|\boldsymbol{\mu}_{\mathbf{x}_i}, T^{-1}) &\propto \tau^{\frac{NP}{2}} e^{-\frac{\tau}{2}\sum_{i=1}^{N}\mathbf{x}_i^T\mathbf{x}_i} \\ &\propto \tau^{\frac{NP}{2}} e^{-\frac{\tau}{2}\|X\|_F^2}, \end{aligned} \tag{2.42}$$

where $\|.\|_F$ denotes Frobenius norm. By substituting equation (2.41) and equation (2.42) into equation (2.40), we will then have

$$p(\tau|-) \propto \tau^{(a_0+\frac{NP}{2})-1} e^{-(b_0+\frac{1}{2}\|X\|_F^2)\tau} I_{[0,\infty)}(\tau) \tag{2.43}$$

or equivalently,

$$p(\tau|-) \sim \text{Gamma}(a_0 + \frac{NP}{2}\ , \ b_0 + \frac{1}{2}\left\|X\right\|_F^2). \tag{2.44}$$

- Posterior of the noise precision ε:

$$p(\varepsilon|-) \propto p(Y|\mathbf{s},X)p(\varepsilon|\theta_0,\theta_0), \tag{2.45}$$

where

$$p(\varepsilon|\theta_0,\theta_1) \propto \varepsilon^{\theta_0-1}e^{-\varepsilon\theta_1}$$

and

$$p(Y|\mathbf{s},X) \propto \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{s},\mathbf{x}_n), \tag{2.46}$$

where

$$p(\mathbf{y}_n|\mathbf{s},\mathbf{x}_n) \propto \varepsilon^{\frac{M}{2}}e^{-\frac{1}{2}\varepsilon\|\mathbf{y}_n-A(\mathbf{s}\circ\mathbf{x}_n)\|_2^2},\ \forall n=1,2,...,N. \tag{2.47}$$

Therefore,

$$p(Y|\mathbf{s},X) \propto \varepsilon^{\frac{MN}{2}}e^{-\frac{1}{2}\varepsilon\|Y-A(\mathbf{s}\circ X)\|_F^2}. \tag{2.48}$$

Finally, we can substitute equation (2.47) and equation (2.46) into equation (2.45) and reach to the result below.

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

In other words,

$$p(\varepsilon|-) \sim \text{Gamma}\big(\theta_0+\frac{MN}{2}\ ,\ \theta_1+\frac{1}{2}\|Y-A(\mathbf{s}\circ X)\|_F^2\big). \tag{2.49}$$

CHAPTER 3

ON THE BLOCK-SPARSE SOLUTION OF SINGLE MEASUREMENT VECTORS

3.1 Introduction

Finding the solution of single measurement vector (SMV) problem with an unknown block-sparsity structure is considered. Here, we propose a sparse Bayesian learning (SBL) algorithm simplified via the approximate message passing (AMP) framework. In order to encourage the block-sparsity structure, we incorporate a parameter called Sigma-Delta as a measure of clumpiness in the supports of the solution. Using the AMP framework reduces the computational load of the proposed SBL algorithm and as a result makes it faster in terms of the execution time. Furthermore, in terms of the mean-squared error between the true and the reconstructed solution, the algorithm demonstrates an encouraging improvement compared to the other algorithms.

The SMV problem is a set of linear noisy measurements from a sparse signal $\boldsymbol{x}$ and is modeled as $\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$, where $\boldsymbol{x} \in \mathbb{R}^N$ is the signal of interest to be reconstructed and $\boldsymbol{e}$ denotes the noise. In this model, $A \in \mathbb{R}^{M\times N}$ is a known sensing matrix with $M \ll N$. In some practical applications the non-zero entries of the sparse signal $\boldsymbol{x}$ appear in clusters, so they more or less clump together. This feature has been referred to as block-sparsity in the literature [20, 37, 38, 43, 51, 58]. One example is magnetoencephalography, which seeks the locations where most brain activities are produced. Such activities exhibit contiguity i.e., they occur in localized regions [58].

For the purpose of promoting the recovery performance of the SMV problem with an unknown block-sparsity framework, Zhang and Rao proposed a sparse Bayesian learning (SBL) algorithm that incorporates intra-block correlation (correlation structure in each block) [51]. In this model, they defined a Gaussian-distributed prior with zero-mean and the covariance that depends on the multiplication of a nonnegative scaling parameter followed by a covariance matrix for each block. Such blocks are assumed to be uncorrelated with each other. In order to simplify the model, reduce the complexity, and suppress the overfitting of the parameters in the model, they further considered the same underlying covariance matrix with different scaling parameters for the blocks as a prior. This matrix is updated via the expectation-maximization (EM) algorithm. In [37], a hierarchical Bayesian approach was proposed to deal with the block-sparse multiple measurement vectors (MMVs) problem. In this case, a prior is incorporated which encourages both contiguity and sparsity in the solution. This prior is based on the parameter referred to as Sigma-Delta ($\Sigma\Delta$), which is a measure of contiguity over the supports of the solution. The Sigma-Delta parameter is defined as follows

$$(\Sigma\Delta)_{\boldsymbol{s}} = \sum_{n=2}^{N} |s_n - s_{n-1}|, \tag{3.1}$$

where $\boldsymbol{s}$ is the support learning vector of the solution and has binary values with "1" denoting the active entry of $\boldsymbol{x}$. The prior on $\boldsymbol{s}$ depends on the term $e^{-\alpha(\Sigma\Delta)_{\boldsymbol{s}}}$ where $\alpha > 0$. The larger the weight α is, the more clumpy the solution becomes. Experimental receiver operating curves (ROC) indicate that the performance of the algorithm was satisfactory and encouraging. However, the runtime of the algorithm was not fast. In [38], we proposed the SDsRandOMP algorithm which is essentially a sparse version of RandOMP [95] and incorporates the Sigma-Delta parameter to encourage the block-sparsity. This algorithm is almost greedy-based in which the behavior of Sigma-Delta is modeled by a Gamma distribution with $(\Sigma\Delta)_{\boldsymbol{s}} \sim \Gamma(1, \theta)$. Although this algorithm is faster than the SBLs, it requires more information i.e., noise variance.

Recently, research in this area has turned to reducing the computational complexity of their algorithms while preserving the success in the support recovery at a high rate. Such work includes using approximate message passing (AMP) and the expectation-maximization (EM) for the hyperparameters of the SBL algorithms [36, 96, 97]. The AMP is essentially a simplified version of the message passing where the number of messages to be propagated is reduced based on Taylor series approximation, averaging over the messages passed to the same node, and the central limit theorem [36, 96]. In this case, Al-Shoukairi and Rao incorporated the AMP to their earlier SBL model for both the SMV and MMV problem to reduce the runtime of their algorithms [36]. They avoided imposing a Bernoulli-Gaussian prior on the solution vector and instead simply used a zero-mean Gaussian prior. This simplification causes all the distributions on the factor graph to become Gaussian and results in fewer computations when using message passing.

In this chapter, we propose a new SBL algorithm for the block-sparse SMV problem based on the AMP-SBL algorithm proposed in [36] . For this purpose, we use our measure of clumpiness over the supports of the solution (Sigma-Delta) proposed in [37,38] to the AMP-SBL. We refer to our new algorithm as AMP-B-SBL where "-B-" denotes the block-sparsity. To demonstrate the performance of our algorithm, we compare the proposed algorithm with two other algorithms in terms of the normalized mean-squared error between the true and the reconstructed solution. Furthermore, to show the behavior of the algorithm in the signal reconstruction, a random block-sparse case scenario for the SMV has been made. Since it benefits from the AMP, this algorithm turns out to be faster than our previously proposed SBLs [37, 38].

3.2 AMP-B-SBL for Solving SMV

In order to solve for the SMV problem with an unknown block-sparsity structure, we combine our measure of clumpiness of the solution (Sigma-Delta) proposed in [37, 38] with the AMP-SBL algorithm introduced in [36]. Here, we define the Sigma-Delta as

$$(\Sigma\Delta)_{(\text{support of } \boldsymbol{x})} = \sum_{n=2}^{N} |b(x_n, T) - b(x_{n-1}, T)|, \tag{3.2}$$

where T is a predetermined threshold. The function $b(.,.)$ in (3.2) returns a binary value and is defined as follows

$$b(x_n, T) = \begin{cases} 1 & \text{if } |x_n| > T \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

In our previous SBL model [37] we did not have such soft thresholding (3.3), and regardless

Table 3.1: Behavior of α_n with respect to Sigma-Delta

$(\Sigma\Delta)\|_{b(x_n,.)=1}$	$(\Sigma\Delta)\|_{b(x_n,.)=0}$	α_n
cte	*cte*	$\downarrow$
$\uparrow$	*cte*	$\downarrow$
cte	$\downarrow$	$\downarrow$
cte	$\uparrow$	$\uparrow$
$\downarrow$	*cte*	$\uparrow$

of having a very small or large value on x_n, the corresponding support s_n became active ($s_n = 1$). In contrast, here we simply discard such small values by setting the corresponding $s_n = 0$ because such x_p does not have considerable contribution in our measurements. In order not to discard important portions of the signal of interest, we set the threshold T to a small value.

Based on the assumption that the solution vector $\boldsymbol{x}$ is sparse, we consider an i.i.d. zero-mean Gaussian prior on the components of $\boldsymbol{x}$. The variance on x_n is defined as α_n which accounts for learning the block-sparsity structure in the solution. These distributions are defined below.

$$\begin{aligned} &\forall n = 1, \ldots, N, \quad x_n \sim \mathcal{N}(0, \alpha_n) \\ &\alpha_n \sim \mathcal{N}\big(e^{\left\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0} - (\Sigma\Delta)|_{b(x_n,.)=1} - 1}{\theta_1}\right\}}, \theta_2\big), \end{aligned} \tag{3.4}$$

where $(\Sigma\Delta)|_{b(x_n,.)=1}$ is the Sigma-Delta evaluation of the supports of $\boldsymbol{x}$ in case where s_n is forced to be active, θ_1 is the emphasizing parameter on our measure of clumpiness, and θ_2 is the prior variance on the variance of the variable x_n. The rationale behind assuming this prior on α_n is described in Table 3.1, in which *cte* denotes a constant value. As an example of Table 3.1, consider the case where forcing either $s_n = 0$ or $s_n = 1$ does not make any change in the evaluation of Sigma-Delta. In this case, though it promotes the clumpiness in the solution, it discourages the solution to be sparse as we do this a couple of times. Therefore, α_n needs to be decreased.

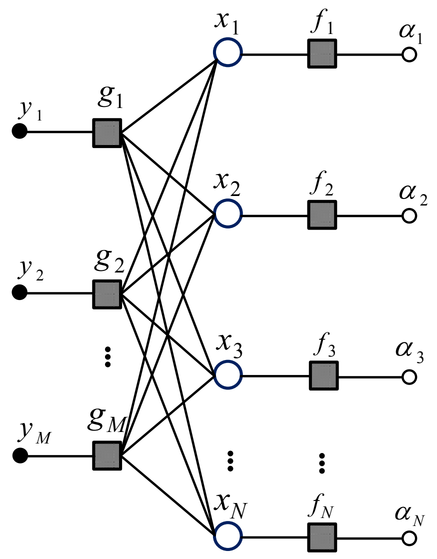


Fig. 3.1: Factor graph for B-SMV.

According to the defined prior distributions in (3.4), the joint probability distribution of our model becomes

$$p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\alpha}, \theta_1, \theta_2, \sigma^2) \propto p(\boldsymbol{y}|\boldsymbol{x}, \sigma^2 I_N) \prod_{n=1}^{N} \left(p(x_n; 0, \alpha_n) p(\alpha_n; e^{\left\{ \frac{(\Sigma\Delta)|_{b(x_n,.)=0} - (\Sigma\Delta)|_{b(x_n,.)=1} - 1}{\theta_1}, \right\}}, \theta_2) \right), \tag{3.5}$$

where the measurement noise is assumed to be $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 I_N)$. Under this model, all the distributions of the joint, conditional, and posterior densities become Gaussian. Therefore, we only need to pass the mean and variance of the messages rather than computing and propagating the full actual messages. The factor graph of our model for the block-sparse SMV (B-SMV) problem is illustrated in Fig. 3.1. In the graphical model Fig. 3.1, the large circle nodes represent the random variables of interest, the shaded squares show the function nodes, the small shaded circles are the observations, and the small unshaded circles denote the hyperparameters [74]. Since this main layer of our algorithm is the same as the one introduced in [36, 97], here we use the same notations as [36]. Function nodes shown in Fig.

3.1 are defined as follows

$$\begin{aligned} g_m &:= p(y_m|\boldsymbol{x}, \boldsymbol{\alpha}), \quad m = 1, 2, \dots, M \\ f_n &:= p(x_n; 0, \alpha_n), \quad n = 1, 2, \dots, N. \end{aligned} \tag{3.6}$$

Notice that our proposed algorithm adds an additional layer to [36] by incorporating the prior (3.4) in order to encourage the block-sparsity. This change in the model does not appear in the factor graph Fig. 3.1. We now describe our reasoning behind the proposed model for the B-SMV problem. As a prior knowledge, we expect the solution vector $\boldsymbol{x}f$ to be sparse. Therefore, in order to encourage the sparsity, we assume a zero-mean Gaussian distribution for $\boldsymbol{x}$ with the variance α_n, $n = 1, 2, ..., N$ on its components. The supports of the solution are then specified by the binary function $b(., T)$ where T is a predetermined threshold. In other words, based on the threshold we discard the small-valued components of $\boldsymbol{x}$ from being considered as the support of the solution. The smaller α_n is, the higher probability it provides to x_n removed from the supports. Based on our previous results in [37, 38], we observed that having more clumpy supports in the solution causes higher value of α and vice versa. Therefore, here we made the mean of our Gaussian prior on α_n be proportional to the term $\exp\left\{(\Sigma\Delta)|_{b(x_n,T)=0} - (\Sigma\Delta)|_{b(x_n,.)=1}\right\}$.

Below, we represent the message that propagates from each function node to a variable node and vice versa of Fig. 3.1. The derived messages in (3.7)-(3.9) are the same as those derived in [36, 98].

- Message from a function node to a variable node:

$$M_{g_m \to x_n} \propto \mathcal{N}(a_{mn}x_n; z_{mn}, c_{mn}), \tag{3.7}$$

where

$$z_{mn} = y_m - \sum_{q \neq n} a_{mq}\mu_q \quad \text{and} \quad c_{mn} = \sigma^2 + \sum_{q \neq n} |a_{mq}|^2 \Sigma_q.$$

- Message from a variable node to a function node:

$$M_{x_n \to g_m} \propto \mathcal{N}\big(x_n; \sum_{l \neq m} \frac{a_{nl} z_{ln} \alpha_n}{c_n + \alpha_n \sum_{l \neq m} a_{ln}^2}, \frac{c_n \alpha_n}{c_n + \alpha_n \sum_{l \neq m} a_{ln}^2}\big),$$

where c_{nl} (under the large-system-limit) is approximated by [98]

$$c_{nl} \simeq c_n := \frac{1}{M} \sum_{m=1}^{M} c_{mn}.$$

Using the fact that we have already normalized the sensing matrix A with respect to its columns, we can further approximate

$$\sum_{l \neq m} a_{ln}^2 \simeq \sum_{m=1}^{M} a_{mn}^2 = 1.$$

Therefore,

$$M_{x_n \to g_m} \propto \mathcal{N}\big(x_n; \sum_{l \neq m} a_{nl} z_{ln} \frac{\alpha_n}{c_n + \alpha_n}, \frac{c_n \alpha_n}{c_n + \alpha_n}\big). \tag{3.8}$$

- Estimating the posterior on x_n:

$$p(x_n|\boldsymbol{y}) \propto p(x_n; \alpha_n) \prod_{m=1}^{M} p(y_m|x_n) \propto M_{f_n \to x_n} \prod_{m=1}^{M} M_{g_m \to x_n}.$$

After simplification, we obtain [36]

$$p(x_n|\boldsymbol{y}) \propto \mathcal{N}(x_n; \mu_n, \nu_n),$$

where

$$\mu_n = \sum_{m=1}^{M} a_{mn} z_{mn} \left(\frac{\alpha_n}{c_n + \alpha_n}\right), \quad \text{and} \quad \nu_n = \frac{c_n \alpha_n}{c_n + \alpha_n}. \tag{3.9}$$

Below, we provide the update rules for the hyperparameters of our algorithm using the expectation-maximization (EM) algorithm. As was mentioned earlier, since we add a layer to the AMP-SBL algorithm to account for the block-sparsity structure, the update rules become different from [36]. We start with describing the update rule for the variance α_n of x_n. Notice that as a prior we consider $\boldsymbol{x}$ is $x_n \sim \mathcal{N}(0, \alpha_n), \forall n = 1, \ldots, N$. Let us define

$$(\Sigma\Delta)_{n,k} := (\Sigma\Delta)|_{b(x_n,\cdot)=k}, \forall k = 0, 1. \tag{3.10}$$

- Update rule for α_n:

$$\alpha_n{}^{[k+1]} = \arg\min_{\alpha_n} \ln(\alpha_n) + \frac{1}{\alpha_n}\mathbb{E}_{\boldsymbol{x}|\alpha_n{}^{[k]},-}[x_n^2] + \frac{1}{\theta_2}\big(\alpha_n - e^{\{\frac{(\Sigma\Delta)_{n,0}-(\Sigma\Delta)_{n,1}-1}{\theta_1}\}}\big)^2.$$

Therefore,

$$\alpha_n{}^{[k+1]} = \arg\min_{\alpha_n} \ln(\alpha_n) + \frac{\mu_n^2 + \nu_n}{\alpha_n} + \frac{1}{\theta_2}\big(\alpha_n - e^{\{\frac{(\Sigma\Delta)_{n,0}-(\Sigma\Delta)_{n,1}-1}{\theta_1}\}}\big)^2. \tag{3.11}$$

Hence we solve for α_n in

$$\alpha_n^3 - e^{\left\{\frac{(\Sigma\Delta)_{n,0}-(\Sigma\Delta)_{n,0}-1}{\theta_1}\right\}}\alpha_n^2 + \frac{\theta_2}{2}\alpha_n - \frac{\theta_2}{2}(\mu_n^2 + \nu_n) = 0. \tag{3.12}$$

A solution among all the three possible roots for (3.12) which minimizes (3.11) is the update rule for α_n at the next iteration.

- Update rule for σ^2:

$$\begin{aligned}\sigma^{2[k+1]} &= \arg\max_{\sigma^2} \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\boldsymbol{\alpha},\sigma^{2[k]}}\left[p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\alpha}, \theta_1, \theta_2, \sigma^2)\right] \\ &= \arg\min_{\sigma^2} 2M\log\sigma + \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[\|\boldsymbol{y} - A\boldsymbol{x}\|_2^2\right]\end{aligned}$$

Therefore,

$$\begin{aligned}\sigma^{2[k+1]} &= \frac{\|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \mathrm{Tr}(\Sigma_{x|-}A^T A)}{M} \\ &= \frac{\|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \sum_{n=1}^N \|\boldsymbol{a}_n\|_2^2\Sigma_{x|-}}{M},\end{aligned} \tag{3.13}$$

where $\boldsymbol{\mu}_{x|-} := [\mu_1, \ldots, \mu_N]^T$ and $\Sigma_{x|-} := \text{diag}\{\nu_1, \ldots, \nu_N\}$, and $\Sigma_{x|-}$ is an $N \times N$ diagonal matrix.

- Update rule for θ_2:

$$\theta_2{}^{[k+1]} = \arg \min_{\theta_2} \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_2{}^{[k]},-} \big[N \ln(\theta_2) + \frac{1}{\theta_2} \sum_{n=1}^{N} (\alpha_n - e^{\{(\Sigma\Delta)_{n,0} - (\Sigma\Delta)_{n,1} - 1\}})^2\big].$$

  Therefore,

$$\theta_2{}^{[k+1]} = \frac{\sum_{n=1}^{N} (\alpha_n - e^{\{(\Sigma\Delta)_{n,0} - (\Sigma\Delta)_{n,1} - 1\}})^2}{N}. \tag{3.14}$$

Finally, based on the AMP algorithm in [36], our AMP-B-SBL is described as follows. The stopping condition of the algorithm can be based on a predetermined number of iterations or convergence of the solution to a tight bound.

AMP-B-SBL Algorithm for the block-sparse SMV problem:

- **Definitions**

$F_n(k_n, \alpha_n, c) = k_n \frac{\alpha_n}{c+\alpha_n}$

$G_n(\alpha_n, c) = \frac{c.\alpha_n}{c+\alpha_n}$

$F'_n(\alpha_n, c) = \frac{\alpha_n}{c+\alpha_n}$

- **Message updates**

```
For $n = 1, 2, \dots, N$
    $k_n = \sum_{m=1}^{m} a^{\star}_{mn} z_m + \mu_n$
    $\mu_n = F_n(k_n, \alpha_n, c)$
    $\nu_n = G_n(\alpha_n, c)$
End
$c = \sigma^2 + \frac{1}{M}\sum_{n=1}^{N} \nu_n$
$z_m = y_m - \sum_{n=1}^{N} a_{mn}\mu_n + \frac{z_m}{M}\sum_{n=1}^{N} F'_n(\alpha_n, c), \forall m = 1, \dots, M$
```

- **Parameter updates**

Updating $\boldsymbol{\alpha}$:

$\forall n = 1, 2, \dots, N$, solve for α_n in

$\alpha_n^3 - e^{\left\{\frac{(\Sigma\Delta)_{n,0} - (\Sigma\Delta)_{n,1} - 1}{\theta_1}\right\}} \alpha_n^2 + \frac{\theta_2}{2}\alpha_n - \frac{\theta_2(\mu_n^2 + \nu_n)}{2} = 0$

which is the minimizer of

$$f(\alpha_n) = \ln(\alpha_n) + \frac{\mu_n^2 + \nu_n}{\alpha_n} + \frac{1}{\theta_2}\left(\alpha_n - e^{\left\{\frac{(\Sigma\Delta)_{n,0} - (\Sigma\Delta)_{n,1} - 1}{\theta_1}\right\}}\right)^2$$

Updating the noise variance σ^2:

$\sigma^{2[k+1]} = \frac{\|\boldsymbol{y} - A\boldsymbol{\mu}\|_2^2 + \sum_{n=1}^{N} \|\boldsymbol{a}_n\|_2^2 \nu_n}{M}$

Updating the variance of $\boldsymbol{\alpha}$:

$\theta_2^{[k+1]} = \frac{1}{N}\sum_{n=1}^{N}\left(\alpha_n - e^{\left\{\frac{(\Sigma\Delta)_{n,0} - (\Sigma\Delta)_{n,1} - 1}{\theta_1}\right\}}\right)^2$

In the above algorithm, the hyperparameter θ_1 is a tuning parameter. Since we use Gaussian message passing modeling, it is faster than the conventional SBL modeling. In other words, it provides a trade-off between the exactness of the SBLs and the fast runtime of the greedy-based algorithms.

3.3 Simulation Results

This section contains the simulation results on both synthetic and real-data for the proposed algorithm against some of the existing algorithms in this area.

3.3.1 Simulations on the Synthetically Generated Data

Here, we demonstrate the performance of our algorithm compared to two versions the SBL algorithm borrowed from [99] and the approximate message passing SBL (AMP-SBL) [36]. For the simulation purposes, the elements of vector $\boldsymbol{x}_{np}$ are drawn i.i.d. from Gaussian distribution with zero-mean and variance $\sigma_x^2 = 1$, where $\boldsymbol{x}_{np}$ denotes a non-sparse solution vector. The supports of the solution are binary and randomly drawn from a Bernoulli distribution in such a way to have a random block-sparsity structure. The true block-sparse solution is then constructed from $\boldsymbol{x} = \boldsymbol{s} \circ \boldsymbol{x}_{np}$, where "$\circ$" denotes the Hadamard product. The sensing matrix A is 100×200 and is randomly drawn from $\mathcal{N}(0,1)$ and then it is normalized with respect to its columns. The elements of the noise component are drawn from $e_{mn} \sim \mathcal{N}(0, \sigma_n^2)$. We vary σ_n^2 to illustrate the performance over different SNRs. The measurement vector $\boldsymbol{y}$ is 100×1 and is computed from $\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$. We evaluate the performance of the proposed algorithm using the normalized mean-squared error defined as

$$NMSE := \mathbb{E}\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2^2 / \mathbb{E}\|\boldsymbol{x}\|_2^2,$$

where $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}$ denote the estimated solution and the true block-sparse solution, respectively. In Fig. 3.2, we generate 100 different SMV problems with the aforementioned features and then average over the obtained results. All of the compared algorithms in this figure are set to 1000 iterations. In this figure, we set $\sigma_x^2 = 1$ and change the noise variance σ_n^2 for

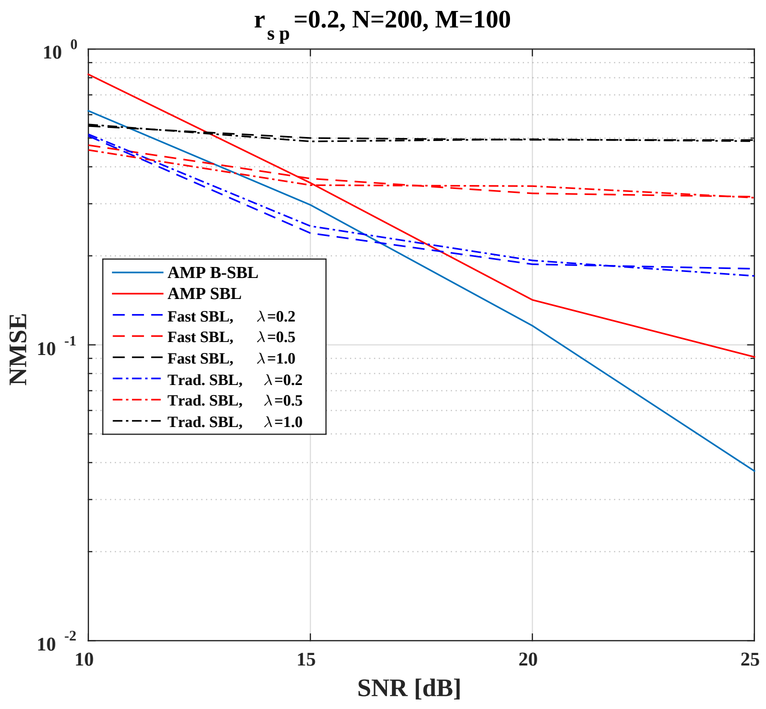


Fig. 3.2: Performance when varying the SNR.

evaluating the performance at different SNRs. In the legend of Fig. 3.2, “AMP-B-SBL” and “AMP-SBL” denote our algorithm and the algorithm proposed in [36], respectively. Also, “Trad.-SBL” and “Fast-SBL” are the traditional SBL and the EM-based fast version of the traditional SBL, respectively [99]. The term λ is the emphasis parameter on the sparsity over the error term, and r_{sp} is defined as the sparsity level divided by the length of the original signal i.e., $r_{sp} = K_{sp}/N$. We chose K_{sp} in such a way that on the overage $r_{sp} \simeq 0.2$. It can be seen from Fig. 3.2 that the normalized mean-squared error for both Trad.-SBL and Fast-SBL are the same for the same SNR values and almost remain constant as the SNR varies. Our algorithm shows less NMSE compared to those algorithms and also compared to the AMP-SBL. The reason that our algorithm works better than the AMP-SBL is due to the fact that the AMP-B-SBL accounts for the unknown block-sparsity in the solution.

We now apply AMP-B-SBL to a specific example. In this case, we randomly generated a block-sparse signal and the corresponding set of measurements as described earlier. In our case, the true supports of the solution (non-zeros of $\boldsymbol{s}$) are $\{28 : 30, 89 : 95, 106, 117, 118, 128 :$

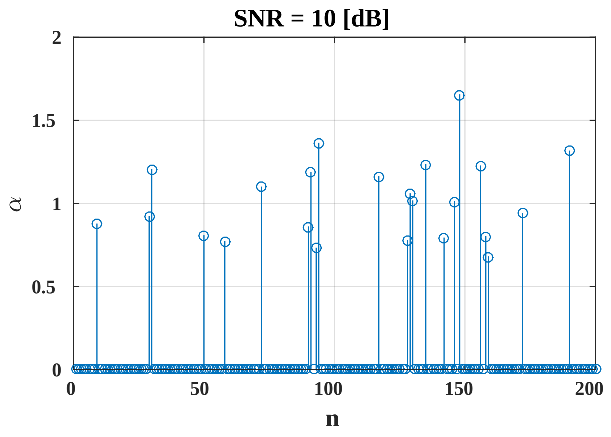


Fig. 3.3: Updated $\boldsymbol{\alpha}$ for SNR=10[dB] when using AMP-B-SBL.

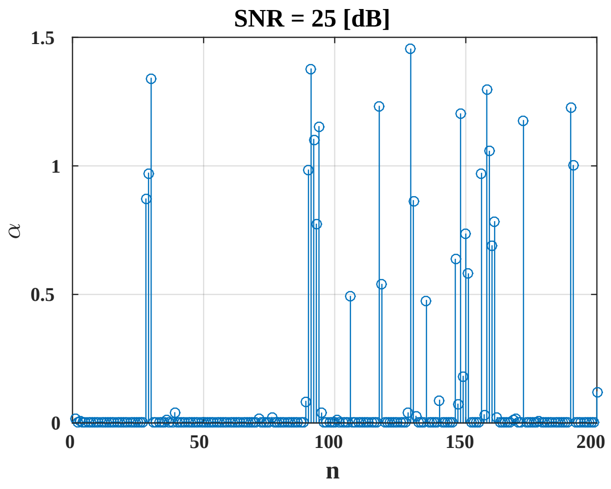


Fig. 3.4: Updated $\boldsymbol{\alpha}$ for SNR=25[dB] when using AMP-B-SBL.

$131, 135, 146 : 151, 156 : 162, 172, 190,$

$191, 200\}$ with $\{28 : 30\}$ denoting the 28th through 30th entries of $\boldsymbol{s}$. For the initialization of the parameters, we set $\theta_1 = 100$, $c^{[0]} = 5$, $\theta_2^{[0]} = 0.002$, and the threshold $T = 0.001$. Fig. 3.3 and Fig. 3.4 illustrate the obtained $\alpha_n, \forall n = 1, 2, \ldots, N$ after 1000 iterations for the SNR=10 [dB] and 25 [dB], respectively.

In Fig. 3.5 and Fig. 3.6 we demonstrate the comparison between the true and the estimated solution of AMP-B-SBL algorithm for our case scenario. It can be seen from Fig. 3.5 that at the lower SNR, our algorithm could almost successfully find the active entries of the solution. There are also some false supports found by the algorithm and those caused

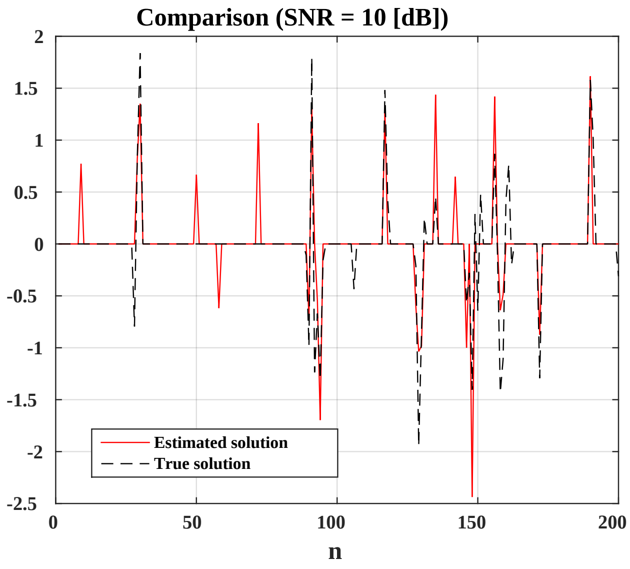


Fig. 3.5: Comparison of $\boldsymbol{x}$ with $\hat{\boldsymbol{x}}$ for SNR=10[dB].

the estimated amplitudes of the solution differ from the true ones. According to Fig. 3.6, the performance of the algorithm for SNR=25[dB] is satisfactory in terms of both the support detection and the non-zero amplitudes of the true solution. Based on Fig. 3.6, although the AMP-B-SBL also found some other supports which are not in the true solution, those false supports clumped together and have very small amplitudes. Such clumpiness is due to the fact that the AMP-B-SBL encourages the Sigma-Delta measure of the solution to be small. Also, in Fig. 3.7 and Fig. 3.8 we show the support recovery of our example as it is being updated for the SNRs of 10 [dB] and 25 [dB], respectively.

We now demonstrate the performance of AMP-B-SBL algorithm compared to orthogonal matching pursuit (OMP) [76], basis pursuit denoising (BPDN) [44], MFOCUSS [53], and AMP-SBL [36] algorithms for solving the SMV problem. In all the simulations, we used the default settings for MFOCUSS algorithm. For the OMP, the stopping condition is set to $\sqrt{0.5N\sigma^2}$. In AMP-B-SBL, we set $c^{[10]}=10$, $\alpha_n=2\times10^{-4}, \forall n=1,\dots,N$, $T=0.001$, $\theta_1=8$, $\theta_2^{[0]}=0.6$, and the number of iterations is set to 1000. For the AMP-SBL algorithm, we set $c^{[0]}=10$, $\gamma_n=2\times10^{-4}, \forall n=1,\dots,N$, and the number of iterations is set to 1000. Finally, for BPDN we stopped the algorithm once $\|\boldsymbol{y}-A\hat{\boldsymbol{x}}\|$ becomes less than 0.75.

In the first set of experiments, we generate 200 independent trials and then average

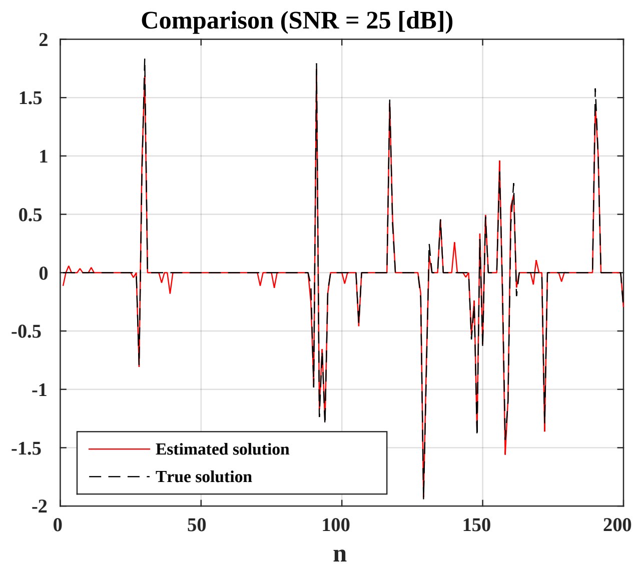


Fig. 3.6: Comparison of $\boldsymbol{x}$ with $\hat{\boldsymbol{x}}$ for SNR=25[dB].

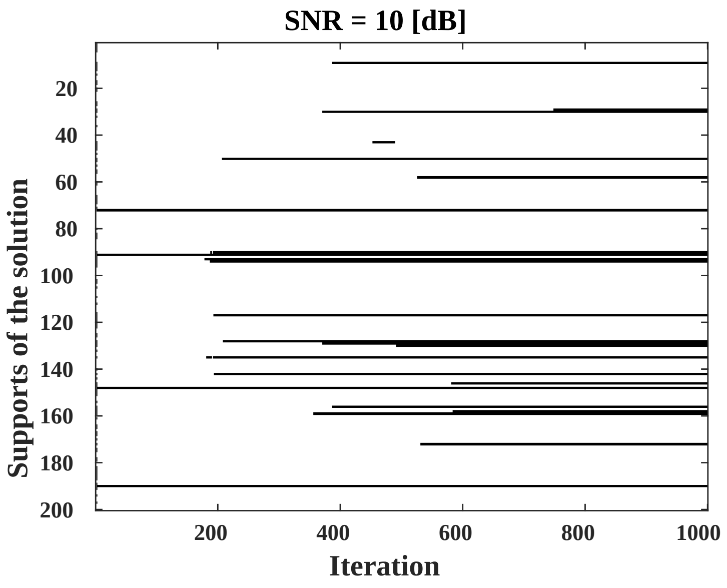


Fig. 3.7: Support recovery using AMP-B-SBL for SNR=10[dB].

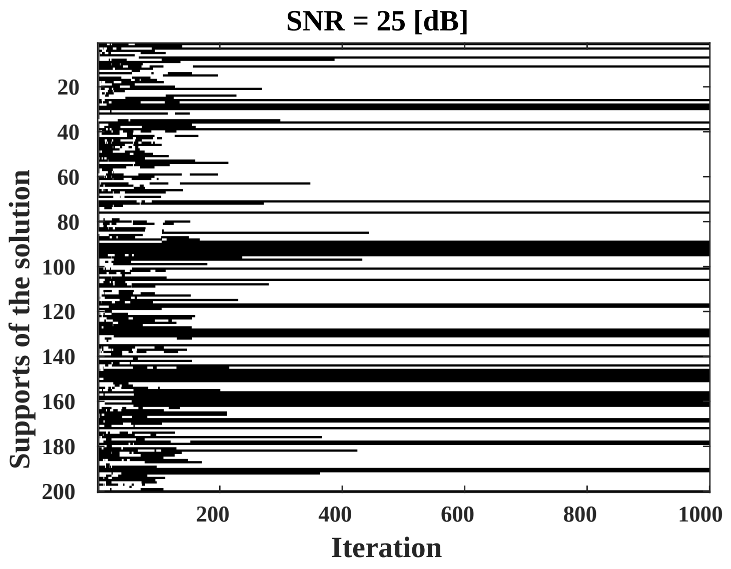


Fig. 3.8: Support recovery using AMP-B-SBL for SNR=25[dB].

over the obtained results. In this case, we evaluate the performance of the algorithms via detection and false alarm rate in support estimation, and also the normalized mean-squared error (NMSE) between the true and the estimation solutions.

The trials are obtained from the same way explained earlier in this section. In Fig. 3.9, the empirical results of detection rate *vs.* the ratio $\lambda = M/N$ are illustrated. In the simulations, we discarded those estimated supports which their corresponding amplitudes become less than 0.01. In Fig. 3.9 we see that AMP-B-SBL shows the highest detection rate for $\lambda \in [0.05, 0.2]$. For $0.25 \leq \lambda \leq 0.55$ AMP-B-SBL, AMP-SBL, BPDN, and MFOCUSS demonstrate almost the same detection rate. Finally, for $\lambda \geq 0.55$ AMP-B-SBL, AMP-SBL, OMP, and MFOCUSS show approximately the same performance.

In Fig. 3.10, the false alarm rate (the rate of deciding on wrong supports in the solution) *vs.* λ is illustrated. In Fig. 3.10 we observe that our algorithm has a higher false alarm rate compared to the other algorithms for $\lambda = [0, 0.2]$. Comparing the detection rate with the false alarm rate of AMP-B-SBL for such range of λ shows the trade off that exists between the detection rate and false alarm rate. For $0.2 < \lambda \leq 0.3$ all the algorithms show the same performance. However, our algorithm illustrates the *lowest* false alarm rate for $\lambda > 0.3$. Fig. 3.11 demonstrates the overall performance of the algorithms as a combination of support

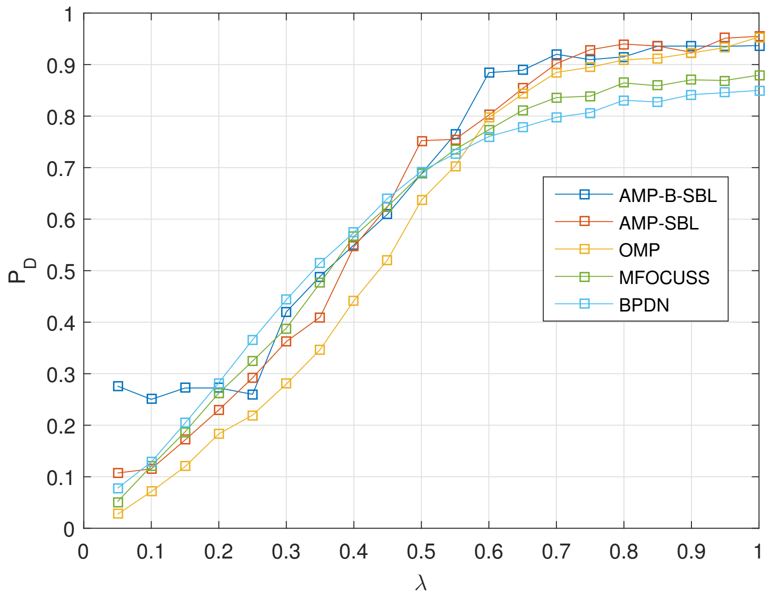


Fig. 3.9: Comparison in terms of detection rate.

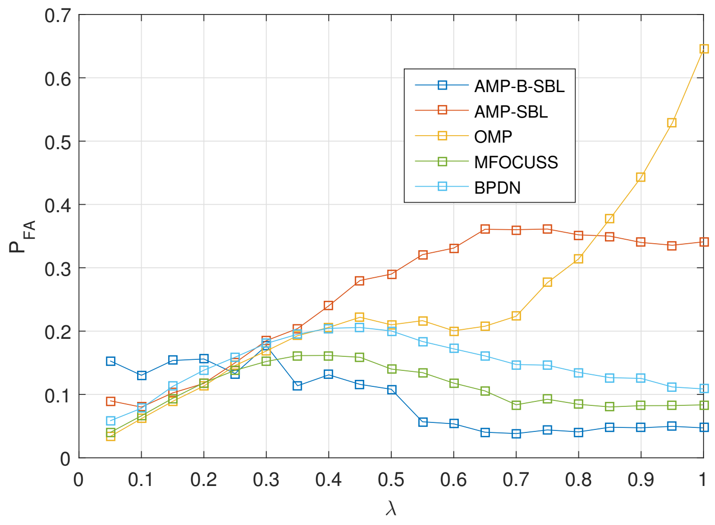


Fig. 3.10: Comparison of false alarm rate in support recovery.

detection, false alarm rate, and the number of measurements.

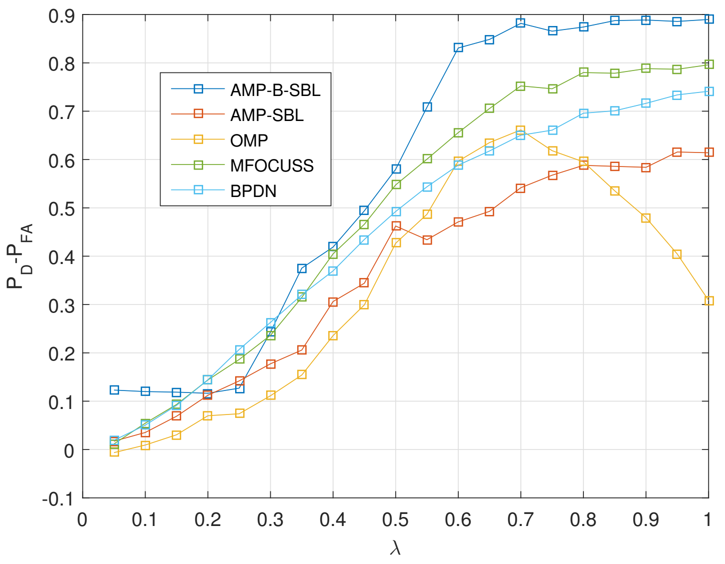


Fig. 3.11: Comparison based on the difference between the experimental detection and false alarm rate.

From Fig. 3.11, we observe that the best performance belongs to the AMP-B-SBL algorithm. This shows that incorporating our measure of contiguity in the supports into the SBL algorithm boosts the recovery performance for the clustered pattern sparse signals. Finally, Fig. 3.12 shows the performance comparison in terms of normalized mean-squared error between the true and estimated solutions. According to Fig. 3.12, AMP-B-SBL outperforms the other algorithms in terms of estimating the true solution.

3.3.2 Simulations on Real-Data

Here, we further evaluate the performance of the algorithms via the following example. In this case we use the image of 112×200 pixels, where we consider the black pixels as the “interesting” locations. The image is shown in Fig. 3.13. The image is in fact a representation of total electron content (TEC) of somewhere in the upper atmosphere, where is demonstrated as a black and white image here for the simulation purposes.

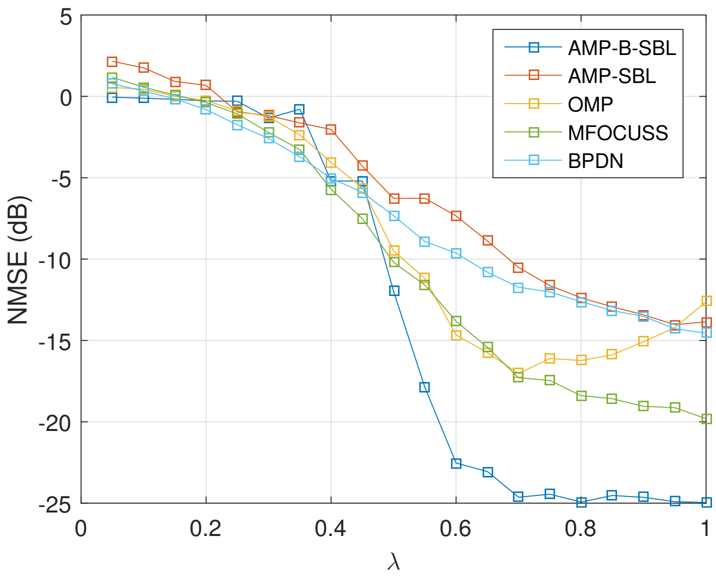


Fig. 3.12: Comparison in terms of normalized mean-squared error between the true and estimated solution.

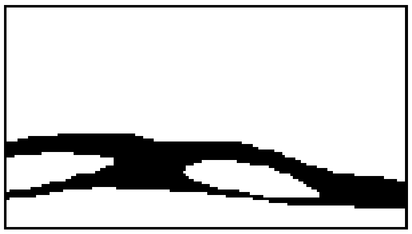


Fig. 3.13: True image.

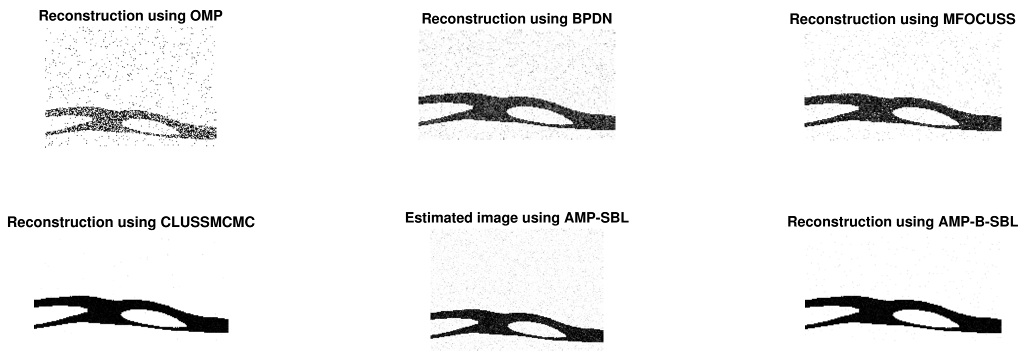

Reconstruction using OMP
Reconstruction using BPDN
Reconstruction using MFOCUSS
Reconstruction using CLUSSMCMC
Estimated image using AMP-SBL
Reconstruction using AMP-B-SBL

Fig. 3.14: Results of reconstructed images for OMP, BPDN, MFOCUSS, CLUSSMCMC, AMP-SBL, and AMP-B-SBL.

Table 3.2: NMSE and PSNR comparison in image reconstruction.

Algorithm	**NMSE (dB)**	**PSNR (dB)**	**speed**
OMP	-1.7124	10.9425	Very Fast
BPDN	-8.7905	17.0205	Very Fast
MFOCUSS	-10.6522	18.8822	Slow
CLUSSMCMC	-27.42	35.6504	Slow
AMP-SBL	-13.4642	22.6943	Fast
AMP-B-SBL	-24.5600	32.7901	Fast

In the simulations for this image, the value of 1 is assigned to the pixels with black color and 0 to the white ones. Then, the matrix corresponding to the image is reshaped and the vector $\boldsymbol{x} \in \mathbf{R}^{11200 \times 1}$ is constructed. The number of measurements are set to 5040, i.e., $\lambda = 0.45$. We construct the sensing matrix in the same way we described earlier. Then, the measurements are obtained from $\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$ with SNR=25 dB. We feed all the algorithms with the measurement vector $\boldsymbol{y}$ and the sensing matrix A. The reconstructed images are illustrated in Fig. 3.14. In all the algorithms, we applied the threshold of 0.01, meaning that we discarded those estimated supports that corresponded to non-zero elements with the absolute value of less than 0.01. Below, we also include the reconstruction based on CLUSSMCMC algorithm [68]. In order to compare the reconstructed images illustrated in Fig. 3.14, we compare the obtained results based on the NMSE and peak-SNR (PSNR) as shown in Table 3.2. According to Table 3.2, we observe that CLUSSMCMC provides the best performance in terms of NMSE and PSNR. However, it is much slower compared to other algorithms. In contrast, AMP-B-SBL algorithm provides high performance for NMSE, PSNR, and the speed of the algorithm. By comparing the results of AMP-SBL with AMP-B-SBL, we also see that using the measure of clumpiness into the SBL algorithm definitely improves the reconstruction performance.

3.4 Summary

A new algorithm for solving the block-sparse SMV problem is proposed. By assuming Gaussian distributions for all the variables, we were able to use the factor graph which only needs to propagate the mean and variance of the full messages. We further reduced the computational load of the problem using the approximate message passing. It was shown that dependence of the solution variances on Sigma-Delta encourages the solution to have a block-sparsity structure and also reduces the normalized mean-squared error between the true and the estimated sparse signal.

3.5 Appendix

Here, more details on the update rule of the parameters and variables of the proposed model are provided.

3.5.1 Factor Graph

The factor graph of our model for the B-SMV problem is illustrated in Fig. 3.15. In such graphical probabilistic model, the large circle nodes represent the random variables and the shaded boxes show the function nodes. The small circles on the right hand-side represent the set of hyper-parameters and the shaded circles on the left denote the set of observations.

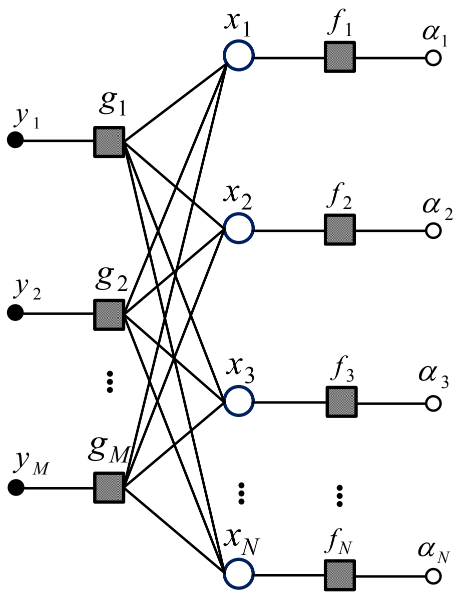


Fig. 3.15: Factor graph for the B-SMV problem.

In Fig. 3.15, the term g_m denotes the likelihood function based on the observation y_m and is defined as

$$g_m = p(\boldsymbol{y}|\boldsymbol{x}). \tag{3.15}$$

Assume that we have M observations collected into $\boldsymbol{y} = [y_1, \ldots, y_M]^T$ and the model is $\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$ with $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 I_N)$. Therefore, we have

$$p(\boldsymbol{y}|\boldsymbol{x}; \sigma^2) = (2\pi\sigma^2)^{-\frac{M}{2}} e^{\{-\frac{1}{2\sigma^2}\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2\}}. \tag{3.16}$$

The prior distribution on the solution vector $\boldsymbol{x}$ is defined as $x_n \sim \mathcal{N}(0, \alpha_n), \forall n = 1, \ldots, N$. In this factor graph, the function node f_n is defined as follows

$$f_n = p(x_n; 0, \alpha_n),\ \forall n = 1, \ldots, N.$$

3.5.2 Message Passing from a Function Node to a Variable Node

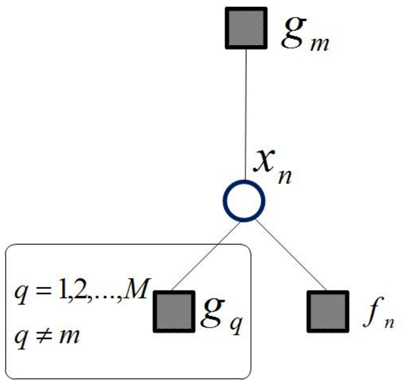


Fig. 3.16: Message passing from a function node to a variable node when there also exist other function nodes connected to the same variable node.

According to Fig. 3.16, the message propagates from the function node g_m to the variable node x_n can be obtained as

$$M_{g_m \to x_n}(x_n) \propto \int_{x_{-n}} g_m\big(\mathrm{Ne}(g_m)\big) \prod_{q \neq n} M_{x_q \to g_m}(x_q), \tag{3.17}$$

where $\int_{x_{-n}}$ denotes the integration over all $x_i, i \neq n$.

Suppose that the message (containing the mean and variance) of a variable node $x_n, \forall n = 1, \ldots, N$ to a function node $g_m, \forall m = 1, \ldots, M$ is as follows

$$M_{g_m \to x_n} \propto \mathcal{N}(a_{mn}x_n; z_{mn}, c_{mn}), \tag{3.18}$$

where

$$z_{mn} = y_m - \sum_{q \neq n} a_{mq}\mu_q$$

$$c_{mn} = \sigma^2 + \sum_{q \neq n} |a_{mq}|^2 \Sigma_q.$$

For simplicity and without loss of generality, suppose that the vector $\boldsymbol{x}$ has only two elements and is defined as $\boldsymbol{x} := [x_1, x_2]^T$. Suppose that we are interested in finding the message passes from the function node g_m to the variable node x_1. Let's have the following definitions

$$X_1 := a_{m1}x_1, \quad \bar{y}_m := y_m - X_1.$$

Now the posterior inference on the variable x_1 can be obtained from the followings

$$\begin{aligned} p(X_1|-) &\propto \int \exp\{-\frac{1}{2\sigma^2}(y_m - a_{m1}x_1 - a_{m2}x_2)^2 - \frac{1}{2}\sum_{q\neq 1}^{2}\frac{1}{\Sigma_q}(x_q-\mu_q)^2\}dx_2 \\ &\propto \int \exp\{-\frac{1}{2\sigma^2}(a_{m2}x_2 - \bar{y}_m)^2 - \frac{1}{2\Sigma_2}(x_2-\mu_2)^2\}dx_2 \\ &\propto \int \exp\{-\frac{1}{2\sigma^2}(a_{m2}^2x_2^2 - 2a_{m2}\bar{y}_m x_2 + \bar{y}_m^2) - \frac{1}{2\Sigma_2}(x_2^2 - 2\mu_2 x_2)\}dx_2 \\ &\propto \exp\{-\frac{1}{2\sigma^2}\bar{y}_m^2\}\int \exp\{-\frac{1}{2}((\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})x_2^2 - 2x_2(\frac{a_{m2}}{\sigma^2}\bar{y}_m + \frac{\mu_2}{\Sigma_2}))\}dx_2. \end{aligned}$$

Now, we factor out the term $\frac{a_{mn}^2}{\sigma^2} + \frac{1}{\Sigma_2}$ (inside the integral) and complete the square with respect to the variable x_2.

$$\begin{aligned} p(X_1|-) &\propto \exp^{\{-\frac{1}{2\sigma^2}\bar{y}_m^2\}} \times \\ &\int e^{(\{-\frac{1}{2}\left((\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})(x_2-(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}(\frac{a_{m2}}{\sigma^2}\bar{y}_m+\frac{\mu_2}{\Sigma_2}))^2-(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}(\frac{a_{m2}}{\sigma^2}\bar{y}_m+\frac{\mu_2}{\Sigma_2})^2\right)\}}dx_2 \end{aligned}$$

Notice that the term

$$\int \exp\{-\frac{1}{2}\big((\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})(x_2 - (\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}(\frac{a_{m2}}{\sigma^2}\bar{y}_m + \frac{\mu_2}{\Sigma_2}))^2\big)\}dx_2$$

is the integration of a probability distribution function under its whole support and is equal to 1. Therefore,

$$\begin{aligned}
p(X_1|-) &\propto \exp\{-\frac{1}{2}\big(\frac{1}{\sigma^2}\bar{y}_m^2 - (\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}(\frac{a_{m2}}{\sigma^2}\bar{y}_m+\frac{\mu_2}{\Sigma_2})^2\big)\} \\
&\propto \exp\{-\frac{1}{2}\big(\frac{1}{\sigma^2}\bar{y}_m^2 - (\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}(\frac{a_{m2}^2}{\sigma^4}\bar{y}_m^2+2\frac{a_{m2}}{\sigma^2}\frac{\mu_2}{\Sigma_2}\bar{y}_m)\big)\} \\
&\propto \exp\{-\frac{1}{2\sigma^2}\big((y_m-X_1)^2 - (\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}(\frac{a_{m2}^2}{\sigma^2}(y_m-X_1)^2+2a_{m2}\frac{\mu_2}{\Sigma_2}(y_m-X_1))\big)\} \\
&\propto e^{\{-\frac{1}{2\sigma^2}\big(X_1^2(1-(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}\frac{a_{m2}^2}{\sigma^2})-2X_1(y_m-(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}(\frac{a_{m2}^2}{\sigma^2}y_m+a_{m2}\frac{\mu_2}{\Sigma_2}))\big)\}}
\end{aligned}$$

Therefore,

$$\begin{aligned}
&p(X_1|-) = p(a_{m1}x_1|-) \propto \\
&e^{\{-\frac{1}{2}\big(\frac{1}{\sigma^2}(1-(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}\frac{a_{m2}^2}{\sigma^2})\big)\big(X_1-\frac{\sigma^2}{\sigma^2}(1-(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}\frac{a_{m2}^2}{\sigma^2})^{-1}(y_m-(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}(\frac{a_{m2}^2}{\sigma^2}y_m+a_{m2}\frac{\mu_2}{\Sigma_2}))\big)^2\}}.
\end{aligned} \tag{3.19}$$

Finding the variance

Based on (3.19), the variance of X_1 can be found as follows

$$\begin{aligned}
\Sigma_{X_1} &= \big(\frac{1}{\sigma^2}(1-(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}\frac{a_{m2}^2}{\sigma^2})\big)^{-1} \\
&= \sigma^2\big(1-(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})^{-1}\frac{a_{m2}^2}{\sigma^2}\big)^{-1}.
\end{aligned} \tag{3.20}$$

Based on the matrix inversion lemma, we have

$$(A-BD^{-1}C)^{-1} = A^{-1}+A^{-1}B(D-CA^{-1}B)^{-1}CA^{-1}, \tag{3.21}$$

where for our case, $A=1, B=1, C=\frac{a_{m2}^2}{\sigma^2}$, and $D=(\frac{a_{m2}^2}{\sigma^2}+\frac{1}{\Sigma_2})$. By substituting (3.21) into (3.20) we will have

$$\begin{aligned}
\Sigma_{X_1} &= \sigma^2\big(1+(\frac{a_{m2}^2}{\sigma^2}+\frac{2}{\Sigma_2}-\frac{a_{m2}^2}{\sigma^2})^{-1}\frac{a_{m2}^2}{\sigma^2}\big) \\
&= \sigma^2(1+\Sigma_2\frac{a_{m2}^2}{\sigma^2}).
\end{aligned}$$

Therefore,

$$\Sigma_{X_1} = \sigma^2 + a_{m2}^2 \Sigma_2. \tag{3.22}$$

Finding the mean

According to (3.19), the mean of the random variable X_1 i.e., $a_{m1}E[x_1]$ is

$$\begin{aligned}
\mu_{X_1} &= \big(1 - (\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})^{-1} \frac{a_{m2}^2}{\sigma^2}\big)^{-1} \big(y_m - (\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})^{-1} \frac{a_{m2}^2}{\sigma^2} y_m - (\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})^{-1} a_{m2} \frac{\mu_2}{\Sigma_2}\big) \\
&= \big(1 - (\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})^{-1} \frac{a_{m2}^2}{\sigma^2}\big)^{-1} \big(y_m(1 - (\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})^{-1} \frac{a_{m2}^2}{\sigma^2}) - (\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})^{-1} a_{m2} \frac{\mu_2}{\Sigma_2}\big) \\
&= y_m - \big(1 - (\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})^{-1} \frac{a_{m2}^2}{\sigma^2}\big)^{-1} (\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})^{-1} a_{m2} \frac{\mu_2}{\Sigma_2} \\
&= y_m - \Big((\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})\big(1 - (\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2})^{-1} \frac{a_{m2}^2}{\sigma^2})\Big)^{-1} a_{m2} \frac{\mu_2}{\Sigma_2} \\
&= y_m - \big((\frac{a_{m2}^2}{\sigma^2} + \frac{1}{\Sigma_2}) - \frac{a_{m2}^2}{\sigma^2}\big)^{-1} a_{m2} \frac{\mu_2}{\Sigma_2} \\
&= y_m - \Sigma_2 a_{m2} \frac{\mu_2}{\Sigma_2}.
\end{aligned}$$

Therefore,

$$\mu_{x_1} = y_m - a_{m2}\mu_2. \tag{3.23}$$

Finally, it is straight forward to show that the generalized form of the message that propagates from the function node g_m to the variable node x_n is as follows

$$M_{g_m \to x_n} \propto \mathcal{N}(a_{mn}x_n; z_{mn}, c_{mn}),$$

where

$$\begin{aligned}
z_{mn} &= y_m - \sum_{q \neq n} a_{mq}\mu_q \\
c_{mn} &= \sigma^2 + \sum_{q \neq n} |a_{mq}|^2 \Sigma_q.
\end{aligned}$$

3.5.3 Product of Messages Passed from Function Nodes to a Variable Node

Based on what we obtained in (3.18), we then have

$$\begin{aligned}\prod_{l\neq m} M_{g_l\to x_n} &\propto \exp\Big\{-\frac{1}{2}\sum_{l\neq m}\frac{(a_{ln}x_n - z_{ln})^2}{c_{ln}}\Big\}\\ &\propto \exp\Big\{-\frac{1}{2}\big((\sum_{l\neq m}\frac{a_{ln}^2}{c_{ln}})x_n^2 - 2x_n(\sum_{l\neq m}\frac{a_{ln}z_{ln}}{c_{ln}})\big)\Big\}.\end{aligned}$$

Therefore,

$$\prod_{l\neq m} M_{g_l\to x_n} \propto \mathcal{N}\Big(x_n; \frac{\sum_{l\neq m}\frac{a_{ln}z_{ln}}{c_{ln}}}{\sum_{l\neq m}\frac{a_{ln}^2}{c_{ln}}}, \frac{1}{\sum_{l\neq m}\frac{a_{ln}^2}{c_{ln}}}\Big). \tag{3.24}$$

In the large system limit we can approximate c_{ln} by

$$c_{ln} \simeq c_n := \frac{1}{M}\sum_{m=1}^{M} c_{mn}. \tag{3.25}$$

Under the assumption that the sensing matrix A have already normalized with respect to its columns, we then can have the following approximation

$$\sum_{l\neq m} a_{ln}^2 \simeq \sum_{m=1}^{M} a_{mn}^2 = 1. \tag{3.26}$$

Substituting (3.25) and (3.26) into (3.24) yields

$$\prod_{l\neq m} M_{g_l\to x_n} \propto \mathcal{N}\Big(x_n; \sum_{l\neq m} a_{ln}z_{ln}, c_{ln}\Big). \tag{3.27}$$

3.5.4 Message Passing from a Variable Node to a Function Node

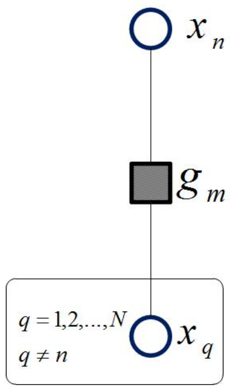


Fig. 3.17: Message passing from a variable node to a function node when there also exist other variable nodes connected to the same function node.

According to Fig. 3.17, the message propagates from the variable node x_n to the function node g_m can be obtained as follows

$$M_{x_n \to g_m} \propto M_{f_n \to x_n} \prod_{l \neq m} M_{g_l \to x_n}, \tag{3.28}$$

where $\prod_{l \neq m} M_{g_l \to x_n}$ was obtained in (3.24) and $M_{f_n \to x_n} \propto \mathcal{N}(x_n; 0, \alpha_n)$. Therefore,

$$\begin{aligned} M_{x_n \to g_m} &\propto \exp\Big\{ -\frac{1}{2}\big(\alpha_n^{-1} x_n^2 + (\sum_{l \neq m} \frac{a_{ln}^2}{c_{nl}})(x_n - \frac{\sum_{l \neq m} \frac{a_{nl} z_{nl}}{c_{nl}}}{\sum_{l \neq m} \frac{a_{nl}^2}{c_{nl}}})^2\big)\Big\} \\ &\propto \exp\Big\{ -\frac{1}{2}\Big((\alpha_n^{-1} + \sum_{l \neq m} \frac{a_{ln}^2}{c_{nl}}) x_n^2 - 2x_n\big((\sum_{l \neq m} \frac{a_{ln}^2}{c_{nl}})\frac{\sum_{l \neq m} \frac{a_{nl} z_{nl}}{c_{nl}}}{\sum_{l \neq m} \frac{a_{ln}^2}{c_{nl}}}\big)\Big)\Big\}. \end{aligned}$$

Using (3.25), we then have

$$\begin{aligned}
M_{x_n \to g_m} &\propto \exp\big\{-\frac{1}{2}\big((\frac{1}{\alpha_n}+\frac{1}{c_n}\sum_{l\neq m} a_{ln}^2)x_n^2 - 2x_n(\frac{1}{c_n}\sum_{l\neq m} a_{nl}c_{nl})\big)\big\} \\
&\propto \exp\big\{-\frac{1}{2}\big((\frac{c_n+\alpha_n\sum_{l\neq m} a_{ln}^2}{c_n\alpha_n})x_n^2 - 2x_n(\frac{1}{c_n}\sum_{l\neq m} a_{nl}c_{nl})\big)\big\} \\
&\propto \exp\big\{-\frac{1}{2}(\frac{c_n+\alpha_n\sum_{l\neq m} a_{ln}^2}{c_n\alpha_n})(x_n - \frac{c_n\alpha_n}{c_n+\alpha_n\sum_{l\neq m} a_{ln}^2}\frac{1}{c_n}\sum_{l\neq m} a_{nl}z_{nl})^2\big\}.
\end{aligned}$$

Therefore, the message that goes from the variable node x_n to the function node g_m is as follows

$$M_{x_n \to g_m} \propto \mathcal{N}\big(x_n; \sum_{l\neq m} a_{nl}z_{ln}\frac{\alpha_n}{c_n+\alpha_n\sum_{l\neq m} a_{ln}^2}, \frac{c_n\alpha_n}{c_n+\alpha_n\sum_{l\neq m} a_{ln}^2}\big). \tag{3.29}$$

Finally, substituting (3.26) into (3.29) leads to the message passing below.

$$M_{x_n \to g_m} \propto \mathcal{N}\big(x_n; \sum_{l\neq m} a_{nl}z_{ln}\frac{\alpha_n}{c_n+\alpha_n}, \frac{c_n\alpha_n}{c_n+\alpha_n}\big). \tag{3.30}$$

3.5.5 Estimating the Posterior on the Variables of Interest

Here, we estimate the posterior inference for the variable x_n.

$$\begin{aligned}
p(x_n|\boldsymbol{y}) &\propto p(x_n;\alpha_n)p(\boldsymbol{y}|x_n,\sigma^2) \\
&\propto p(x_n;\alpha_n)\prod_{m=1}^{M} p(y_m|x_n) \\
&\propto M_{f_n \to x_n}\prod_{m=1}^{M} M_{g_m \to x_n}
\end{aligned}$$

Using (3.24), it is straightforward to show that

$$\prod_{m=1}^{M} M_{g_m \to x_n} \propto \mathcal{N}\big(x_n; \frac{\sum_{m=1}^{M}\frac{a_{mn}z_{mn}}{c_{mn}}}{\sum_{m=1}^{M}\frac{a_{mn}^2}{c_{mn}}}, \frac{1}{\sum_{m=1}^{M}\frac{a_{mn}^2}{c_{mn}}}\big). \tag{3.31}$$

Based on (3.29) and (3.31), we finally get the followings

$$p(x_n|\boldsymbol{y}) \propto \mathcal{N}(x_n; \mu_{x_n|-}, \Sigma_{x_n|-}), \tag{3.32}$$

where

$$\mu_{x_n|-} = \sum_{m=1}^{M} a_{mn} z_{mn} \Big(\frac{\alpha_n}{c_n + \alpha_n \sum_{m=1}^{M} a_{mn}^2}\Big),$$
$$\Sigma_{x_n|-} = \frac{c_n \alpha_n}{c_n + \alpha_n \sum_{m=1}^{M} a_{mn}^2}.$$

Again, substituting (3.25) and (3.26) into (3.32), yields

$$\mu_{x_n|-} = \sum_{m=1}^{M} a_{mn} z_{mn} \Big(\frac{\alpha_n}{c_n + \alpha_n}\Big),$$
$$\Sigma_{x_n|-} = \frac{c_n \alpha_n}{c_n + \alpha_n}.$$

3.5.6 Update Rules for the Hyper-Parameters

Below, we describe the update equations for our model using the EM algorithm.

3.5.7 Update Rule for α

In this section, we describe the update rule for the variance α_n of x_n using the expectation-maximization (EM) algorithm. Notice that our prior on the entries of the solution vector $\boldsymbol{x}$ is $x_n \sim \mathcal{N}(0, \alpha_n), \forall n = 1, \ldots, N$. The reason for considering such prior on $\boldsymbol{x}$ is to encourage the sparsity in the solution. We further consider the following prior distribution on each α_n and make it depend on the measure of clumpiness i.e., Sigma-Delta

$$\alpha_n \sim \mathcal{N}\big(e^{\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0} - (\Sigma\Delta)|_{b(x_n,.)=1} - 1}{\theta_1}\}}, \theta_2\big), \quad \forall n = 1, \ldots, N. \tag{3.33}$$

Therefore,

$$\begin{aligned}
{\alpha_n}^{[k+1]} &= \arg\max_{\alpha_n} \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\sigma^2,{\alpha_n}^{[k]}}\big[\log\{p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\alpha},\theta_1,\theta_2,\sigma^2)\}\big] \\
&= \arg\max_{\alpha_n} \mathbb{E}_{\boldsymbol{x}|-}\big[\log\{p(\boldsymbol{y}|\boldsymbol{x},\sigma^2 I_N)\prod_{n=1}^{N} p(x_n;0,\alpha_n)p(\alpha_n;e^{\left\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0}-(\Sigma\Delta)|_{b(x_n,.)=1}-1}{\theta_1}\right\}},\theta_2)\}\big]
\end{aligned} \tag{3.34}$$

Notice that

$$\begin{aligned}
&p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\alpha},\theta_1,\theta_2,\sigma^2) \propto (2\pi\sigma^2)^{-\frac{M}{2}} e^{\{-\frac{1}{2\sigma^2}\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2\}} \times \\
&\prod_{n=1}^{N}(2\pi\alpha_n)^{-\frac{1}{2}} e^{\{-\frac{1}{2\alpha_n}x_n^2\}}(2\pi\theta_2)^{-\frac{1}{2}} e^{\{-\frac{1}{2\theta_2}\left(\alpha_n-e^{\left\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0}-(\Sigma\Delta)|_{b(x_n,.)=1}-1}{\theta_1}\right\}}\right)^2\}}.
\end{aligned} \tag{3.35}$$

Substituting (3.35) into (3.34) and discarding the terms that are independent of α_n yields

$$\begin{aligned}
&{\alpha_n}^{[k+1]} \\
&= \arg\max_{\alpha_n} \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\sigma^2,{\alpha_n}^{[k]}}\Big[\log\Big\{{\alpha_n}^{-\frac{1}{2}} e^{\{-\frac{1}{2\alpha_n}x_n^2\}} e^{\left\{-\frac{1}{2\theta_2}\left(\alpha_n-e^{\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0}-(\Sigma\Delta)|_{b(x_n,.)=1}-1}{\theta_1}\}}\right)^2\right\}}\Big\}\Big] \\
&= \arg\min_{\alpha_n} \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\sigma^2,{\alpha_n}^{[k]}}\Big[\log\{\alpha_n\} + \frac{1}{\alpha_n}x_n^2 + \frac{1}{\theta_2}\left(\alpha_n - e^{\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0}-(\Sigma\Delta)|_{b(x_n,.)=1}-1}{\theta_1}\}}\right)^2\Big] \\
&= \arg\min_{\alpha_n} \log\{\alpha_n\} + \frac{1}{\alpha_n}\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\sigma^2,{\alpha_n}^{[k]}}[x_n^2] + \frac{1}{\theta_2}\left(\alpha_n - e^{\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0}-(\Sigma\Delta)|_{b(x_n,.)=1}-1}{\theta_1}\}}\right)^2.
\end{aligned}$$

Notice that

$$\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\sigma^2,{\alpha_n}^{[k]}}[x_n^2] = \mu_{x_n|-}^2 + \Sigma_{x_n|-}.$$

Therefore,

$${\alpha_n}^{[k+1]} = \arg\min_{\alpha_n} f(\alpha_n), \tag{3.36}$$

where

$$f(\alpha_n) = \log\{\alpha_n\} + \frac{\mu^2_{x_n|-} + \Sigma_{x_n|-}}{\alpha_n} + \frac{1}{\theta_2}\big(\alpha_n - e^{\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0} - (\Sigma\Delta)|_{b(x_n,.)=1} - 1}{\theta_1}\}}\big)^2$$

We now perform the maximization step of the EM algorithm, which for our case it becomes a minimization problem.

$$\frac{\partial f(\alpha_n)}{\partial \alpha_n} = 0$$

and hence we solve

$$\alpha_n^3 - e^{\Big\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0} - (\Sigma\Delta)|_{b(x_n,.)=0} - 1}{\theta_1}\Big\}}\alpha_n^2 + \frac{\theta_2}{2}\alpha_n - \frac{\theta_2}{2}(\mu^2_{x_n|-} + \Sigma_{x_n|-}) = 0. \qquad (3.37)$$

Finally, a solution among all the three possible roots for (3.37) which minimizes (3.36) is the update rule for ${\alpha_n}^{[k+1]}$.

3.5.8 Update Rule for σ^2

Here, we describe the update rule to estimate the noise variance using the EM algorithm.

$$\begin{aligned}\sigma^{2[k+1]} &= \arg\max_{\sigma^2} \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\boldsymbol{\alpha},\sigma^{2[k]}}\big[\log\big\{p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\alpha},\theta_1,\theta_2,\sigma^2)\big\}\big] \\ &= \arg\max_{\sigma^2} \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\boldsymbol{\alpha},\sigma^{2[k]}}\big[\log\big\{(2\pi\sigma^2)^{-\frac{M}{2}} e^{\{-\frac{1}{2\sigma^2}\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2\}}\big\}\big]\end{aligned}$$

Notice that in the above maximization problem, we have discarded all the terms that were independent of σ^2. Therefore,

$$\begin{aligned}\sigma^{2[k+1]} &= \arg\ \min_{\sigma^2}\ \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\boldsymbol{\alpha},\sigma^{2[k]}}\left[2M\log\{\sigma\} + \frac{1}{\sigma^2}\|\boldsymbol{y} - A\boldsymbol{x}\|_2^2\right] \\ &= \arg\ \min_{\sigma^2}\ \{2M\log\{\sigma\} + \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\boldsymbol{\alpha},\sigma^{2[k]}}\left[\|\boldsymbol{y} - A\boldsymbol{x}\|_2^2\right]\}\end{aligned} \tag{3.38}$$

Remark 3.1: Here we simplify the following expectation

$$\begin{aligned}E_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\boldsymbol{\alpha},\sigma^{2[k]}}\left[\|\boldsymbol{y} - A\boldsymbol{x}\|_2^2\right] &= \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[\|\boldsymbol{y} - A\big(\boldsymbol{\mu}_{x|-} + (\boldsymbol{x} - \boldsymbol{\mu}_{x|-})\big)\|_2^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[\|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \|A(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})\|_2^2\right] + ... \\ &(\boldsymbol{y} - A\boldsymbol{\mu}_{x|-})^T A\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[\boldsymbol{x} - \boldsymbol{\mu}_{x|-}\right] \\ &= \|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[\|A(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})\|_2^2\right]\end{aligned}$$

Notice that $\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[\boldsymbol{x} - \boldsymbol{\mu}_{x|-}\right] = 0$. Therefore,

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\boldsymbol{\alpha},\sigma^{2[k]}}\left[\|\boldsymbol{y} - A\boldsymbol{x}\|_2^2\right] &= \|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})^T A^T A(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})\right] \\ &= \|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[\operatorname{Tr}\big((\boldsymbol{x} - \boldsymbol{\mu}_{x|-})^T A^T A(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})\big)\right] \\ &= \|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[\operatorname{Tr}\big((\boldsymbol{x} - \boldsymbol{\mu}_{x|-})(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})^T A^T A\big)\right] \\ &= \|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \operatorname{Tr}\big(\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})^T A^T A\right]\big).\end{aligned}$$

Here, we define

$$\Sigma_{x|-} := \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},-}\left[(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})(\boldsymbol{x} - \boldsymbol{\mu}_{x|-})^T\right],$$

which is the variance of the posterior inference on $\boldsymbol{x}$ and is a diagonal matrix. Therefore,

$$\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\theta_2,\boldsymbol{\alpha},\sigma^{2[k]}}\left[\|\boldsymbol{y} - A\boldsymbol{x}\|_2^2\right] = \|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \operatorname{Tr}(\Sigma_{x|-}A^T A). \tag{3.39}$$

Substituting (3.39) into (3.38) yields

$$\sigma^{2[k+1]} = \arg\ \min_{\sigma^2}\ f(\sigma), \tag{3.40}$$

where

$$f(\sigma) := 2M\log\sigma + \frac{1}{\sigma^2}\big(\|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \mathrm{Tr}(\Sigma_{x|-}A^TA)\big)$$

For the minimization purposes, we set $\frac{\partial f(\sigma)}{\partial \sigma} =$, which leads to

$$2M\frac{1}{\sigma} = \frac{2}{\sigma^3}\big(\|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \mathrm{Tr}(\Sigma_{x|-}A^TA)\big).$$

Therefore,

$$\begin{aligned}\sigma^{2[k+1]} &= \frac{\|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \mathrm{Tr}(\Sigma_{x|-}A^TA)}{M} \\ &= \frac{\|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \sum_{n=1}^N \|\boldsymbol{a}_n\|_2^2\Sigma_{x_n|-}}{M}.\end{aligned} \tag{3.41}$$

In the case where the sensing matrix A is normalized with respected to its columns, (3.41) simplifies further as follows

$$\sigma^{2[k+1]} = \frac{\|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \sum_{n=1}^N (\Sigma_{x_n|-})}{M}. \tag{3.42}$$

3.5.9 Update Rule for θ_2

In this section, we describe the update rule for the variance on $\alpha_n, \forall n = 1, \ldots, N$ using the EM algorithm. notice that our prior on each α_n has been defined as

$$\alpha_n \sim \mathcal{N}\big(e^{\left\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0} - (\Sigma\Delta|_{b(x_n,.)=1} - 1)}{\theta_1}, \theta_2\right\}}\big),\ \forall n = 1, \ldots, N. \tag{3.43}$$

Based on the EM algorithm we have

$$\begin{aligned}{\theta_2}^{[k+1]} &= \arg\ \max_{\theta_2}\ \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\boldsymbol{\alpha},\sigma^2,{\theta_2}^{[k]}}\big[\log\big\{p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\alpha},\theta_1,\theta_2,\sigma^2)\big\}\big] \\ &= \arg\ \max_{\theta_2}\ \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\boldsymbol{\alpha},\sigma^2,{\theta_2}^{[k]}}\Big[\log\Big(\prod_{n=1}^N (2\pi\theta_2)^{-\frac{1}{2}} e^{\left\{-\frac{1}{2\theta_2}\left(\alpha_n - e^{\left\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0} - (\Sigma\Delta)|_{b(x_n,.)=1} - 1}{\theta_1}\right\}}\right)^2\right\}}\Big)\Big].\end{aligned}$$

Notice that in the above joint probability inside the expectation, we have discarded all the terms that were independent of θ_2. Therefore,

$$\begin{aligned}{\theta_2}^{[k+1]} &= \arg\ \min_{\theta_2}\ \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\theta_1,\boldsymbol{\alpha},\sigma^2,{\theta_2}^{[k]}}\big[N\log\{\theta_2\} + \frac{1}{\theta_2}\sum_{n=1}^{N}(\alpha_n - e^{\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0}-(\Sigma\Delta)|_{b(x_n,.)=1}-1}{\theta_1}\}})^2\big]\\ &= \arg\ \max_{\theta_2}\ f(\theta_2),\end{aligned}$$

where

$$f(\theta_2) = N\log\{\theta_2\} + \frac{1}{\theta_2}\sum_{n=1}^{N}(\alpha_n - e^{\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0}-(\Sigma\Delta)|_{b(x_n,.)=1}-1}{\theta_1}\}})^2.$$

For the minimization, we set $\frac{\partial f(\theta_2)}{\partial\theta_2} = 0$, which leads to

$$\frac{N}{\theta_2} = \frac{1}{\theta_2^2}\sum_{n=1}^{N}\big(\alpha_n - e^{\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0}-(\Sigma\Delta)|_{b(x_n,.)=1}-1}{\theta_1}\}}\big)^2.$$

Therefore,

$${\theta_2}^{[k+1]} = \frac{\sum_{n=1}^{N}\big(\alpha_n - e^{\{\frac{(\Sigma\Delta)|_{b(x_n,.)=0}-(\Sigma\Delta)|_{b(x_n,.)=1}-1}{\theta_1}\}}\big)^2}{N}. \tag{3.44}$$

CHAPTER 4
SPARSE BAYESIAN LEARNING BOOSTED BY PARTIAL ERRONEOUS SUPPORT KNOWLEDGE

4.1 Introduction

Recovery of sparse signals with unknown clustering pattern in the case of having partial erroneous prior knowledge on the supports of the signal is considered. In this case, we provide a modified sparse Bayesian learning model to incorporate prior knowledge and simultaneously learn the unknown clustering pattern. For this purpose, we add one more layer to support-aided sparse Bayesian learning algorithm (SA-SBL). This layer adds a prior on the shape parameters of Gamma distributions, those modeled to account for the precision of the solution elements. We make the shape parameters depend on the total variations on the estimated supports of the solution. Based on the simulation results, we show that the proposed algorithm is able to modify its erroneous prior knowledge on the supports of the solution and learn the clustering pattern of the true signal by filtering out the incorrect supports from the estimated support set.

Compressive sensing (CS) provides tools to represent a sparse or compressible signal from a small set of non-adaptive linear measurements [39]. In linear CS, the high dimensional signal $\boldsymbol{x} \in \mathbb{R}^N$ is modeled by the linear equation $\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$, where $A \in \mathbb{R}^{M\times N}$ is a wide sensing matrix with $M \ll N$. The case where the sensing matrix is known has been referred to as single measurement vector (SMV) problem [43]. In the CS context, it is assumed that $\boldsymbol{x}$ is sparse (has few non-zero elements) under some proper basis. Besides the sparsity, in some practical applications the nonzero entries of the sparse signal $\boldsymbol{x}$ may appear in clusters. This feature has been referred to as clustered-pattern or block-sparsity in the literature [15, 51]. Moreover, there exist cases where a partial erroneous support set of the solution is available as a prior knowledge. This type of information may be obtained from the followings. It can be the estimate of the supports inferred from the set of measurements taken from a phenomenon of interest at the last time instant. For example, in magnetoencephalography (MEG) for the brain activities and direction of arrival (DOA) estimation problems, it turns out that when taking successive measurements, the supports of the underlying signal remain almost constant or may experience very small variations [58, 100]. This problem has been treated as a multiple measurement vector (MMV) problem with slowly time-varying supports (sources) in the literature [101, 102].

Here, we investigate the sparse recovery problem of signals with unknown clustering pattern for the case where some prior information on the supports of the solution is available. More specifically, we assume that we are provided with a partial erroneous support set of the solution. One application of our proposed algorithm is that instead of solving an MMV with slowly time-varying supports directly, one can break the problem into a collection of SMVs and solve each SMV one at a time where they are now boosted by the support set estimated from the solution of the SMV of the previous time instant. As the second case, prior knowledge can be the output of some other CS recovery algorithm with the goal of performing some post processing to improve the overall performance in both increasing the success rate in support recovery and removing the number of indexes that have been incorrectly considered as the active supports of the solution. In both of the above cases, the available estimate of the support set usually contains a subset of the true supports accompanied with some other indexes that are incorrectly assumed to be active or they belonged to the estimated support set inferred from the last time instant set of measurements. This set of supports has been referred to as partial erroneous support set in the literature [52].

In case of having prior knowledge on the supports of the solution, some algorithms such as MBPDN and SA-SBL have been recently proposed [52, 103]. MBPDN algorithm is a modified version of basis pursuit de-noising algorithm for the case where a subset of true support set is available [103]. Recently, it has been shown [52] that MBPDN is sensitive to the accuracy of the prior knowledge on the available support set. In [52], Fang et al. proposed a modified version of the conventional sparse Bayesian learning model for the purpose of using prior information on the support set in order to obtain better estimate of the true underlying sparse signal. The conventional SBL algorithm considers a Gaussian-inverse-Gamma distribution on the elements of the solution vector $\boldsymbol{x}$. In [52], one more layer was added to the conventional SBL. This layer incorporates a prior on the rate parameter of the Gamma distribution to take advantage of the available support knowledge of the solution. Following the same notations as was used in [52], suppose that $\mathcal{T}$ is the set of all the true supports in the solution and $\mathcal{S} \subset \mathcal{T}$ is a subset of true supports that is available. Furthermore, assume that $\mathcal{E} \subset \mathcal{T}^c$ contains the error subset that is incorrectly considered as a part of available true supports. Notice that $\mathcal{T} \cup \mathcal{T}^c = \{1, 2, \ldots, N\}$ and $\mathcal{P} = \mathcal{S} \cup \mathcal{E}c$, where $\mathcal{P}$ is the erroneous support set that is available to us. As discussed earlier, one can think of $\mathcal{P}$ as the support set of the previous column of the solution matrix in the MMV problem.

Below, we briefly describe the priors that were considered in [52]. Each element x_n of the solution was assumed to be drawn i.i.d. from zero mean Gaussian distribution with corresponding precision α_n, i.e.,

$$x_n \sim \mathcal{N}(x_n; 0, \alpha_n^{-1}), \ \forall n = 1, \ldots, N, \tag{4.1}$$

where the precisions are random variables defined as

$$\alpha_n \sim \text{Gamma}(\alpha_n; a, b_n), \ \forall n = 1, \ldots, N, \tag{4.2}$$

where the shape parameter is fixed to $a = 10^{-10}$. In order to incorporate the available and probably erroneous support knowledge, Fang et al. [52] defined the two following cases on the rate parameters of the Gamma distributions

$$\begin{cases} b_n \sim \text{Gamma}(p, q), & \text{if } n \in \mathcal{P} \\ b_n = 10^{-15}, & \text{otherwise} \end{cases} \tag{4.3}$$

This means that only when the index n belongs to the set $\mathcal{P}$ the corresponding precision α_n will be governed by another Gamma distribution with hyper-parameters p and q. In this case, SA-SBL algorithm was proposed in [52].

The proposed algorithm is essentially a modified version of SA-SBL, in which we also account for the unknown clustering pattern that may exist in the original signal. For this purpose, we incorporate the measure of clumpiness over the supports of the solution $(\Sigma\Delta)$ proposed in [67] into SA-SBL algorithm. We refer to the proposed algorithm as CSA-SBL where the letter "C" stands for the "clustered" pattern signals. The main difference between our proposed algorithm and SA-SBL is that we further put a prior on the shape parameter of the Gamma distribution defined in (4.2) while it was set to a constant in SA-SBL [52]. Specifically, we impose that the shape parameter is to be controlled by the estimated measure of contiguity in the supports of the solution i.e., total variation on the supports of the solution. Based on the simulations, we show that this modification improves the overall performance in estimating the supports of the solution.

4.2 CSA-SBL Algorithm for Solving SMVs

In this section we describe our modified version of the conventional SBL algorithm to solve for $\boldsymbol{x}$ in the SMV problem defined by $\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$. It is assumed that an erroneous support set $\mathcal{P} = \mathcal{S} \cup \mathcal{E}$ is available, where $\mathcal{S}$ is a subset of the true supports of the solution and $\mathcal{E}$ is a set of incorrectly considered supports. The partition $\mathcal{S}$ and $\mathcal{E}$ in $\mathcal{P}$ is assumed unknown [52]. In order to account for the clustering pattern that may exist in the solution, we borrow the measure of clumpiness from [67], which is defined as

$$(\Sigma\Delta)_{(\text{support of } \boldsymbol{x})} = \sum_{n=2}^{N} |b(x_n, T) - b(x_{n-1}, T)|, \tag{4.4}$$

where T is a predetermined threshold. The function $b(\cdot,\cdot)$ in (4.4) returns a binary value and is defined as follows

$$b(x_n, T) = \begin{cases} 1 & \text{if } |x_n| > T \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

The more clustered the solution becomes, the lower value $(\Sigma\Delta)$ in (4.4) will possess. Based on (4.5), the entries of $\boldsymbol{x}$ with the amplitude less than the threshold T are set to zero and their corresponding index will not be considered as supports of the solution. This is due to the fact that the elements do not have significant contribution in our measurements. We experimentally set $T = 10^{-6}$.

Below we describe the prior distributions that we consider in our hierarchical Bayesian model. Similar to SBL and for the purpose of promoting sparsity in the solution, we assume that the elements of the solution are drawn i.i.d. from a zero-mean Gaussian distribution as follows

$$x_n \sim \mathcal{N}(x_n; 0, \alpha_n^{-1}),\ \forall n = 1, \ldots, N, \qquad \text{(Revisiting (4.1))}$$

where the precision α_n is modeled as

$$\alpha_n \sim \text{Gamma}(\alpha_n; a_n, b_n),\ \forall n = 1, \ldots, N. \tag{4.6}$$

Unlike (4.2), we do not assign the same shape parameter to the precisions α_n, $\forall n = 1, \ldots, N$ in our model.

In order to incorporate the available erroneous support knowledge, we use the same model as defined [52] for the rate parameters defined in (4.6).

$$\begin{cases} b_n \sim \text{Gamma}(b_n; p, q), & \text{if } n \in \mathcal{P} \\ b_n = 10^{-15}, & \text{otherwise} \end{cases}, \qquad \text{(Revisiting (4.3))}$$

where $p = q = 0.1$ as suggested in [52].

The reason for having different shape parameters for the precisions α_n in (4.6) is to promote the clustered pattern in the support set of the solution, where the pattern is learned the via measure of total variation on the supports defined in (4.4). In other words, we let each shape parameter a_n be controlled via the estimated $(\Sigma\Delta)$. For this purpose, we add another hyper-prior to our model as follows

$$a_n \sim \text{Gamma}(a_n; g_n, h),\ \forall n = 1, \ldots, N, \tag{4.7}$$

where

$$g_n := \theta \exp\left\{\left((\Sigma\Delta) - (\Sigma\Delta)_{n,0}\right)\right\}, \tag{4.8}$$

where $(\Sigma\Delta)$ is the initial measure of the clumpiness based on the available erroneous support set $\mathcal{P}$ and is computed from (4.4), and $(\Sigma\Delta)_{n,0}$ is the measure of clumpiness when forcing $x_n = 0$. In (4.8), θ is an emphasizing parameter on the amount of clumpiness over the supports of the solution and one can make it depend on the ratio M/N. This means that when the number of measurements is very low, we may not wish to emphasize on the clustered solutions. Otherwise, the algorithm may also remove some of the true supports in the set $\mathcal{P}$ due to the small number of measurements and the lack of information on the full true support set.

Remark 4.1: In the simulations, we set $h = 1$ but in general, "h" is selected based on the belief on the maximum permissible amplitude of the elements of $\boldsymbol{x}$.

Finally, the prior on the noise is defined as

$$\sigma^2 \sim \mathcal{N}(0, \gamma^{-1}), \gamma \sim \text{Gamma}(c, d), \tag{4.9}$$

where we set $c = d = 10^{-4}$ as suggested in [52].

According to the above prior distributions, the joint probability distribution of our model becomes

$$\begin{aligned} p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b}, \sigma^2, \gamma) \propto & \, p(\boldsymbol{y}|\boldsymbol{x}, \sigma^2 I_M) p(\boldsymbol{x}; 0, \boldsymbol{\alpha}^{-1}) \times \\ & p(\boldsymbol{\alpha}; \boldsymbol{a}, \boldsymbol{b}) P(\boldsymbol{a}; \boldsymbol{g}, h) p(\sigma^2; 0, \gamma^{-1}) p(\gamma; c, d) \prod_{n \in \mathcal{P}} p(b_n; p, q), \end{aligned} \tag{4.10}$$

where the measurement noise is assumed to be $\boldsymbol{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_M)$.

According to (4.10), the marginalized posterior distributions for the variables of interest can be represented as follows. In these descriptions, conditioning on $-$, as in $(x_n|-)$, denotes the inference on x_n conditioning upon all relevant variables and the observations.

- $p(\boldsymbol{x}|-) \propto p(\boldsymbol{y}|\boldsymbol{x}, \sigma^2 I_M)p(\boldsymbol{x}|\boldsymbol{\alpha}^{-1}I_N)$. Therefore,

$$(\boldsymbol{x}|-) \sim \mathcal{N}(\boldsymbol{\mu}_{x|-}, \Sigma_{x|-}), \tag{4.11}$$

where

$$\boldsymbol{\mu}_{x|-} = \gamma\Sigma_{x|-}A^T\boldsymbol{y}, \Sigma_{x|-} = (\gamma A^T A + D)^{-1},$$

where D is a diagonal matrix with $\boldsymbol{\alpha}$ as its main diagonal i.e., $[D]_{n,n} = \alpha_n$.

- $p(\boldsymbol{\alpha}|-) \propto p(\boldsymbol{x}|\boldsymbol{\alpha}^{-1}I_N)p(\boldsymbol{\alpha}; \boldsymbol{a}, \boldsymbol{b})$. Therefore,

$$(\alpha_n|-) \sim \begin{cases} \text{Gamma}(\bar{a}_n, b_n + \frac{\mu^2_{x_n|-} + \sigma^2_{x_n|-}}{2}), \text{ if } n \in \mathcal{P} \\ \text{Gamma}(\bar{a}_n, 10^{-15} + \frac{\mu^2_{x_n|-} + \sigma^2_{x_n|-}}{2}), \text{ otherwise} \end{cases}, \tag{4.12}$$

In the above equation, $\bar{a}_n := a_n + \frac{1}{2}$ and $\sigma^2_{x_n|-} := \alpha_n^{-1}$.

- $P(a_n|-) \propto P(\alpha_n; a_n, b_n)P(a_n|g_n, h), \forall n = 1, \ldots, N$. Hence

$$(a_n|-) \sim \text{Gamma}(g_n, h + \log \alpha_n),\ \forall n = 1, \ldots, N, \tag{4.13}$$

where $g_n = \theta \exp\{(\hat{\Sigma\Delta})^{[k]} - (\hat{\Sigma\Delta})^{[k]}_{n,0}\}$ and k denotes the kth iteration and $(\hat{\Sigma\Delta})$ is the estimated measure of clumpiness.

- $p(b_n|-) \propto p(\alpha_n; a_n, b_n)p(b_n; p, q), \forall n \in \mathcal{P}$. As a result,

$$(b_n|-) \sim \text{Gamma}(p, q + \alpha_n),\ \forall n \in \mathcal{P}. \tag{4.14}$$

- Finally, $p(\gamma|-) \propto p(\boldsymbol{y}|\boldsymbol{x}, \gamma)p(\gamma; c, d)$ and therefore,

$$(\gamma|-) \sim \text{Gamma}\Big(c + \frac{M}{2}, d + \frac{1}{2}\big(\|\boldsymbol{y} - A\boldsymbol{\mu}_{x|-}\|_2^2 + \text{tr}(A^T A\Sigma_{x|-})\big)\Big). \tag{4.15}$$

4.3 Simulation Results

In this section, the performance of the proposed algorithm is compared against the SA-SBL algorithm proposed in [52]. For simulation purposes, the supports of the solution are randomly drawn from a Bernoulli distribution in such a way to exhibit a random clustered-sparsity pattern. In all the simulations, the number of true supports is set to $|\mathcal{T}| = 25$. The non-zero elements of solution vector $\boldsymbol{x}$, corresponding to true supports, are drawn i.i.d. from Gaussian distribution with zero-mean and variance $\sigma_x^2 = 1$. The sensing matrix A is $M \times 100$ and is randomly drawn from $\mathcal{N}(0, 1)$ and then it is normalized with respect to its columns. We vary the number of measurements M from 5 up to N to evaluate the performance. The elements of the noise component are drawn i.i.d. from $e_n \sim \mathcal{N}(0, \sigma_n^2)$. In all the simulations, we set SNR= 25 dB. Finally, the measurement vector $\boldsymbol{y}$ is computed from $\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{e}$. For each run we randomly select 80% of the true supports and collect them in the set $\mathcal{S}$. Then, 10 more indexes that are not in the true supports are randomly chosen and collected into $\mathcal{E}$. We then feed both our algorithm and SA-SBL with an erroneous support set $\mathcal{P} = \{\mathcal{S} \cup \mathcal{E}\}$. Neither of the algorithms is aware of the number of true supports in set $\mathcal{P}$. For each value of M, we run 500 random cases and then average over all of the obtained results both for our algorithm and SA-SBL.

In Fig. 4.1, we compare the detection rate (P_D) in the support recovery between our algorithm and SA-SBL.

It can be seen in Fig. 4.1 that when the ratio $M/N = 0$, the detection rate $P_D = 0.8$ and it is due to the fact that we provided both algorithms with 80% of the true supports. SA-SBL algorithm provides better performance compared to CSA-SBL algorithm and the reason is it does not wish to remove its prior knowledge. It worth noting that statistically speaking, CSA-SBL provides better performance in cases of having less prior knowledge on the supports.

In Fig. 4.2, we compare the false alarm rate (P_{FA}) in the support estimation between our algorithm and SA-SBL. According to the results shown in Fig. 4.2, we see that our algorithm has a very low false alarm rate, much lower than the one obtained from SA-SBL.

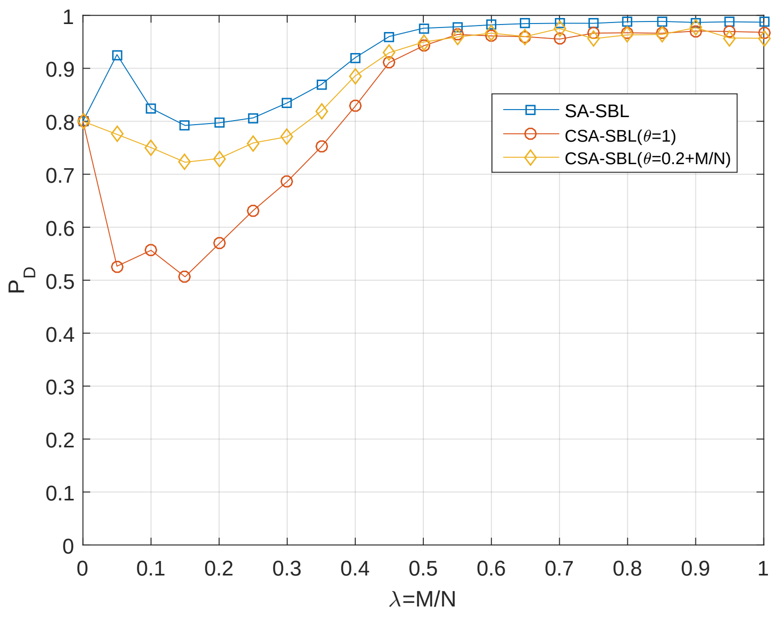


Fig. 4.1: Detection rate comparison.

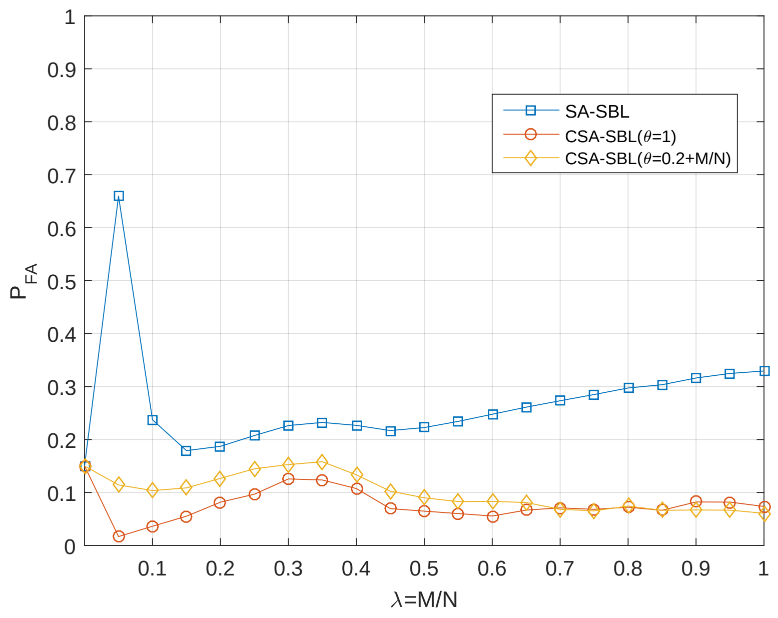


Fig. 4.2: False alarm rate comparison.

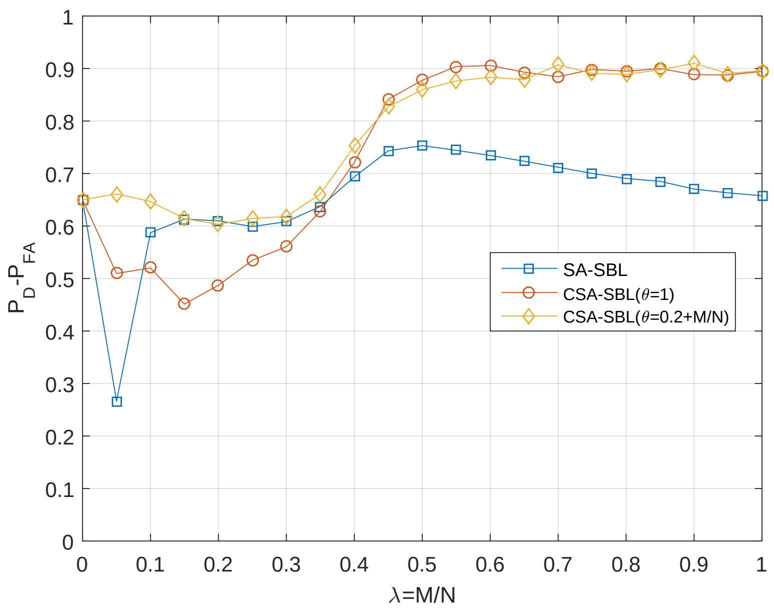


Fig. 4.3: An overall measure of performance in terms of both detection rate and false alarm rate.

The performance is essentially an evidence of the role of incorporating the measure of $(\Sigma\Delta)$ to account for the unknown clustering pattern in the solution. In other words, our algorithm is able to filter out those estimated supports, that cause less clustering solutions among other possible solutions.

In Fig. 4.3 we illustrate another measure of performance which is based on the difference between the detection and false alarm rate in the estimated support set. This figure essentially illustrates the overall performance of the algorithms based on P_D, P_{FA}, and M/N, all together. Finally, in Fig. 4.4, we demonstrate the performance of our modified version of SA-SBL compared to SA-SBL in terms of success rate in estimating the true clustering pattern. Our measure of success, P_r, is defined as follows

$$P_r = 1 - \frac{|(\hat{\Sigma\Delta}) - (\Sigma\Delta)|}{(\Sigma\Delta)},$$

where $(\Sigma\Delta)$ and $(\hat{\Sigma\Delta})$ are the total number of transitions in the true support set and the estimated solution, respectively. It is clear from Fig. 4.4 that our propose modified version of SA-SBL (CSA-SBL) outperforms SA-SBL and that is because of the role of incorporating

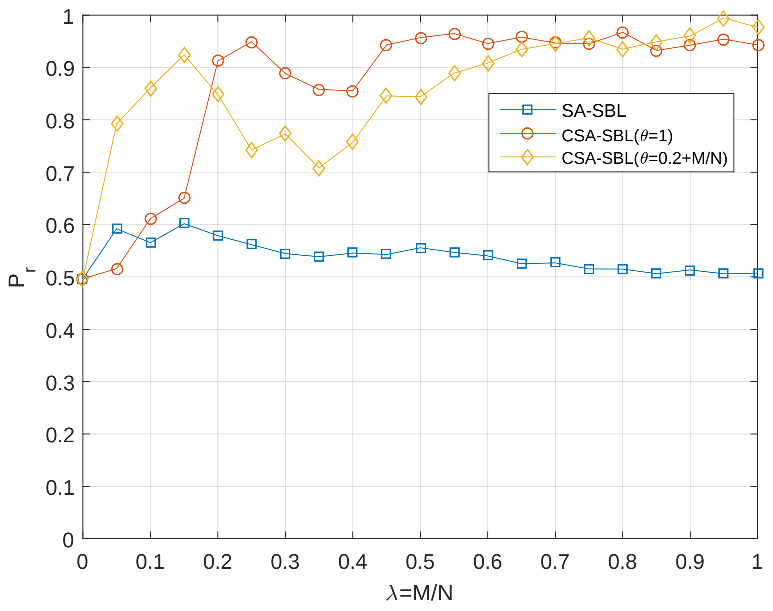


Fig. 4.4: Comparison in terms of pattern recovery.

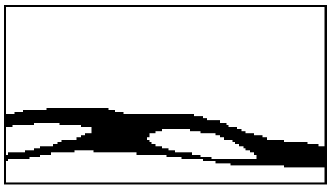


Fig. 4.5: True image.

the measure of clumpiness in our algorithm.

Finally, the performance of SA-SBL and CSA-SBL are also compared algorithms via the following example. In this case we use the image of 112×200 pixels as illustrated in Fig. 4.5. In this image, we treat the black pixels as the "interesting" locations.

For the simulation purposes, we assign the value of 1 to the pixels with black color and 0 to the white ones. We then, assume that the matrix corresponding to the image is the true solution matrix $X \in \mathbf{R}^{112\times 200}$ of an MMV problem, where the supports of the solution (sources) slowly change across the columns of X. We construct the sensing matrix in the same way we described earlier in this section. The measurements are obtained from $Y = AX + E$ with SNR=25 dB. We then break the problem into $M = 112$ SMVs, where the available information to the "m"th SMV is matrix A, the mth column of Y, and the

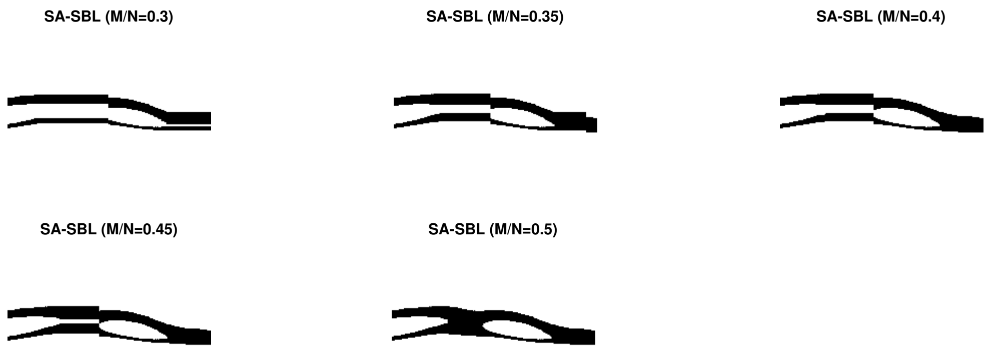


Fig. 4.6: Results of reconstructed images for SA-SBL for different sampling ratios (M/N).

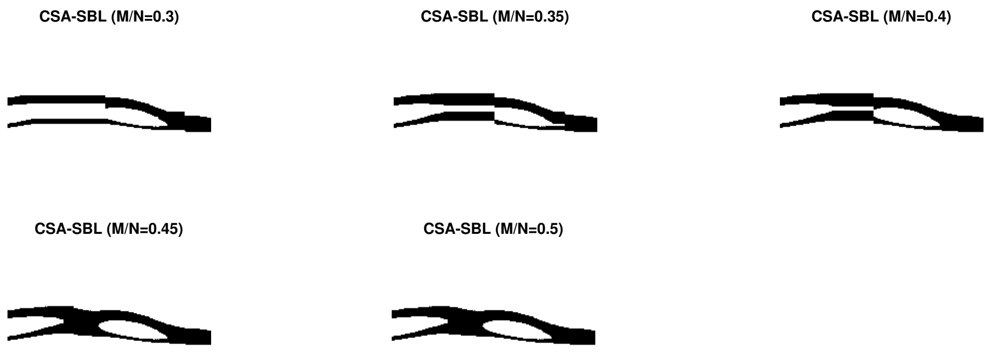


Fig. 4.7: Results of reconstructed images for CSA-SBL for different sampling ratios (M/N).

estimated supports obtained from solving the SMV corresponding to the $(m-1)$st column of X. As illustrated in Fig. 4.5, the support set changes slowly across the columns of X. Therefore, even if we were provided with the true supports of the $(m-1)$st column of X, the information still would become the erroneous partial support set that was available for solving the "m"th SMV. In order to solve the SMV corresponding to the first column of X, we feed both SA-SBL and CSA-SBL with a union of 80% of the true supports of the column and 10 randomly chosen incorrect supports. In the simulations, we set the threshold to $T = 0.4$ for both SA-SBL and CSA-SBL, meaning that we discarded the estimated supports with corresponding absolute value of less than 0.4. The reconstructed images for the sampling ratio of $M/N = 0.3, 0.35, 0.4, 0.45$, and 0.5 are illustrated in Fig. 4.6 and Fig. 4.7. In order

Table 4.1: NMSE and PSNR comparison in image reconstruction.

	SA-SBL	**CSA-SBL**
Sampling ratio (M/N)	**PSNR (dB)**	**PSNR (dB)**
$M/N = 0.30$	12.8467	13.5155
$M/N = 0.35$	15.1456	15.8384
$M/N = 0.40$	16.4610	18.1877
$M/N = 0.45$	18.6170	30.9498
$M/N = 0.50$	Inf	Inf

to show the quality of the reconstructed images we compare the obtained results based on peak-SNR (PSNR) as shown in Table 4.1. According to Table 4.1, we observe that CSA-SBL provides better reconstruction in terms of PSNR compared to SA-SBL algorithm.

4.4 Summary

A new hierarchical Bayesian model is proposed to solve the clustered-pattern sparse signals via single measurement vector model in case where some erroneous information about the support set of the solution is available. The proposed algorithm is essentially a modified version of the conventional sparse Bayesian learning model and SA-SBL algorithm. The simulation results demonstrated that making the shape parameter of the Gamma distribution in the SA-SBL algorithm for the problem improves the performance in the support recovery.

CHAPTER 5

SPARSE RECOVERY VIA VARIATIONAL BAYESIAN INFERENCE: BERNOULLIS-GAUSSIANS-INVERSE GAMMA *VS.* GAUSSIANS-INVERSE GAMMAS MODELING

5.1 Introduction

In this chapter, we investigate the performance of sparse signal recovery from a set of compressively sensed noisy measurements using variational Bayesian (VB) inference. The framework considered here is ordinary sparse Bayesian learning where no specific structure other than sparsity is assumed on the underlying signal of interest. Two models on the signal are considered here, the issues of each model are studied, and the performances in reconstruction are compared. In the first model, the sparse signal is considered as the combination of a solution vector, where each component is modeled as a Gaussian distribution with the same precision, and the support vector, where each component is modeled by a Bernoulli distribution. This model is denoted by Bernoullis-Gaussians-inverse Gamma modeling. In the second model, the components of the solution are modeled by Gaussian distributions but with different precisions, referred to as Gaussians-inverse Gammas modeling. Although these two models have been already applied to the compressive sensing problems using sparse-Bayesian learning via expectation-maximization (EM) algorithm, Markov chain Carlo (MCMC), etc., for structured solutions, the issues when using variational Bayes inference for only promoting the sparsity have not been investigated. This work provides details on such modelings using VB inference.

Compressive sensing (CS) provides techniques for signal acquisition and reconstruction in a sub-Nyquist sampling sense. The goal in the CS framework is to capture the important information of the underlying signal via a small number of measurements while retaining the ability to reconstruct the signal. CS operates under the assumption that the signal is compressible or sparse, where the number and location of dominating non-zeros are unknown [17, 39, 40]. The notion of compressibility or sparsity simply means that under some proper basis representation the signal has very few non-zero elements while the other elements, the majority of components in the signal, are either zero or of very small amplitudes compared to the dominating amplitudes. Therefore, the small amplitudes, which are of low interest in most practical applications, can be treated as noise. CS has been used in a variety of applications such as in single-pixel camera, recovery of images with missing pixels and inpainting removal, medical imaging using magnetic resonance imaging (MRI), communications with examples of blind multiband sampling of narrow band signals, sparse channel estimation, spectrum sharing of radar and communication signals, and many more [1, 2, 8, 15, 16, 57, 59, 104–111].

In the linear CS framework the problem is posed as

$$\boldsymbol{y} = A\boldsymbol{x}_s + \boldsymbol{e}, \tag{5.1}$$

where $\boldsymbol{y} \in \mathbb{R}^M$ represents measurements, $\boldsymbol{x}_s \in \mathbb{R}^N$ is the sparse signal, $\boldsymbol{e}$ is noise accounting for either the measurement noise or the insignificant coefficients of $\boldsymbol{x}_s$, and generally $M \ll N$ [17, 39]. The matrix $A = \Phi\Psi$ is the measurement matrix, where Φ is the sensing design matrix and Ψ is a proper sparsifying basis. There exist different approaches to solve for $\boldsymbol{x}_s$ in the inverse problem of linear CS including greedy-based, convex-based, and sparse Bayesian learning (SBL) algorithms [37, 44, 49, 50, 69, 112–115]. Typically, the performance of CS reconstruction is determined in terms of mean-squared reconstruction error. We are also interested in the more strenuous requirements of probability of detection and probability of false alarm of the nonsparse components.

The focus of this work is on sparse Bayesian learning (SBL) for the CS problem. The advantage of Bayesian learning models is that one can incorporate prior knowledge on the characteristics of the underlying signal into the model. A prior favoring the sparsity or compressibility in $\boldsymbol{x}$ can be represented to the framework by using Gaussian-inverse Gamma (denoted by GiG), Laplace, Bernoulli-Gaussian (denoted by BG), spike-and-slab priors, and so forth [60,66,68,69,116]. Inference on parameters and variable estimation on these models can be made using Markov chain Monte Carlo (MCMC), approximate message passing (AMP) and its generalized versions, variational Bayesian (VB) inference, etc. [37, 67, 68, 73, 117–121]. Here, we study two models for solving the inverse problem of compressive sensing; Bernoullis-Gaussians-inverse Gamma (BGiG) prior and Gaussians-inverse Gammas (GiG) prior to promote sparsity in the solution. We use variational Bayesian (VB) inference to estimate the variables and parameters of the model. The reason for preferring VB to the MCMC inference is that although MCMC usually provides good inference, it is not computationally efficient and the convergence diagnostic may be difficult or at least require extra work for measuring the convergence, such as potential scale reduction factor (PSRF) [68, 92, 114, 122, 123]. In contrast, it has been shown that VB inference in many cases leads to reasonable approximation of the exact posteriors more efficiently than MCMC, with less effort to monitor the convergence [122–125]. For both BGiG and GiG models, we represent the update rules of the parameters and variables using VB inference, provide discussions on the issues associated with each model, and illustrate some motivational examples. We then describe how to improve the reconstruction performance for BGiG model via training the hyperparameters of the model. Also, we describe a learning approach to improve the performance of the algorithm associated with the GiG model.

The rest of this work is organized as follows. Section 5.2 presents a brief background on VB inference. In Section 5.3, we provide Bernoullis-Gausssians-inverse Gammas modeling for CS using VB. Also, some motivational examples are provided to show the issue with this approach. Section 5.4 represents Gaussians-inverse Gammas modeling, the associated update rules using VB inference, and motivational examples on the issues with this algorithm. Section 5.5 describes how to improve the performance for each algorithm. Finally, Section 5.6 concludes this work.

5.2 Variational Bayesian Inference

Variational Bayes is an effective approach to approximate intractable integrals that occur in Bayesian inference. VB provides analytical approximation to the posterior distributions of the parameters and hidden variables of statistical models using a lower bound on the marginal likelihood of the observations. VB is essentially an extension of the expectation-maximization (EM) algorithm [124,125]. Suppose there is a probabilistic model with parameters Θ, hidden variables $\boldsymbol{x}$, and a set of observations denoted by $\boldsymbol{y}$. Then, the approximation to the joint density $p(\boldsymbol{x}, \Theta|\boldsymbol{y})$ can be represented by $p(\boldsymbol{x}, \Theta|\boldsymbol{y}) \approx q_x(\boldsymbol{x}) q_\theta(\Theta)$. In the ideal case, we desire to select $Q_{x,\theta}(\boldsymbol{x}, \Theta) = q_x(\boldsymbol{x}) q_\theta(\Theta)$ to be as close as possible to $p(\boldsymbol{x}, \Theta|\boldsymbol{y})$. Since computing the normalization factor (probability of the observation $p(\mathbf{y})$) is intractable, we write the logarithm of the evidence in terms of the integral of the joint probability $p(\boldsymbol{x}, \Theta, \boldsymbol{y})$ and then incorporate $Q_{x,\theta}(\boldsymbol{x}, \Theta)$ into the integrand. Using Jensen's inequality, we obtain a lower bound on the logarithm of evidence. The problem turns into maximizing this lower bound to make $Q_{x,\theta}(\boldsymbol{x}, \Theta)$ close to $p(\boldsymbol{x}, \Theta|\boldsymbol{y})$. The lower bound on the model log-marginal likelihood can be iteratively optimized by the following updates [124, 126]

$$q_x^{[t+1]}(\boldsymbol{x}) \propto \exp\left\{\mathrm{E}_{q_\theta^{[t]}}[\log p(\boldsymbol{x}, \boldsymbol{y}|\Theta)]\right\} \tag{5.2}$$

$$q_\Theta^{[t+1]}(\Theta) \propto p(\Theta) \exp\left\{\mathrm{E}_{q_x^{[t]}}[\log p(\boldsymbol{x}, \boldsymbol{y}|\Theta)]\right\}. \tag{5.3}$$

5.3 Bernoullis-Gaussians-Inverse Gamma Modeling and SBL-VB(BGiG) Algorithm

The goal in the inverse problem of CS defined in (5.1) is to recover the sparse vector $\boldsymbol{x}_s$. In the Bernoullis-Gaussian-inverse Gamma model, the sparse solution is defined as $\boldsymbol{x}_s = (\boldsymbol{s} \circ \boldsymbol{x})$, where $\boldsymbol{s}$ is a binary support vector indicating the non-zero locations in the solution, $\boldsymbol{x}$ represents values of the solution, and $\circ$ is Hadamard product [37]. We refer to the algorithm associated with this Bayesian modeling inferred via VB as ordinary SBL-VB(BGiG). SBL using VB inference for clustered pattern of sparse signal has already been investigated in the recent literature [114, 122, 123]. Here, we consider the ordinary SBL using VB inference modeling without promoting any structure on the supports other than sparsity itself. We show that when the number of measurements is small the reconstruction performance is sensitive to the selection of the support-related hyperparameters.

For the inverse CS problem, we define a set of priors as follows [37]. We model the elements of vector $\boldsymbol{s}$ as

$$s_n \sim \text{Bernoulli}(\gamma_n), \gamma_n \sim \text{Beta}(\alpha_0, \beta_0), \forall n,\ \alpha_0 \ll \beta_0 \tag{5.4}$$

α_0 and β_0 are the support-related hyperparameters. Setting α_0 and β_0 to small values and with $\alpha_0 \ll \beta_0$ encourages $\boldsymbol{s}$ to be sparse on average. The prior on the solution value vector is defined as

$$\boldsymbol{x} \sim \mathcal{N}(0, \tau^{-1} I_N), \quad \tau \sim \text{Gamma}(a_0, b_0). \tag{5.5}$$

Here, τ is the value precision. Finally, the prior on the noise is

$$\boldsymbol{e} \sim \mathcal{N}(0, \varepsilon^{-1} I_M), \quad \varepsilon \sim \text{Gamma}(\theta_0, \theta_1). \tag{5.6}$$

where θ_0 and θ_1 are set to small positive values.

5.3.1 Update Rules of SBL-VB(BGiG) Using VB Inference

According to the VB algorithm defined in (5.2) and (5.3), the update rule of the variables and parameters of the BGiG model can be simplified as follows [120].

- Update rule for the support vector $\boldsymbol{s}$

$$q(s_n|-) \sim \text{Bernoulli}\left(\frac{1}{1 + c_n \kappa_n}\right),\ \forall n = 1, \ldots, N,$$

(here, conditioning on $-$ denotes conditioning on all relevant variables and observations) and therefore

$$\tilde{s}_n = \frac{1}{1 + c_n \kappa_n},\ \forall n = 1, \ldots, N, \tag{5.7}$$

where

$$c_n := e^{\psi(\beta_{1,n})-\psi(\alpha_{1,n})},$$
$$\kappa_n := e^{\frac{1}{2}\tilde{\varepsilon}\left(\|_n\|_2^2(\tilde{x_n}^2+\sigma_{\tilde{x_n}}^2)-2\tilde{x_n}\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n}\right)},$$
$$\tilde{y}_m^{-n} := y_m - \sum_{l\neq n}^{N} a_{ml}\tilde{s}_l\tilde{x}_l,$$

and where ψ is the digamma function, which is the logarithmic derivative of the gamma function, and $\tilde{\boldsymbol{y}}^{-n} = [\tilde{y}_1^{-n}, \ldots, \tilde{y}_M^{-n}]^T$.

- Update rule for the solution value matrix $\boldsymbol{x}$

$$q(\boldsymbol{x}|-) \sim \mathcal{N}(\tilde{\boldsymbol{x}}, \Sigma_{\tilde{x}}), \text{ where}$$

$$\Sigma_{\tilde{x}} = \left(\tilde{\tau} I_N + \tilde{\varepsilon}\tilde{\Phi}\right)^{-1} \text{ and } \tilde{\boldsymbol{x}} = \tilde{\varepsilon}\Sigma_{\tilde{x}}\text{diag}(\tilde{\boldsymbol{s}})A^T\boldsymbol{y}, \tag{5.8}$$

where

$$\tilde{\Phi} := \left[(A^TA) \circ \left(\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \text{diag}(\tilde{\boldsymbol{s}} \circ (1-\tilde{\boldsymbol{s}}))\right)\right]. \tag{5.9}$$

- Update rule for γ_n

$$q(\gamma_n|-) \sim \text{Beta}(\alpha_{1,n}, \beta_{1,n}), \ \forall n = 1, \ldots, N.$$

Therefore,

$$\tilde{\gamma_n} = \frac{\alpha_{1,n}}{\alpha_{1,n} + \beta_{1,n}}, \ \forall n = 1, \ldots, N, \tag{5.10}$$

where $\alpha_{1,n} := \alpha_0 + \tilde{s}_n$ and $\beta_{1,n} := \beta_0 + 1 - \tilde{s}_n$.

- Update rule for the solution precision τ

$$q(\tau|-) \sim \text{Gamma}\big(a_0 + \frac{N}{2}, b_0 + \frac{1}{2}(\|\tilde{\boldsymbol{x}}\|_2^2 + \text{Tr}\,(\Sigma_{\tilde{x}}))\big),$$

where $\Sigma_{\tilde{x}} = \text{diag}(\sigma_{\tilde{x}_1}^2, \ldots, \sigma_{\tilde{x}_N}^2)$. Thus

$$\tilde{\tau} = \frac{a_0 + \frac{N}{2}}{b_0 + \frac{1}{2}\left(\|\tilde{\boldsymbol{x}}\|_2^2 + \sum_{n=1}^{N}\sigma_{\tilde{x}_n}^2\right)}. \tag{5.11}$$

- Update rule for the noise precision ε

$$q(\varepsilon|-) \sim \text{Gamma}(\theta_0 + \frac{M}{2}, \theta_1 + \frac{1}{2}\tilde{\Psi}).$$

This yields to the following update rule for the precision on the noise component.

$$\tilde{\varepsilon} = \frac{\theta_0 + \frac{M}{2}}{\theta_1 + \frac{1}{2}\tilde{\Psi}}, \tag{5.12}$$

where $\tilde{\Psi} := \Big(\boldsymbol{y}^T\boldsymbol{y} - 2(\tilde{\boldsymbol{x}} \circ \tilde{\boldsymbol{s}})^T A^T \boldsymbol{y} + \text{Tr}\left((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}})\tilde{\Phi}\right)\Big)$, $\tilde{\Phi}$ was defined earlier in (5.9).

The stopping criterion of the algorithm can be made based on the log-marginalized likelihood. Learning $\boldsymbol{s}$ is probably the most important variable to us, since if we could learn $\boldsymbol{s}$ accurately then it would not be hard to compute $\boldsymbol{x}_s$. We define the stopping condition in terms of $L := \log\{p(\boldsymbol{y}|\boldsymbol{s}, \varepsilon, \tau)\}$. The marginalized likelihood can be written as $p(\boldsymbol{y}|\boldsymbol{s}, \varepsilon, \tau) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}, \varepsilon) p(\boldsymbol{x}|\tau I_N) d\boldsymbol{x}$. After some simplification, the negative log-likelihood is proportional to

$$-L \propto \log|\Sigma_0^{-1}| + \boldsymbol{y}^T \Sigma_0 \boldsymbol{y}, \tag{5.13}$$

where $\Sigma_0 = (\varepsilon^{-1} I_M + \tau^{-1} A\tilde{S}^2 A^T)^{-1}$, and $\tilde{S} := \text{diag}\{\tilde{\boldsymbol{s}}\}$. Therefore, the stopping condition can be made as

$$|L^{[t+1]} - L^{[t]}|/|L^{[t]}| \leq T_0,$$

for some small value of threshold T_0 [114]. We refer to the algorithm corresponding to the above modeling as SBL-VB(BGiG) denoting that this is an ordinary sparse Bayesian learning algorithm inferred via VB using Bernoullis-Gaussians-inverse Gamma modeling. By the term ordinary, we mean that the algorithm only promotes sparsity and it does not incorporate any parameter to account for the clustered pattern supports or correlation among the elements of the solution. The pseudocode of the algorithm is provided below.

SBL-VB(BGiG) Algorithm:

```
$[\hat{\boldsymbol{x}}, \hat{\boldsymbol{s}}]$ = **SBL-VB-BGiG**$(Y, A)$
Iter= 1
While $\frac{|L^{[\text{Iter}]} - L^{[\text{Iter}-1]}|}{|L^{[\text{Iter}-1]}|} \geq 10^{-6}$
    Compute $\tilde{s_n}$ from (5.7), $\forall n = 1, \ldots, N$ % (Support vector component )
    Compute $\Sigma_{\tilde{x}}$ and $\tilde{\boldsymbol{x}}$ from (5.8)    % (Solution-value matrix component)
    Compute $\boldsymbol{\alpha}_1$ and $\boldsymbol{\beta}_1$ from (5.10)    % (Parameters of the hyperprior $\boldsymbol{\gamma}$)
    Compute $\tau$ from (5.11)    % (Precision on the solution)
    Compute $\varepsilon$ from (5.12)    % (Precision on the noise)
    Compute $L^{[\text{Iter}]}$ from (5.13) and then Iter=Iter+1
End While
```

5.3.2 Issues with SBL-VB(BGiG) and Study Examples

In this section, we show that the estimated solution using SBL-VB(BGiG) algorithm is sensitive to support-related hyperparameters, i.e., α_0 and β_0 in (5.4). We provide three examples to demonstrate the issues associated with SBL-VB(BGiG). We generated three trials, where the true solution $\boldsymbol{x} \in \mathbf{R}^{100}$ for each trial has the sparsity level of $k := 25$, that is, the true $\boldsymbol{x}$ (or $\boldsymbol{s}$) has k active elements. The active elements of $\boldsymbol{s}$ were drawn randomly in our trials. The nonzeros of $\boldsymbol{x}$, corresponding to the active locations of $\boldsymbol{s}$, were drawn from $\mathcal{N}(0, \sigma_x^2)$, with $\sigma_x^2 = 1$. Each entry of the sensing matrix A was drawn *i.i.d.* from a Gaussian distribution $\mathcal{N}(0, 1)$, and then normalized so each column has Euclidian norm 1. The elements of measurement noise were drawn from $\mathcal{N}(0, \sigma^2)$ with SNR=25 dB, where SNR:$= 20\log_{10}(\sigma_x/\sigma)$. The hyperparameters were set to $a_0 = b_0 = 10^{-3}$ and $\theta_0 = \theta_1 = 10^{-6}$. In examples 1-3 we set the pair (α_0, β_0) to $(1.4, 2)$, $(0.1, 0.9)$, and $(0.01, 0.99)$, respectively. From the top to the bottom row of each of Figs. 5.1–5.3, we illustrate the performance for the cases where the number of measurements is set to 85, 75, and 65 (that is, the sample ratio λ is 0.85, 0.75, and 0.65) respectively. In each row of Figs. 5.1–5.3 from left to right are, respectively: the comparison between the measurements $\boldsymbol{y}$ and the computed measurements based on $\hat{\boldsymbol{y}} = A(\tilde{\boldsymbol{s}} \circ \tilde{\boldsymbol{x}})$; the true signal $\boldsymbol{x}_s = \boldsymbol{s} \circ \boldsymbol{x}$ and the reconstructed signal $\tilde{\boldsymbol{x}} = \hat{\boldsymbol{s}} \circ \hat{\boldsymbol{x}}$; the true support vector $\boldsymbol{s}$ and the estimated support vector $\tilde{\boldsymbol{s}}$; and the estimated supports *vs.* the iterations.

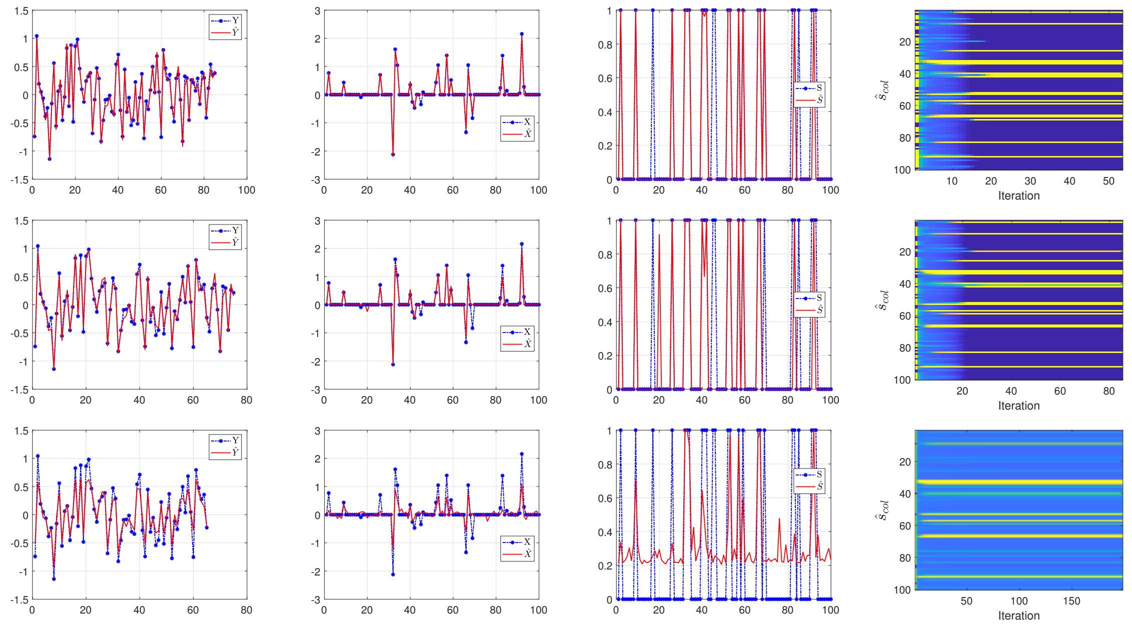


Fig. 5.1: Example 1: $(\alpha_0, \beta_0) = (1.4, 2)$. From top to bottom, the rows show the results of SBL-VB(BGiG) for the sampling ratio $\lambda = 0.85, 0.75, 0.65$, respectively. The term $hats_{coll}$ is the estimated support vector in each iteration of the algorithm.

According to the results shown in Fig. 5.1, setting (α_0, β_0) to $(1.4, 2)$ seems to be a reasonable choice for high sampling ratios (over 70%), while it is not a good choice for the lower sampling ratios. This issue can be seen in the plot of the supports in the 3rd row of Fig. 5.1. One may argue that the estimated support vector $\hat{\boldsymbol{s}}$ can be filtered via some threshold value (such as 0.3), meaning that one may discard the estimated supports lower than 0.3. However, thresholding will adversely affect detection rate. Also, we should account for the effect of the filtered supports since their corresponding estimated components in $\hat{\boldsymbol{x}}$ have contribution in fitting the model to the measurements, as well. In Table 5.1, we summarize

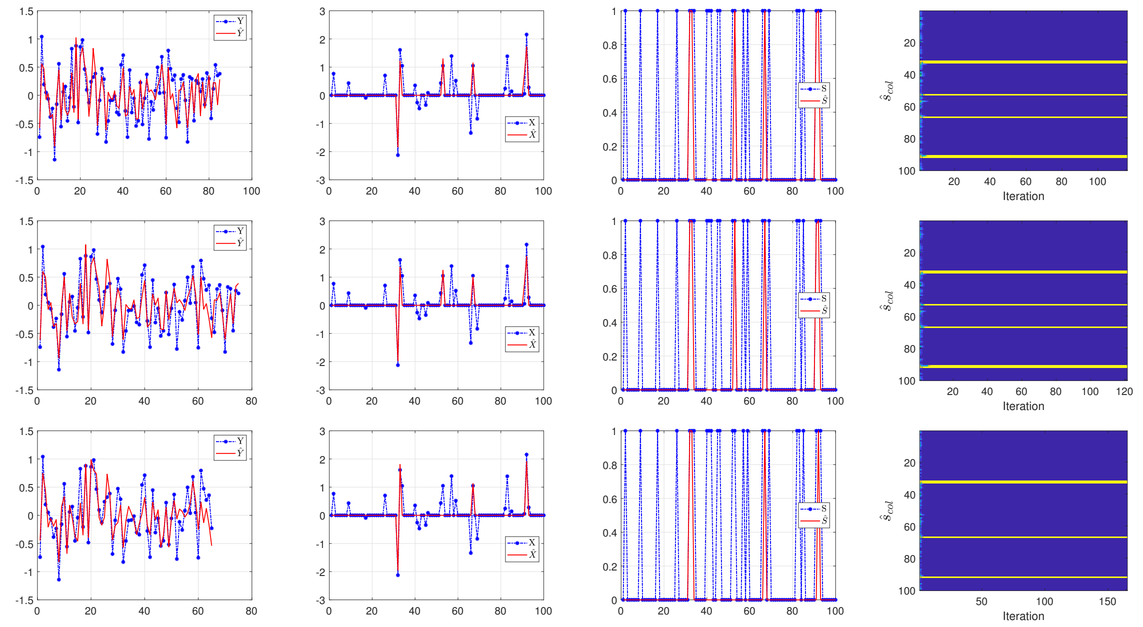


Fig. 5.2: Example 2: $(\alpha_0, \beta_0) = (0.1, 0.9)$. From top to bottom, the rows show the results of SBL-VB(BGiG) for the sampling ratio 0.85, 0.75, 0.65, respectively. The term $hats_{coll}$ is the estimated support vector in each iteration of the algorithm.

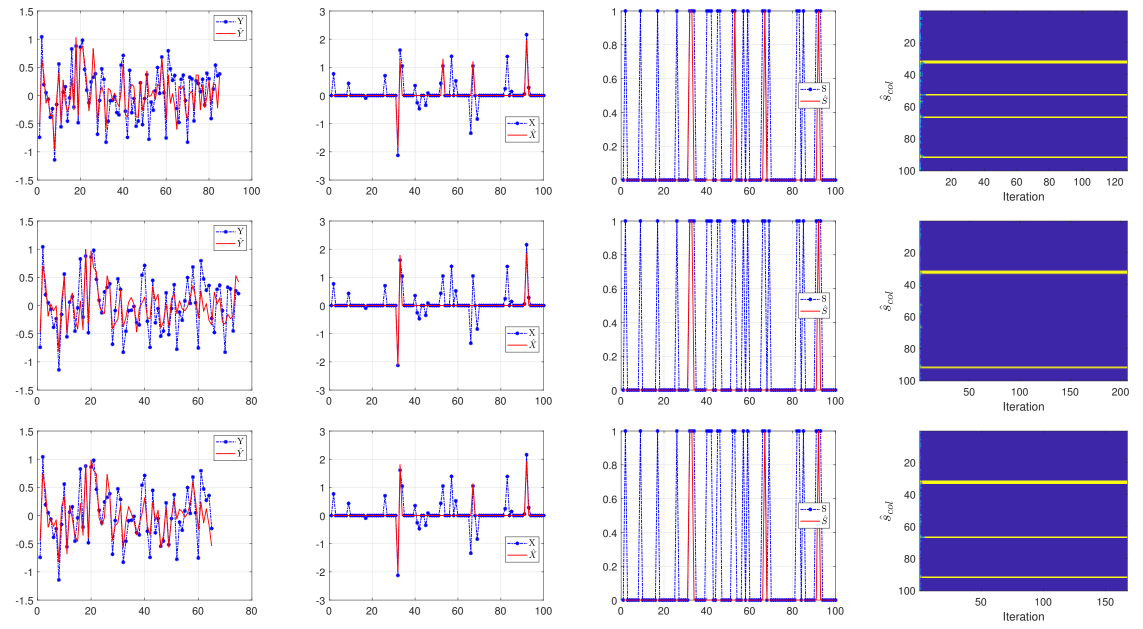


Fig. 5.3: Example 3: $(\alpha_0, \beta_0) = (0.01, 0.99)$. From top to bottom, the rows show the results of SBL-VB(BGiG) for the sampling ratio 0.85, 0.75, 0.65, respectively. The term $hats_{coll}$ is the estimated support vector in each iteration of the algorithm.

Table 5.1: Performance results of SBL-VB(BGiG) for Example 1.

$\lambda = M/N$	P_D	P_{FA}	$P_D - P_{FA}$	NMSE (dB)
Example 1, λ=0.85	0.7200	0	0.7200	-17.0833
Example 1, λ=0.75	0.6800	0.0133	0.6667	-11.7855
Example 1, λ=0.65	1	1	0	-4.6194

the performance of example 1. This also shows that for sampling ratio of $\lambda = 0.65$, the algorithm fails to provide reasonable results.

As stated earlier, in Example 2, (α_0, β_0) is set to $(0.1, 0.9)$. When $(\alpha_0, \beta_0) = (0.1, 0.9)$ we are essentially modeling the support of the solution to have sparsity level of $N \times \alpha_0/(\alpha_0 + \beta_0) = 10$ on the average, according to the prior defined in (5.4). In constrast, the average sparsity level as a prior was set to $N \times \alpha_0/(\alpha_0 + \beta_0) \approx 41$ in Example 1, which is not very sparse. Also, according to the update rule (5.10), the pair (α_0, β_0) has lower weight in the update of $\alpha_{1,n}$ and $\beta_{1,n}$ than $\tilde{s}_n$ when comparing Example 2 with Example 1. This means that the prior in Example 2 has lower impact on the estimation than the contribution of the measurements (observations), which is reasonable when there is no information about the true signal other than the measurements is available to us.

Fig. 5.2 and Table 5.2 present results related to Example 2. Unfortunately, Fig. 5.2 shows that the setting for (α_0, β_0) in Example 2 fails to provide very good results even for high sampling ratios. Similarly, based on Fig. 5.3 and Table 5.3, the settings for (α_0, β_0) in Example 3 do not provide very good results even for high sampling ratios. More specifically, it turns out that Example 2 and Example 3 tend to provide very sparse solutions for the sampling ratios within $M/N = [0, 1]$. These initial experiments suggest that there is no fixed setting for (α_0, β_0) performing reasonably well for all sampling ratios and the selection of the hyperparameters (α_0, β_0) should be made with care.

Continuing this examination, in Figs. 5.4-5.6, we illustrate the negative log-likelihood, the precision on the noise, and the precision on the generated true solution in Examples 1-3, respectively. The horizontal axis shows the iterations until the stopping rule has met.

As expected as the sampling ratio increases, the algorithm requires fewer iterations to

Table 5.2: Performance results of SBL-VB(BGiG) for Example 2.

$\lambda = M/N$	P_D	P_{FA}	$P_D - P_{FA}$	NMSE (dB)
Example 2, λ=0.85	0.2400	0	0.2400	-3.4818
Example 2, λ=0.75	0.2400	0	0.2400	-3.4116
Example 2, λ=0.65	0.1600	0	0.1600	-3.3134

Table 5.3: performance results of SBL-VB(BGiG) for Example 3.

$\lambda = M/N$	P_D	P_{FA}	$P_D - P_{FA}$	NMSE (dB)
Example 3, λ=0.85	0.2000	0	0.2000	-3.6776
Example 3, λ=0.75	0.1200	0	0.1200	-2.9097
Example 3, λ=0.65	0.1600	0	0.1600	-3.3135

meet its stopping condition because it is provided with more measurements. This can be seen on the negative log-marginalized likelihood plots in Figs. 5.4–5.6. According to Fig. 5.4, for the sampling ratio of 0.85 and 0.75, the hyperparameter settings in Example 1 seem satisfactory in terms of log-likelihood function and the precisions on both the noise and the solution components. As a result, we observe good performance for Example 1 in Fig. 5.1 and Table 5.1. However, for lower sampling ratio, Example 1 could not learn the precision on the measurement noise and the precision on the solution, resulting in poor performance. This effect is illustrated in the 3rd row of Fig. 5.1, Table 5.1, and also in Fig. 5.4. Similarly, poor performance occurred in Examples 2 and 3 as illustrated in Fig. 5.4 and Fig. 5.5, respectively. The main issue of the failures can be found in the update rule of the support learning vector $\tilde{\boldsymbol{s}}$ defined in (5.7). It is important to balance between the terms c_n and κ_n, where c_n imposes the effect of hyperprior on $\boldsymbol{s}$ accompanied by the current estimate of s_n, and κ_n imposes the contribution of the current estimates of noise precision, solution, and other supports in fitting the model to the measurements. Therefore, if we impose a strong weight on the sparsity via c_n, then the solution tends to neglect the effect of κ_n and vice versa. That is why we had very sparse (with poor performance) in Examples 2 and 3 for all the represented sampling ratios, and nonsparse (with poor performance) for lower

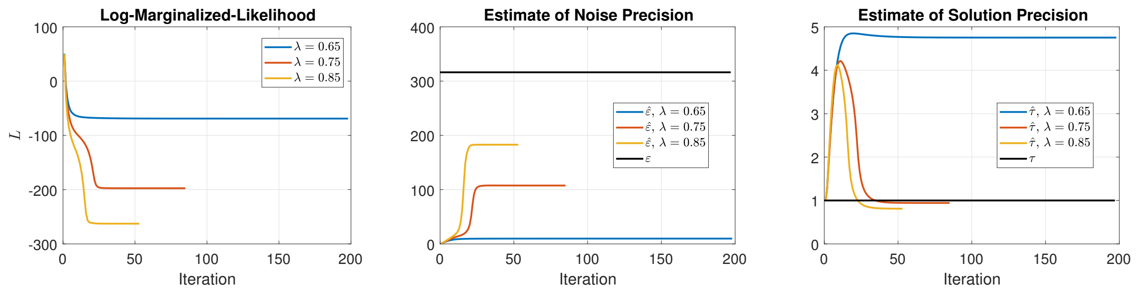


Fig. 5.4: Example 1: Performance evaluation of SBL-VB(BGiG).

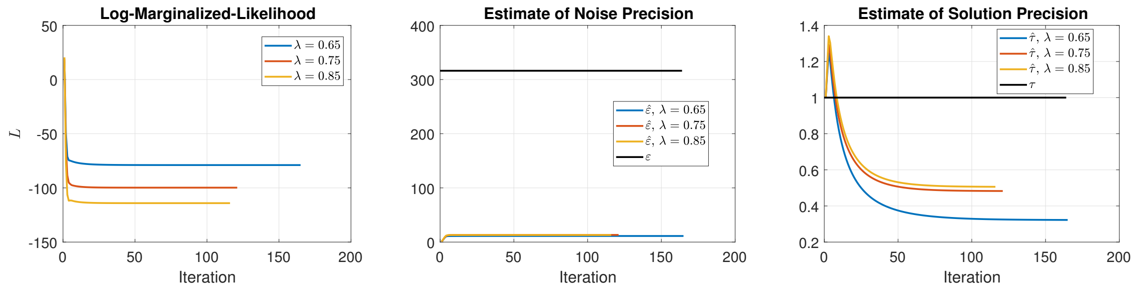


Fig. 5.5: Example 2: Performance evaluation of SBL-VB(BGiG).

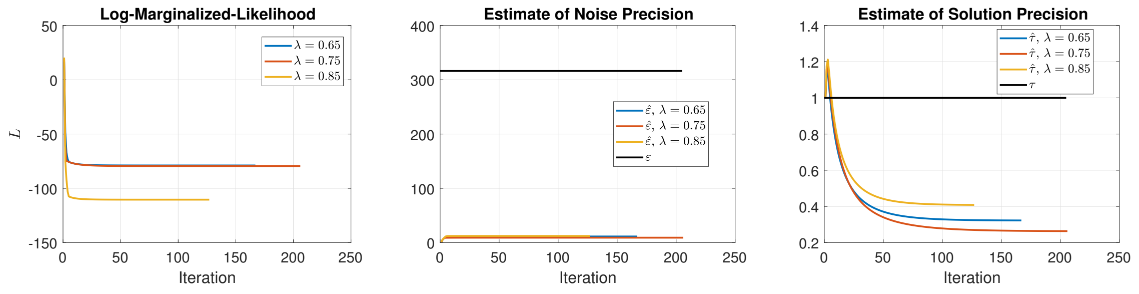


Fig. 5.6: Example 3: Performance evaluation of SBL-VB(BGiG).

sampling ratio in Example 1. These results suggest that the algorithm and its update rules are sensitive to the selection of hyperparameters on the Gamma prior on the support vector $\boldsymbol{s}$. In Section 5.5, we will discuss how one may be able to deal with this issue and obtain better performance for the SBL-VB(BGiG) algorithm.

5.4 Gaussians-Inverse Gammas Modeling and SBL-VB(GiG) Algorithm

In this section, we consider the Gaussians-inverse Gammas model. In this model, the components of the solution are modeled by zero-mean Gaussians with different precisions. The main difference between this model and the model defined in Section 5.3 is that here we do not have the support vector $\boldsymbol{s}$ and instead different precisions are considered on the components of the solution vector $\boldsymbol{x}_s$ in (5.1). The set of priors in this model are defined as follows.

$$x_n \sim \mathcal{N}(0, \tau_n^{-1}),\ \tau_n \sim \text{Gamma}(a_0, b_0),\ \forall n, \tag{5.14}$$

where a_0 and b_0 denote the shape and rate of the Gamma distribution, respectively. The entries of the noise component $\boldsymbol{e}$ are defined the same as (5.6), i.e., $\boldsymbol{e} \sim \mathcal{N}(0, \varepsilon^{-1} I_M)$, $\varepsilon \sim \text{Gamma}(\theta_0, \theta_1)$, where θ_0 and θ_1 are set to small positive values.

The estimation of the parameters in this model is carried out using VB inference as discussed below.

5.4.1 Update Rules of SBL-VB(GiG) Using VB Inference

According to the VB algorithm described in (5.2) and (5.3), the update rule of the variables and parameters of the GiG model can be simplified as follows.

- Update rule for the precision τ_n on x_n using VB

$$q(\tau_n) \sim \text{Gamma}\big(a_0 + \frac{1}{2}, b_0 + \frac{1}{2}\|x_n^2\|_{q_{xn}}\big), \forall n = 1, \ldots, N.$$

Thus

$$\tilde{\tau_n} = \frac{a_0 + \frac{1}{2}}{b_0 + \frac{1}{2}(\tilde{x}_n^2 + \sigma_{\tilde{x}_n}^2)},\ \forall n = 1, 2, \ldots, N. \tag{5.15}$$

- Update rule for the noise precision ε using VB

$$q(\varepsilon) \sim \text{Gamma}\big(\theta_0 + \frac{M}{2}, b_0 + \frac{1}{2}\tilde{\Psi}\big)$$

which yields

$$\tilde{\varepsilon} = \frac{\theta_0 + \frac{M}{2}}{\theta_1 + \frac{1}{2}\tilde{\Psi}}, \tag{5.16}$$

where $\tilde{\Psi} := \boldsymbol{y}^T\boldsymbol{y} - 2\tilde{\boldsymbol{x}}^T A^T \boldsymbol{y} + \text{Tr}\left((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}})A^T A\right)$.

- Update rule for the solution vector $\boldsymbol{x}$ using VB

$$q_x(\boldsymbol{x}) \sim \mathbf{N}(\tilde{\boldsymbol{x}}, \Sigma_{\tilde{x}}), \tag{5.17}$$

where $\Sigma_{\tilde{x}} := (\tilde{T} + \tilde{\varepsilon} A^T A)^{-1}$, $\tilde{\boldsymbol{x}} := \tilde{\varepsilon}\Sigma_{\tilde{x}} A^T \boldsymbol{y}$, and $\tilde{T} := \text{diag}\{[\tilde{\tau}_1, \ldots, \tilde{\tau}_N]\}$.

We set the stopping rule of the algorithm using the marginalized likelihood (evidence) defined as

$$p(\boldsymbol{y}|\varepsilon,\tau) = \int p(\boldsymbol{y}|\boldsymbol{x},\varepsilon,\tau)p(\boldsymbol{x}|\tau)d\boldsymbol{x}.$$

After simplification and for the comparison purposes of $L^{[t+1]}$ with $L^{[t]}$ in the updating process, we have

$$L \propto \log|\Sigma_0| - \boldsymbol{y}^T\Sigma_0\boldsymbol{y},$$

where is defined as $\Sigma_0 := (\varepsilon^{-1}I_M + T^{-1}AA^T)^{-1}$.

Below, we provide the pseudo code for SBL-VB(GiG) algorithm for the Gaussians-inverse Gammas modeling.

SBL-VB(GiG) Algorithm:

```
$\hat{\boldsymbol{x}}$ = **SBL-VB-GiG**$(Y, A)$
Iter= 1
**While** $\frac{|L^{[\text{Iter}]} - L^{[\text{Iter}-1]}|}{|L^{[\text{Iter}-1]}|} \geq 10^{-6}$
        Compute $\Sigma_{\tilde{x}}$ and $\tilde{\boldsymbol{x}}$ from (5.17)      % (Solution-value matrix component)
        Compute $T$ from (5.15)      % (Precisions on the solution)
        Compute $\varepsilon$ from (5.16)      % (Precision on the noise)
        Compute $L^{[\text{Iter}]}$ and then Iter=Iter+1
**End While**
```

5.4.2 Issues with SBL-VB(GiG) and a Study Example

An issue with the SBL-VB(GiG) algorithm is that the solution becomes nonsparse due to the fact that it does not incorporate a binary vector $\boldsymbol{s}$ (hard-thresholding or soft-thresholding if the expected value is used) as we had in SBL-VB(BGiG). This may have no major effect on the signal reconstruction for high sampling ratios. However, the nonsparseness effect appears in low sampling ratios by misleading the algorithm to wrongly activate some components in the estimated signal, yet providing a good fit of the model to the measurements. In Figs. 5.7 and 5.8, we illustrate the results of Example 3 by applying the SBL-VB(GiG) algorithm. More specifically, in the top row of Fig. 5.7 we show the comparison of $\boldsymbol{y}$ with $\hat{\boldsymbol{y}} = A\hat{\boldsymbol{x}}$ for sampling ratios of 0.85, 0.75, 0.65, and 0.50 from left to right. In the second row of this figure, the results for the true and the estimated solution are demonstrated. The third row illustrates the estimated solution components. In Fig. 5.8, we demonstrate the negative log-marginalized likelihood comparison and the also the estimated noise precision against the true noise precision, as well.

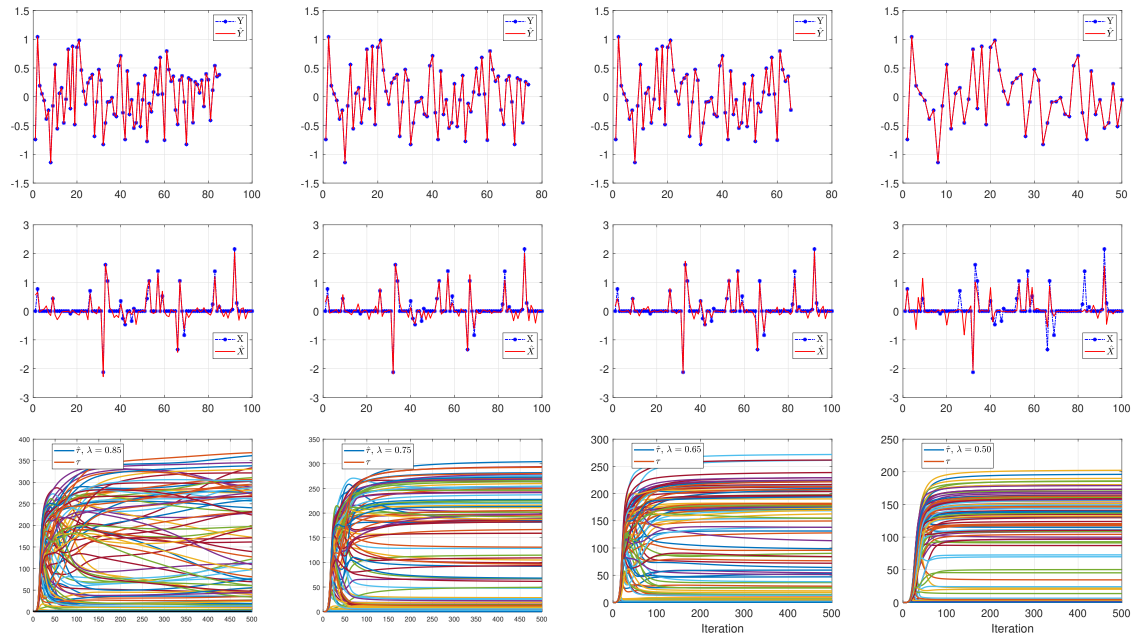


Fig. 5.7: Example 3: From left to right, we show the results of SBL-VB(GiG) for the sampling ratio 0.85, 0.75, 0.65, and 0.50, respectively. The first row demonstrates the estimated measurements compared with the true ones. The second row shows the solution estimates against the true solutions.

From the results shown in Fig. 5.7 and Fig. 5.8, we observe that the solution does not tend to be sparse. This effect is illustrated in the second row of Fig. 5.7. Also, as the sampling ratio decreases, the solution estimate has poor performance, due not only to the reduction in the number of measurements but also to the nonsparseness behavior. In other

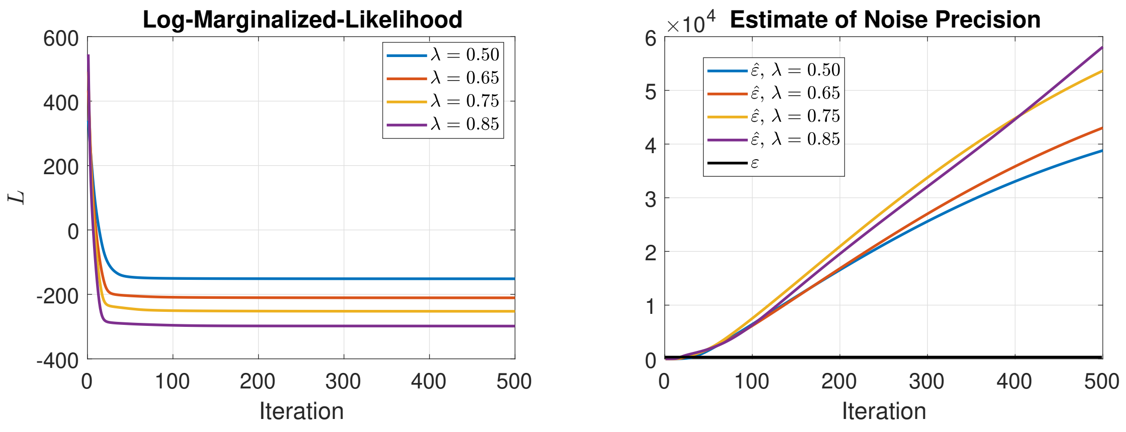


Fig. 5.8: Example 3: From left to right, we show the behavior of negative marginalized log-likelihood and the precision on the noise using SBL-VB(GiG) for the sampling ratios of 0.85, 0.75, 0.65, and 0.50.

words, the sparse solution would have occurred if most of precisions on the solution components had become fairly large. Can we do some extra work on the estimations produced via SBL-VB(GiG) algorithm to make the solution sparse in order to improve the reconstruction performance? This question will be addressed in the next section.

5.5 PreProcessing versus Postprocessing

In this section, we show that in order to improve the performance of Bernoullis-Gaussian-inverse Gamma modeling using the OSBL-VB(BGiG) algorithm, we need to perform some sort of preprocessing. Also, the results in Section 5.4 suggest one can perform some postprocessing for the OSBL-VB(GiG) algorithm to improve the reconstruction performance. Below, we provide more details for each of these algorithms.

5.5.1 Preprocessing for the OSBL-VB(BGiG) Algorithm

Based on the observations made on the performance of SBL-VB(BGiG) in Section 5.3.2, we showed that the pair of hyperparameters (α_0,β_0) should be selected with care. In other words, getting a good performance with this algorithm needs some preprocessing to assess reasonable setting for the parameters. According to the observations made in Examples 1-3, we perform a grid search on the hyperparameters (α_0, β_0) to see whether we can find some

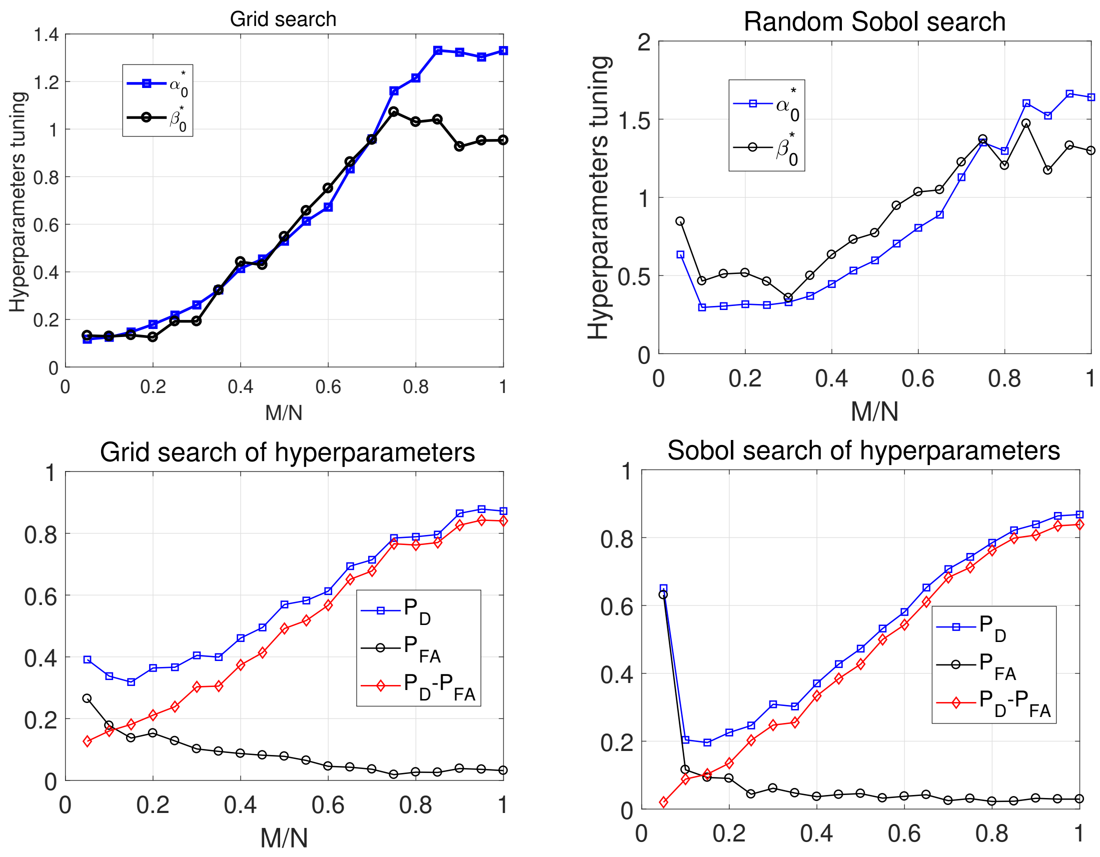


Fig. 5.9: Performance evaluation of SBL-VB(BGiG) using grid and random Sobol search.

common pattern on selecting these parameters for all sampling ratios. The grid search runs the algorithm for different values of α_0 and β_0 with the search range of $[0.1 : 0.1 : 2]$. For each selection of (α_0, β_0) within this range, we ran 200 random trials and then averaged the results. The results were examined to see what values of (α_0, β_0) provided the highest performance in terms of the difference between the detection rate and false alarm rate, $P_D - P_{FA}$. The simulation were executed for a range of sampling ratios in the range $[0.05, 1]$ with the step size of 0.05. The results are demonstrated in Fig. 5.9. In this figure, we also provide the results of performing random Sobol search for (α_0, β_0). Sobol search is more or less a deterministic scheme for getting point sets that are more representative of random uniform draws than actual random uniform draws [127, 128]. The two left plots in Fig. 5.9 show the results for the best setting of (α_0, β_0).

It is clear in Fig. 5.9 that there is no fixed setting for these parameters in order for getting the best performance. The two plots on the right of Fig. 5.9 illustrate the performance based on the best values of these hyperparameters, which provided the best performance, i.e., tuned hyperparameters. We also examined the results of the grid search for the top 10 highest performance for each sampling ratio, where performance is in terms of $P_D - P_{FA}$ and the normalized mean-squared error (NMSE). In Fig. 5.10(a) we demonstrate the top 10 highest performance based on NMSE and $P_D - P_{FA}$ for different sampling ratios. In Fig. 5.10(b) and Fig. 5.10(c), we illustrate the values of (α_0, β_0) which led to the performances shown in Fig. 5.10(a) for different sampling ratios. Fig. 5.11 details the top 10 values of (α_0, β_0) *vs.* sampling ratio.

In Fig. 5.10 (b,c) it can be observed that there is no specific pattern for these hyperparameters. Fig. 5.11 also shows that hyperparameters need to be carefully selected.

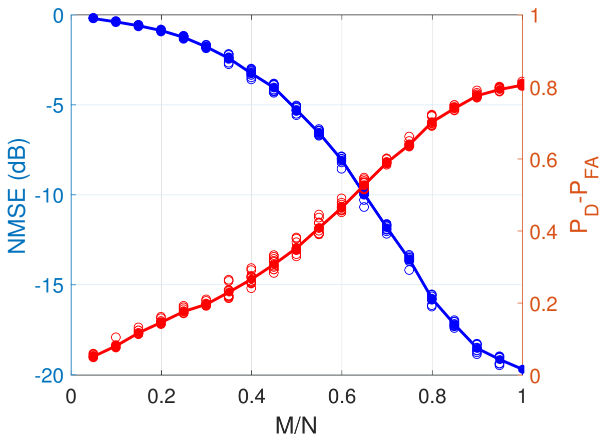
5.5.2 Postprocessing for the OSBL-VB(GiG) Algorithm

Since the OSBL-VB(GiG) algorithm does not include the binary support vector $\boldsymbol{s}$, as SBL-VB(BGiG) possesses, the resulting solution becomes nonsparse. This leads to high detection rate for the location of active supports and high false alarm rate. Thus, as the sampling ratio decreases, there is a high chance that this algorithm misunderstands the locations of the true solution. Therefore, SBL-VB(GiG) requires some postprocessing to discard the components with low amplitudes. This effect can be seen in Fig. 5.12(a). The curves with solid lines in this plot show the detection and false alarm rate in support recovery, and the difference between the rates. This issue can be resolved by performing some postprocessing to find a data-driven threshold. The amplitudes in the reconstructed signal with lower values than the threshold are discarded. For this purpose, we set up 200 random trials, the same as the one explained for SBL-VB(BGiG), and then evaluate the performance in terms of NMSE by varying the threshold. Fig. 5.12(b) shows the averaged results of 200 trials.

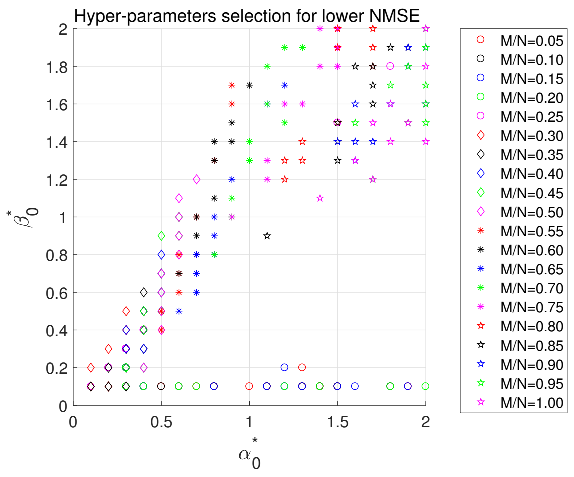
In Fig. 5.12(b) we observe that for low and moderate sampling ratios the postprocessing does not benefit us so much in terms of reconstruction error. However, there is a threshold of around 0.25 for which the postprocessing step gained us the reduction in reconstruction error of approximately 3 dB. We set the threshold to 0.25 and run 200 random trials by applying SBL-VB(GiG) and evaluating the performance based on the detection and false alarm rate in support recovery. The results are illustrated by the dashed lines in Fig. 5.12(a). According to this figure, it should be clear that the additional postprocessing step provides reasonable results. Finally, in Fig. 5.12(c) we compare the performance of the SBL-VB(BGiG) algorithm (with performing the preprocessing step) with the SBL-VB(GiG) algorithm (after performing postprocessing). We see that for low and high sampling ratios, Bernoullis-Gaussians-inverse Gamma implemented via SBL-VB(BGiG) provides better performance. In contrast, for the moderate sampling ratios Gausssians-inverse Gammas modeling implemented via SBL-VB(GiG) performs much better.

5.6 Summary

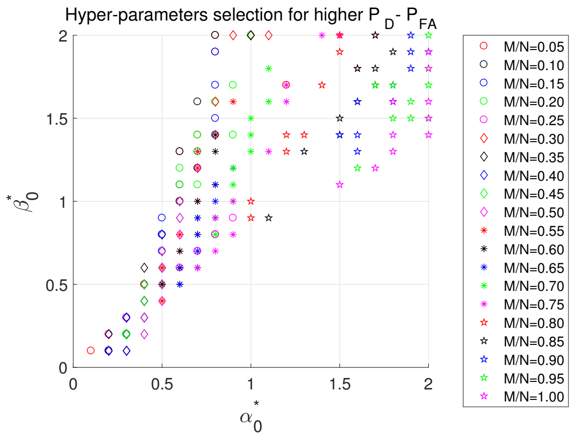
We investigated solving the inverse problem of compressive sensing using VB inference for two sparse Bayesian models of Bernoullis-Gaussians-inverse Gamma and Gaussians-inverse Gammas. The issues of each approach was discussed and the performance between the two models were compared. The simulation results demonstrated that the proposed algorithm competes with the other state-of-the-art algorithms.



(a)

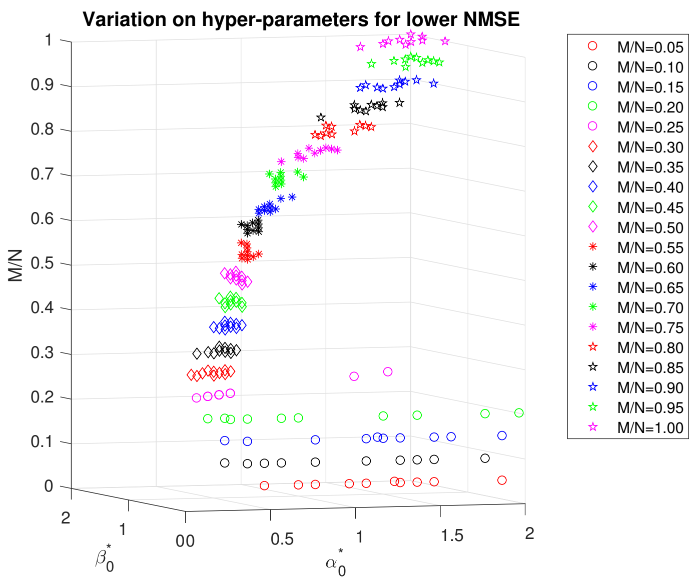


(b)

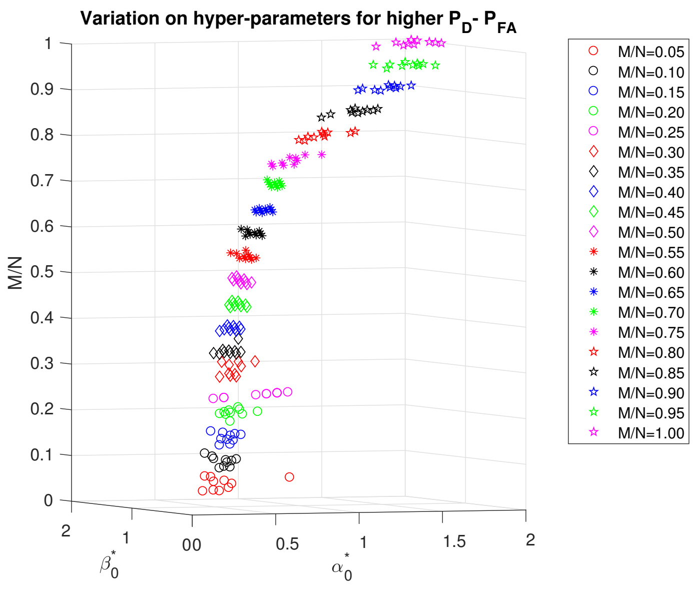


(c)

Fig. 5.10: (a). Performance, (b) Top 10 (α_0, β_0) with lowest NMSE, (c) Top 10 (α_0, β_0) with highest $P_D - P_{FA}$.

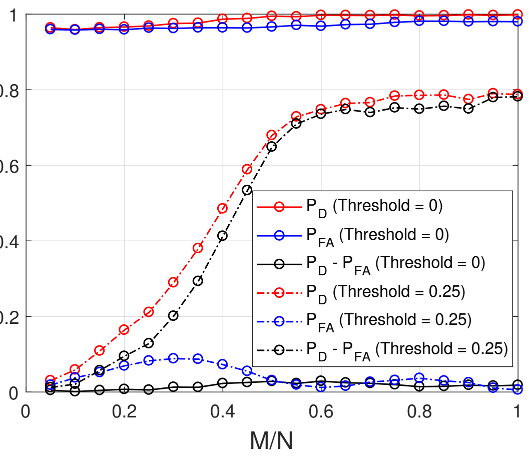


(a)

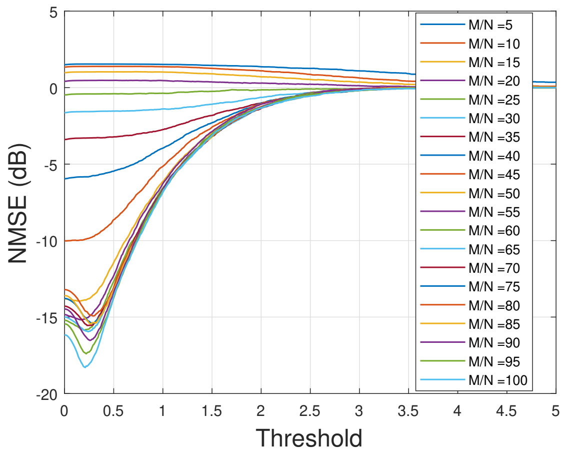


(b)

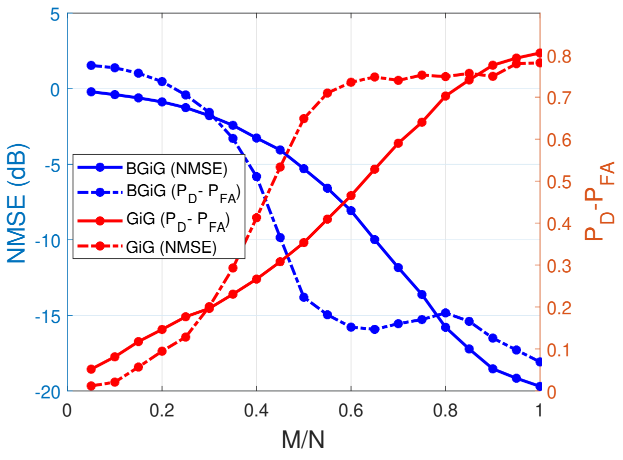
Fig. 5.11: (a). Top 10 (α_0, β_0) with lowest NMSE *vs.* sampling ratio. (b) Top 10 (α_0, β_0) with lowest NMSE *vs.* sampling ratio.



(a) Performance of SBL-VB(GiG) before and after postprocessing.



(b) NMSE of SBL-VB(GiG) *vs.* threshold to remove some components of $\hat{\boldsymbol{x}}$.



(c) Comparison of the two algorithms.

Fig. 5.12: Performance of SBL-VB(BGiG) and SBL-VB(GiG)

5.7 Appendix

In this section, we provide details on the update rules of the parameters and variables for both the SBL-VB(BGiG) and SBL-VB(GiG) algorithms.

5.7.1 Bernoullis-Gaussians-Inverse Gamma Modeling and the SBL-VB(BGiG) Algorithm

- Update rule for the precision τ of the solution value vector $\boldsymbol{x}$ using VB

$$\begin{aligned} q(\tau) &\propto p(\tau; a_0, b_0) \exp\big\{\langle \log\{p(\boldsymbol{x}|\tau I_N)\}\rangle_{q_x}\big\} \\ &\propto \tau^{a_0-1}e^{-b_0\tau} \exp\Big\{\langle \log\{\prod_{n=1}^{N} p(x_n;\tau^{-1})\}\rangle_{q_x}\Big\} \\ &\propto \tau^{a_0-1}e^{-b_0\tau} \exp\big\{\langle \log\{\tau^{\frac{N}{2}} \exp\{-\frac{\tau}{2}\|\boldsymbol{x}\|_2^2\}\}\rangle_{q_x}\big\} \\ &\propto \tau^{a_0+\frac{N}{2}-1} e^{-(b_0+\frac{1}{2}\langle\|\boldsymbol{x}\|_2^2\rangle_{q_x})\tau} \end{aligned}$$

Therefore,

$$q(\tau) \sim \text{Gamma}\big(a_0 + \frac{N}{2}, b_0 + \frac{1}{2}\langle\|\boldsymbol{x}\|_2^2\rangle_{q_x}\big).$$

The update rule for τ can be then defined as follows

$$\tilde{\tau} = \frac{a_0 + \frac{N}{2}}{b_0 + \frac{1}{2}\langle\|\boldsymbol{x}\|_2^2\rangle_{q_x}},$$

where

$$\langle\|\boldsymbol{x}\|_2^2\rangle_{q_x} = \langle\boldsymbol{x}^T\boldsymbol{x}\rangle_{q_x} = \text{tr}(\langle\boldsymbol{x}\boldsymbol{x}^T\rangle_{q_x}) = \text{tr}(\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}}),$$

where $\tilde{\boldsymbol{x}} := \langle\boldsymbol{x}\rangle_{q_x}$ and $\Sigma_{\tilde{x}} = \text{diag}\{\sigma^2_{\tilde{x}_1}, \ldots, \sigma^2_{\tilde{x}_N}\}$. Finally, the update rule for τ can be written as

$$\tilde{\tau} = \frac{a_0 + \frac{N}{2}}{b_0 + \frac{1}{2}\big(\text{tr}(\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}})\big)} = \frac{a_0 + \frac{N}{2}}{b_0 + \frac{1}{2}\big(\|\tilde{\boldsymbol{x}}\|_2^2 + \sum_{n=1}^{N}\sigma^2_{\tilde{x}_n}\big)}.$$

- Update rule for the noise precision ε using VB

$$\begin{aligned}q(\varepsilon) &\propto (\varepsilon;\theta_0,\theta_1)\exp\big\{\langle \log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\}\rangle_{q_xq_s}\big\}\\ &\propto \varepsilon^{\theta_0-1}e^{-\theta_1\varepsilon}\exp\Big\{\langle \log\{\varepsilon^{\frac{M}{2}}\exp\{-\frac{1}{2}\varepsilon\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\}\}\rangle_{q_xq_s}\Big\}\\ &\propto \varepsilon^{\theta_0+\frac{M}{2}-1}e^{-(\theta_1+\frac{1}{2}\big\langle\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\big\rangle_{q_xq_s})\varepsilon}.\end{aligned}$$

Therefore,

$$q(\varepsilon)\sim\text{Gamma}\big(\theta_0+\frac{M}{2},\theta_1+\frac{1}{2}\langle\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\rangle_{q_xq_s}\big).$$

The update rule for ε can be then defined as follows

$$\tilde{\varepsilon}=\frac{\theta_0+\frac{M}{2}}{\theta_1+\frac{1}{2}\langle\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\rangle_{q_xq_s}},$$

where $\langle\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\rangle_{q_xq_s}$ is obtained as follows. Define $S=\text{diag}\{\boldsymbol{s}\}$.

$$\begin{aligned}\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\rangle_{q_xq_s} &= \|\boldsymbol{y}-AS\boldsymbol{x})\|_2^2\rangle_{q_xq_s}\\ &=\boldsymbol{y}^T\boldsymbol{y}-2\langle\boldsymbol{x}^TSA^T\boldsymbol{y}\rangle_{q_xq_s}+<\boldsymbol{x}^TSA^TAS\boldsymbol{x}\rangle_{q_xq_s}\\ &=\boldsymbol{y}^T\boldsymbol{y}-2\langle\boldsymbol{x}\rangle_{q_x}^T\langle S\rangle_{q_s}A^T\boldsymbol{y}+\langle\boldsymbol{x}^TSA^TAS\boldsymbol{x}\rangle_{q_xq_s}\\ &=\boldsymbol{y}^T\boldsymbol{y}-2(\tilde{\boldsymbol{x}}\circ\tilde{\boldsymbol{s}})^TA^T\boldsymbol{y}+\langle\boldsymbol{x}^TM_s\boldsymbol{x}\rangle_{q_xq_s},\end{aligned}$$

where $M_s:=SA^TAS$ and

$$\langle\boldsymbol{x}^TM_s\boldsymbol{x}\rangle_{q_xq_s}=\text{tr}\left(\langle\boldsymbol{x}\boldsymbol{x}^T\rangle_{q_x}\langle M_s\rangle_{q_s}\right)=\text{tr}\left((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T+\Sigma_{\tilde{x}})\langle M_s\rangle_{q_s}\right)$$

$$\begin{aligned}\langle M_s\rangle_s &= \langle SA^TAS\rangle_{q_s}=\langle(A^TA)\circ(\boldsymbol{s}\boldsymbol{s}^T)\rangle_{q_s}=(A^TA)\circ\langle(\boldsymbol{s}\boldsymbol{s}^T)\rangle_{q_s}\\ &=(A^TA)\circ\Big(\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T+\text{diag}\left(\tilde{\boldsymbol{s}}\circ(1-\tilde{\boldsymbol{s}})\right)\Big).\end{aligned}$$

Therefore,

$$\langle \boldsymbol{x}^T M_s \boldsymbol{x} \rangle_{q_x q_s} = \mathrm{tr}\left((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}}) \big((A^T A) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \mathrm{diag}\{\tilde{\boldsymbol{s}} \circ (1 - \tilde{\boldsymbol{s}})\}) \big) \right).$$

Finally, the update rule for the precision of the noise can be written as

$$\tilde{\varepsilon} = \frac{\theta_0 + \frac{M}{2}}{\theta_1 + \frac{1}{2}\tilde{\Psi}},$$

where

$$\begin{aligned} \tilde{\Psi} :=& \langle \| \boldsymbol{y} - A(\boldsymbol{s} \circ \boldsymbol{x}) \|_2^2) \rangle_{q_x q_s} = \\ & \boldsymbol{y}^T \boldsymbol{y} - 2(\tilde{\boldsymbol{x}} \circ \tilde{\boldsymbol{s}})^T A^T \boldsymbol{y} + \mathrm{tr}\left((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}}) \big((A^T A) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \mathrm{diag}\{\tilde{\boldsymbol{s}} \circ (1 - \tilde{\boldsymbol{s}})\}) \big) \right). \end{aligned}$$

Remark 5.1: Notice that $\mathrm{tr}\,(X^T Y) = \sum_{i,j} (X \circ Y)_{ij} = \mathbf{1}^T (X \circ Y) \mathbf{1}$. Therefore,

$$\begin{aligned} & \mathrm{tr}\left((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}}) ((A^T A) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \mathrm{diag}\{\tilde{\boldsymbol{s}} \circ (1 - \tilde{\boldsymbol{s}})\})) \right) \\ &= \mathbf{1}^T \big((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}}) \circ (A^T A) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \mathrm{diag}\{\tilde{\boldsymbol{s}} \circ (1 - \tilde{\boldsymbol{s}})\})) \big) \mathbf{1}, \end{aligned}$$

where $\mathbf{1} = [1, \dots, 1]^T$.

Thus, $\tilde{\Psi}$ can be written as

$$\tilde{\Psi} := \boldsymbol{y}^T \boldsymbol{y} - 2(\tilde{\boldsymbol{x}} \circ \tilde{\boldsymbol{s}})^T A^T \boldsymbol{y} + \mathbf{1}^T \big((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}}) \circ (A^T A) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \mathrm{diag}\{\tilde{\boldsymbol{s}} \circ (1 - \tilde{\boldsymbol{s}})\})) \big) \mathbf{1}.$$

- Update rule for γ_n using VB

$$\begin{aligned}
q(\gamma_n) &\propto p(\gamma_n;\alpha_0,\beta_0)\exp\left(\langle\log\{p(\boldsymbol{x},\boldsymbol{s},\boldsymbol{y}|\theta,\mathbb{M})\}\rangle_{q_x q_s}\right)\\
&\propto \gamma_n^{\alpha_0-1}(1-\gamma_n)^{\beta_0-1}\exp\{\langle\log\{p(s_n|\gamma_n)\}\rangle_{q_x q_s}\}\\
&\propto \gamma_n^{\alpha_0-1}(1-\gamma_n)^{\beta_0-1}\exp\{\langle\log\{\gamma_n^{s_n}(1-\gamma_n)^{1-s_n}\}\rangle_{q_{s_n}}\}\\
&\propto \gamma_n^{\alpha_0-1}(1-\gamma_n)^{\beta_0-1}e^{\langle s_n\rangle_{q_{s_n}}\log\gamma_n}e^{(1-\langle s_n\rangle_{q_{s_n}})\log\{1-\gamma_n\}}\\
&\propto \gamma_n^{\alpha_0-1}(1-\gamma_n)^{\beta_0-1}\gamma_n^{\langle s_n\rangle_{q_{s_n}}}(1-\gamma_n)^{1-\langle s_n\rangle_{q_{s_n}}}\\
&\propto \gamma_n^{\alpha_0+\tilde{s}_n-1}(1-\gamma_n)^{\beta_0-\tilde{s}_n}.
\end{aligned}$$

Therefore,

$$q_{\gamma_n}(\gamma_n)\sim \text{Beta}(\alpha_{1,n},\beta_{1,n}),\ \forall n=1,\dots,N,$$

where $\alpha_{1,n}:=\alpha_0+\tilde{s}_n$ and $\beta_{1,n}:=\beta_0+1-\tilde{s}_n$. The update rule for γ_n can be then defined as follows

$$\tilde{\gamma}_n=\frac{\alpha_{1,n}}{\alpha_{1,n}+\beta_{1,n}}.$$

- Update rule for the solution vector $\boldsymbol{x}$ using VB

$$\begin{aligned}
q_x(\boldsymbol{x}) &\propto \exp\{\langle\log\{p(\boldsymbol{x},\boldsymbol{s},\boldsymbol{y}|\theta,\mathbb{M})\}\rangle_{q_x q_s}\}\\
&\propto \exp\{\langle\log\{p(\boldsymbol{x},\boldsymbol{s}|\theta,\mathbb{M})p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\theta,\mathbb{M})\}\rangle_{q_\theta q_s}\}\\
&\propto \exp\{\langle\log\{p(\boldsymbol{x}|\theta)\}\rangle_{q_\theta}\}\exp\{\langle\log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\theta,\mathbb{M})\}\rangle_{q_\theta q_s}\}\\
&\propto \exp\{\langle\log\{p(\boldsymbol{x}|\tau)\}\rangle_{q_\tau}\}\exp\{\langle\log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\}\rangle_{q_\varepsilon q_s}\}
\end{aligned}$$

For the purpose of updating the elements of $\boldsymbol{x}$, we have

$$\begin{aligned}
p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{s},\varepsilon) &\propto \varepsilon^{\frac{M}{2}}\exp\{-\frac{1}{2}\varepsilon\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\}\\
&\propto \exp\{-\frac{1}{2}\varepsilon\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\langle\log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\}\rangle_{q_\varepsilon q_s} &\propto -\frac{1}{2}\langle\varepsilon\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\rangle_{q_\varepsilon q_s}\\
&\propto -\frac{1}{2}\langle\varepsilon\rangle_{q_\varepsilon}\langle\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\rangle_{q_s}\\
&\propto -\frac{1}{2}\tilde{\varepsilon}\langle|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\rangle_{q_s}
\end{aligned}$$

Also,

$$\begin{aligned}\langle\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\rangle_{q_s} &= <\operatorname{tr}\left(\boldsymbol{y}\boldsymbol{y}^T+(\boldsymbol{x}\circ\boldsymbol{s})^TA^TA(\boldsymbol{x}\circ\boldsymbol{s})-2(\boldsymbol{x}\circ\boldsymbol{s})^TA^T\boldsymbol{y}\right)\rangle_{q_s}\\ &\propto \langle\operatorname{tr}\left((\boldsymbol{x}\circ\boldsymbol{s})^TA^TA(\boldsymbol{x}\circ\boldsymbol{s})-2(\boldsymbol{x}\circ\boldsymbol{s})^TA^T\boldsymbol{y}\right)\rangle_{q_s}\\ &\propto \langle\operatorname{tr}\left(\boldsymbol{x}^TS^TA^TAS\boldsymbol{x}-2\boldsymbol{x}^TSA^T\boldsymbol{y}\right)\rangle_{q_s}\\ &\propto \operatorname{tr}\left(\boldsymbol{x}^T\langle SA^TAS\rangle_{q_s}\boldsymbol{x}-2\boldsymbol{x}^T\tilde{S}A^T\boldsymbol{y}\right),\end{aligned}$$

where $S := \operatorname{diag}\{\boldsymbol{s}\}$. This yields to

$$\langle\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\rangle_{q_s}\propto \boldsymbol{x}^T\langle SA^TAS\rangle_{q_s}\boldsymbol{x}-2\boldsymbol{x}^T\tilde{S}A^T\boldsymbol{y},$$

which results in

$$\langle\log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\rangle_{q_\varepsilon q_s}\propto -\frac{1}{2}\tilde{\varepsilon}(\boldsymbol{x}^T\langle SA^TAS\rangle_{q_s}\boldsymbol{x}-2\boldsymbol{x}^T\tilde{S}A^T\boldsymbol{y}).$$

Now, we can write $q_x(\boldsymbol{x})$ as

$$\begin{aligned}q_x(\boldsymbol{x}) &\propto e^{\left\langle\log\{p(\boldsymbol{x}|\tau)\}\right\rangle_{q_\tau}}e^{\left\langle\log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\theta,\mathbb{M})\}\right\rangle_{q_\theta q_s}}\\ &\propto \exp\{-\frac{1}{2}\tilde{\tau}\boldsymbol{x}^T\boldsymbol{x}\}\exp\{-\frac{1}{2}\tilde{\varepsilon}(\boldsymbol{x}^T\langle SA^TAS\rangle_{q_s}\boldsymbol{x}-2\boldsymbol{x}^T\tilde{S}A^T\boldsymbol{y})\}\\ &\propto \exp\{-\frac{1}{2}(\boldsymbol{x}^T(\tilde{\tau}I_N+\tilde{\varepsilon}\langle SA^TAS\rangle_{q_s})\boldsymbol{x}-2\tilde{\varepsilon}\boldsymbol{x}^T\tilde{S}A^T\boldsymbol{y})\}.\end{aligned}$$

Finally, $q_x(\boldsymbol{x})\sim\mathcal{N}(\tilde{\boldsymbol{x}},\Sigma_{\tilde{x}})$, where

$$\Sigma_{\tilde{x}} := (\tilde{\tau}I_N+\tilde{\tau}\langle SA^TAS\rangle_{q_s})^{-1}$$

and

$$\tilde{\boldsymbol{x}} := \tilde{\varepsilon}\Sigma_{\tilde{x}}\tilde{S}A^T\boldsymbol{y}.$$

Notice that $SA^TAS = (A^TA) \circ (\boldsymbol{s}\boldsymbol{s}^T)$. Since s_n is the outcome of a Bernoulli distribution, $\mathbb{E}_{q_s}[s_n^2] = \mathbb{E}_{q_s}[s_n] = \tilde{s}_n$, and

$$\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T = \begin{bmatrix} \tilde{s}_1^2 & \tilde{s}_1\tilde{s}_2 & \dots & \tilde{s}_1\tilde{s}_n \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{s}_n\tilde{s}_1 & \tilde{s}_n\tilde{s}_2 & \dots & \tilde{s}_n^2 \end{bmatrix}$$

Therefore,

$$\begin{aligned}\langle S^TA^TAS\rangle_{q_s} &= (A^TA) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T - \operatorname{diag}\{\tilde{\boldsymbol{s}} \circ \tilde{\boldsymbol{s}}\} + \operatorname{diag}\{\tilde{\boldsymbol{s}}\}) \\ &= (A^TA) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \operatorname{diag}\{\tilde{\boldsymbol{s}} \circ (1 - \tilde{\boldsymbol{s}})\}),\end{aligned}$$

which yields to $\boldsymbol{x} \sim \mathcal{N}(\tilde{\boldsymbol{x}}, \Sigma_{\tilde{x}})$, where

$$\Sigma_{\tilde{x}} = \Big(\tilde{\tau}I_N + \tilde{\varepsilon}\big((A^TA) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \operatorname{diag}\{\tilde{\boldsymbol{s}} \circ (1 - \tilde{\boldsymbol{s}})\})\big)\Big)^{-1}$$

and

$$\tilde{\boldsymbol{x}} = \tilde{\varepsilon}\Sigma_{\tilde{x}}\tilde{S}A^T\boldsymbol{y}$$

and therefore, the update rule for the solution value vector $\boldsymbol{x}$ is $\tilde{\boldsymbol{x}}$.

- Update rule for the support vector $\boldsymbol{s}$ using VB

$$\begin{aligned} q_{s_n}(s_n) &\sim \exp\{\langle \log\{p(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y}|\theta, \mathbb{M})\}\rangle_{q_\theta q_x}\} \\ &\propto \exp\{\langle \log\{p(\boldsymbol{x}, \boldsymbol{s}|\theta, \mathbb{M})p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}, \theta, \mathbb{M})\}\rangle_{q_\theta q_x}\} \\ &\propto \exp\{\langle \log\{p(\boldsymbol{x}, \boldsymbol{s}|\theta)\}\rangle_{q_\theta q_{\mathbb{M}}}\} \exp\{\langle \log\{p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}, \theta, \mathbb{M})\}\rangle_{q_\theta q_x}\} \\ &\propto \exp\{\langle \log\{p(s_n; \gamma_n)\}\rangle_{q_{\gamma_n}}\} \exp\{\langle \log\{p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}, \varepsilon)\}\rangle_{q_{\boldsymbol{s}-n} q_x q_\varepsilon}\} \\ &\propto \exp\{\langle \log\{\gamma_n^{s_n}(1-\gamma_n)^{1-s_n}\}\rangle_{q_{\gamma_n}}\} \exp\{\langle \log\{p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{s}, \varepsilon)\}\rangle_{q_{\boldsymbol{s}-n} q_x q_\varepsilon}\}, \end{aligned}$$

where

$$e^{\left\langle \log\{\gamma_n^{s_n}(1-\gamma_n)^{1-s_n}\}\right\rangle_{q_{\gamma_n}}} = e^{s_n\left\langle \log\gamma_n\right\rangle_{q_{\gamma_n}}} e^{(1-s_n)\left\langle \log\{1-\gamma_n\}\right\rangle_{q_{\gamma_n}}},$$

for which we have

$$\langle \log\{\gamma_n\}\rangle_{q_{\gamma_n}} \sim \text{Beta}(\alpha_{1,n}, \beta_{1,n}) = \psi(\alpha_{1,n}) - \psi(\alpha_{1,n}+\beta_{1,n})$$

and

$$\langle \log\{1-\gamma_n\}\rangle_{q_{\gamma_n}} \sim \text{Beta}(\alpha_{1,n}, \beta_{1,n}) = \psi(\beta_{1,n}) - \psi(\alpha_{1,n}+\beta_{1,n}),$$

where $\psi(\cdot)$ is digamma function, which is the logarithmic derivative of the gamma function i.e., $\psi(x) = \frac{d}{dx}\log\Gamma(x)$. Therefore,

$$e^{\left\langle \log\{\gamma_n^{s_n}(1-\gamma_n)^{1-s_n}\}\right\rangle_{q_{\gamma_n}}} = e^{s_n\left(\psi(\alpha_{1,n})-\psi(\alpha_{1,n}+\beta_{1,n})\right)} e^{(1-s_n)\left(\psi(\beta_{1,n})-\psi(\alpha_{1,n}+\beta_{1,n})\right)}.$$

Also,

$$\begin{aligned}
e^{\left\langle \log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\}\right\rangle_{q_{\boldsymbol{s}-n}q_x q_\varepsilon}} &\propto e^{-\frac{1}{2}\left\langle \varepsilon\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\right\rangle_{q_{\boldsymbol{s}-n}q_x q_\varepsilon}} \\
&\propto e^{-\frac{1}{2}\tilde{\varepsilon}\left\langle \|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2\right\rangle_{q_{\boldsymbol{s}-n}q_x q_\varepsilon}} \\
&\propto e^{-\frac{1}{2}\tilde{\varepsilon}\left\langle \sum_{m=1}^{M}(y_m-\sum_{n=1}^{N}a_{mn}s_n x_n)^2\right\rangle_{q_{\boldsymbol{s}-n}q_x q_\varepsilon}} \\
&\propto e^{-\frac{1}{2}\tilde{\varepsilon}\left\langle \left((y_1-\sum_{l\neq n}^{N}a_{1n}s_n x_n)-a_{1n}s_n x_n\right)^2+\cdots+\left((y_M-\sum_{l\neq n}^{N}a_{Ml}s_l x_l)-a_{Mn}s_n x_n\right)^2\right\rangle_{q_{\boldsymbol{s}-n}q_x q_\varepsilon}},
\end{aligned}$$

where $y_m^{-n} := y_m - \sum_{l\neq n}^{N} a_{ml}s_l x_l$, $\forall m = 1, 2, \ldots, M$. Therefore,

$$
\begin{aligned}
e^{\left\langle \log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\}\right\rangle_{q_{\boldsymbol{s}^{-n}}q_x q_\varepsilon}} &\propto e^{-\frac{1}{2}\tilde{\varepsilon}\left\langle \sum_{m=1}^{M}(a_{mn}s_n x_n - y_m^{-n})^2\right\rangle_{q_{\boldsymbol{s}^{-n}}q_x}} \\
&\propto e^{-\frac{1}{2}\tilde{\varepsilon}\left\langle \sum_{m=1}^{M}(a_{mn}^2 s_n^2 x_n^2 - 2a_{mn}s_n x_n y_m^{-n})\right\rangle_{q_{\boldsymbol{s}^{-n}}q_x}} \\
&\propto e^{-\frac{1}{2}\varepsilon \sum_{m=1}^{M}\left(a_{mn}^2 s_n^2 \left\langle x_n^2\right\rangle_{q_x} - 2a_{mn}s_n\left\langle x_n y_m^{-n}\right\rangle_{q_{\boldsymbol{s}^{-n}}q_x}\right)} \\
&\propto e^{-\frac{1}{2}\tilde{\varepsilon}\left(\|\boldsymbol{a}_n\|_2^2 s_n^2\left\langle x_n^2\right\rangle_{q_x} - 2\sum_{m=1}^{M}a_{mn}s_n\left\langle x_n y_m^{-n}\right\rangle_{q_x,q_{\boldsymbol{s}^{-n}}}\right)} \\
&\propto e^{-\frac{1}{2}\tilde{\varepsilon}\left(\|\boldsymbol{a}_n\|_2^2 s_n^2(\tilde{x}_n^2+\sigma_{\tilde{x}_n}^2) - 2s_n\left\langle x_n\right\rangle_{x_n}\sum_{m=1}^{M}a_{mn}\left\langle y_m^{-n}\right\rangle_{q_{\boldsymbol{s}^{-n}}q_x}\right)} \\
&\propto e^{-\frac{1}{2}\tilde{\varepsilon}\left(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma_{\tilde{x}_n}^2)s_n^2 - 2s_n\tilde{x}_n\boldsymbol{a}_n^T\left\langle \boldsymbol{y}^{-n}\right\rangle_{q_{\boldsymbol{s}^{-n}}q_x}\right)},
\end{aligned}
$$

where y_m^{-n} contains no x_n component and $\boldsymbol{y}^{-n} := [y_1^{-n}, y_2^{-n}, \dots, y_M^{-n}]$.

$$
\langle y_m^{-n}\rangle_{q_{\boldsymbol{s}^{-n}}q_x} = \langle y_m - \sum_{l\neq n}^{N} a_{ml}s_l x_l\rangle_{q_{\boldsymbol{s}^{-n}}q_x} = y_m - \sum_{l\neq n}^{N} a_{ml}\tilde{s}_l\tilde{x}_l
$$

$\tilde{y}_m^{-n} := \langle y_m^{-n}\rangle_{q_{\boldsymbol{s}^{-n}}q_x}$ and thus $\tilde{\boldsymbol{y}}^{-n} := \langle \boldsymbol{y}^{-n}\rangle_{q_{\boldsymbol{s}^{-n}}q_x}$. Therefore,

$$
e^{\left\langle \log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\}\right\rangle_{q_{\boldsymbol{s}^{-n}}q_x q_\varepsilon}} \propto e^{-\frac{1}{2}\tilde{\varepsilon}\left(\left(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma_{\tilde{x}_n}^2)\right)s_n^2 - 2(\tilde{x}_n\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n})s_n\right)}.
$$

Finally, we have

$$
q_{s_n}(s_n) \propto e^{\left\{s_n\left(\psi(\alpha_{1,n})-\psi(\alpha_{1,n}+\beta_{1,n})\right)+(1-s_n)\left(\psi(\beta_{1,n})-\psi(\alpha_{1,n}+\beta_{1,n})\right)-\frac{1}{2}\tilde{\varepsilon}\left(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma_{\tilde{x}_n}^2)s_n^2-2\tilde{x}_n\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n}s_n\right)\right\}}.
$$

Since s_n is an outcome of a Bernoulli random variable, for the outcome of $s_n = 0$, we have

$$
q_{s_n}(s_n) \propto \exp\left\{\psi(\beta_{1,n}) - \psi(\alpha_{1,n}+\beta_{1,n})\right\}
$$

and for the outcome of $s_n = 1$, we have

$$
q_{s_n}(s_n) \propto \exp\left\{\psi(\alpha_{1,n}) - \psi(\alpha_{1,n}+\beta_{1,n}) - \frac{1}{2}\tilde{\varepsilon}\left(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma_{\tilde{x}_n}^2) - 2\tilde{x}_n\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n}\right)\right\}.
$$

Therefore,

$$\begin{aligned} q_{s_n}(s_n) &\sim \text{Bernoulli}\Big(\frac{q_{s_n}(s_n=1)}{q_{s_n}(s_n=0)+q_{s_n}(s_n=1)}\Big) \\ &\sim \text{Bernoulli}\Big(\frac{1}{1+\frac{q_{s_n}(s_n=0)}{q_{s_n}(s_n=1)}}\Big), \end{aligned}$$

which yields

$$\begin{aligned} &q_{s_n}(s_n) \\ &\sim \quad \text{Bernoulli}\Big(\frac{1}{1+e^{\psi(\beta_{1,n})-\psi(\alpha_{1,n}+\beta_{1,n})}e^{-\psi(\alpha_{1,n})+\psi(\alpha_{1,n}+\beta_{1,n})+\frac{1}{2}\tilde{\varepsilon}\big(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma^2_{\tilde{x}_n})-2\tilde{x}_n\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n}\big)}}\Big) \\ &\quad \sim \text{Bernoulli}\Big(\frac{1}{1+e^{\big\{\psi(\beta_{1,n})-\psi(\alpha_{1,n})+\frac{1}{2}\tilde{\varepsilon}\big(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma^2_{\tilde{x}_n})-2\tilde{x}_n\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n}\big)\big\}}}\Big), \end{aligned}$$

where $\tilde{y}_m^{-n} := y_m - \sum_{l\neq n}^N a_{ml}\tilde{s}_l\tilde{x}_l$ and thus $\tilde{\boldsymbol{y}}^{-n} = \boldsymbol{y} - \sum_{l\neq n}^N \tilde{s}_l\tilde{x}_l\boldsymbol{a}_l$. The update rule for the component s_n can be then written as

$$\tilde{s}_n = \frac{1}{1+e^{\big\{\psi(\beta_{1,n})-\psi(\alpha_{1,n})+\frac{1}{2}\tilde{\varepsilon}\big(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma^2_{\tilde{x}_n})-2\tilde{x}_n\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n}\big)\big\}}},$$

which can also be written as

$$\tilde{s}_n = \frac{1}{1+c_n\kappa_n}, \ \forall n = 1, \dots, N,$$

where

$$c_n := \exp\{\psi(\beta_{1,n}) - \psi(\alpha_{1,n})\}$$

and

$$\kappa_n := \exp\Big\{\frac{1}{2}\tilde{\varepsilon}\big(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma^2_{\tilde{x}_n^2}) - 2\tilde{x}_n\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n}\big)\Big\}.$$

- Stopping rule of the algorithm

We set the stopping rule of the algorithm based on the marginalized likelihood (evidence). This is due to the fact that we would rather follow the effect of $\boldsymbol{s}$ on the evidence. Since it is the most important variable for us, if we learn $\boldsymbol{s}$, then it would be easy to compute $\boldsymbol{x}_s$. Therefore, we marginalize the distribution on $\boldsymbol{y}$ and integrate $\boldsymbol{x}$ out.

$$\begin{aligned}
p(\boldsymbol{y}|\boldsymbol{s},\varepsilon,\tau) &= \int p(\boldsymbol{y},\boldsymbol{x}|\boldsymbol{s},\varepsilon,\tau)d\boldsymbol{x} \\
&= \int p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon,\tau)p(\boldsymbol{x}|\tau)d\boldsymbol{x} \\
&= \int \frac{1}{(2\pi\varepsilon^{-1})^{\frac{M}{2}}} e^{-\frac{1}{2}\varepsilon\|\boldsymbol{y}-A(\boldsymbol{s}\circ\boldsymbol{x})\|_2^2} \frac{1}{(2\pi\tau^{-1})^{\frac{N}{2}}} e^{-\frac{1}{2}\tau\|\boldsymbol{x}\|_2^2} d\boldsymbol{x} \\
&= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}\tau^{\frac{M}{2}} \int \frac{1}{(2\pi)^{\frac{N}{2}}} e^{-\frac{1}{2}\left(\varepsilon\left(\boldsymbol{y}^T\boldsymbol{y}-2(\boldsymbol{s}\circ\boldsymbol{x})^TA^T\boldsymbol{y}+(\boldsymbol{s}\circ\boldsymbol{x})^TA^TA(\boldsymbol{s}\circ\boldsymbol{x})\right)+\tau\boldsymbol{x}^T\boldsymbol{x}\right)} d\boldsymbol{x}
\end{aligned}$$

Therefore,

$$\begin{aligned}
p(\boldsymbol{y}|\boldsymbol{s},\varepsilon,\tau) &= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}\tau^{\frac{N}{2}} \int \frac{1}{(2\pi)^{\frac{N}{2}}} e^{-\frac{1}{2}\left(\varepsilon\left(\boldsymbol{y}^T\boldsymbol{y}-2\boldsymbol{x}^TSA^T\boldsymbol{y}+\boldsymbol{x}^TSA^TAS\boldsymbol{x}\right)+\tau\boldsymbol{x}^T\boldsymbol{x}\right)} d\boldsymbol{x} \\
&= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}\tau^{\frac{N}{2}} e^{-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y}} \int \frac{1}{(2\pi)^{\frac{N}{2}}} e^{-\frac{1}{2}\left(\boldsymbol{x}^T(\tau I_N+\varepsilon SA^TAS)\boldsymbol{x}-2\varepsilon\boldsymbol{x}^TSA^T\boldsymbol{y}\right)} d\boldsymbol{x} \\
&= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}\tau^{\frac{N}{2}} e^{-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y}} |(\tau I_N+\varepsilon SA^TAS)^{-1}|^{\frac{1}{2}} \int \frac{1}{(2\pi)^{\frac{N}{2}}} \frac{1}{|(\tau I_N+\varepsilon SA^TAS)^{-1}|^{\frac{1}{2}}} \times \dots \\
& e^{-\frac{1}{2}\left(\boldsymbol{x}^T(\varepsilon SA^TAS+\tau I_N)\boldsymbol{x}-2\varepsilon\boldsymbol{x}^TSA^T\boldsymbol{y}\right)} d\boldsymbol{x} \\
&= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}\tau^{\frac{N}{2}} e^{-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y}} |(\tau I_N+\varepsilon SA^TAS)^{-1}|^{\frac{1}{2}} \int \frac{1}{(2\pi)^{\frac{N}{2}}} \frac{1}{|(\tau I_N+\varepsilon SA^TAS)^{-1}|^{\frac{1}{2}}} \times \dots \\
& e^{-\frac{1}{2}\left(\left(\boldsymbol{x}-(\tau I_N+\varepsilon SA^TAS)^{-1}\varepsilon SA^T\boldsymbol{y}\right)^T(\tau I_N+\varepsilon SA^TAS)\left(\star\right)-\varepsilon^2\boldsymbol{y}^TAS(\tau I_N+\varepsilon SA^TAS)^{-1}SA^T\boldsymbol{y}\right)} d\boldsymbol{x},
\end{aligned}$$

which results in

$$p(\boldsymbol{y}|\boldsymbol{s},\varepsilon,\tau) = \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}\tau^{\frac{N}{2}} e^{-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y}} |(\tau I_N+\varepsilon SA^TAS)^{-1}|^{\frac{1}{2}} e^{\frac{1}{2}\varepsilon^2\boldsymbol{y}^TAS(\tau I_N+\varepsilon SA^TAS)^{-1}SA^T\boldsymbol{y}}.$$

Thus,

$$\begin{aligned}
\log\{p(\boldsymbol{y}|\boldsymbol{s},\varepsilon,\tau)\} &= -\frac{M}{2}\log\{2\pi\} + \frac{M}{2}\log\{\varepsilon\} + \frac{N}{2}\log\{\tau\} - \frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y} + \dots \\
& \frac{1}{2}\log\{|(\tau I_N+\varepsilon SA^TAS)^{-1}|\} + \frac{1}{2}\varepsilon^2\boldsymbol{y}^TAS(\tau I_N+\varepsilon SA^TAS)^{-1}SA^T\boldsymbol{y}.
\end{aligned}$$

Notice that

$$-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y}+\frac{1}{2}\varepsilon^2\boldsymbol{y}^T AS(\tau I_N+\varepsilon SA^T AS)^{-1}SA^T\boldsymbol{y} = -\frac{1}{2}\boldsymbol{y}^T\big(I_M-\varepsilon AS(\tau I_N+\varepsilon SA^T AS)^{-1}SA^T\big)\boldsymbol{y}$$

Also,

$$\begin{aligned}\frac{N}{2}\log\{\tau\}+\frac{1}{2}\log\{|(\tau I_N+\varepsilon SA^TAS)^{-1}|\} &= \frac{1}{2}\big(\log\{\tau I_N\}+\log\{|(\tau I_N+\varepsilon SA^TAS)^{-1}|\}\big)\\ &= \frac{1}{2}\log\{|\tau I_N||(\tau I_N+\varepsilon SA^TAS)|^{-1}\}\\ &= \frac{1}{2}\log\{|(\tau I_N)(\tau I_N+\varepsilon SA^TAS)^{-1}|\}\\ &= -\frac{1}{2}\log\{|(\tau^{-1}I_N)(\tau I_N+\varepsilon SA^TAS)|\}\\ &= -\frac{1}{2}\log|I_N+\frac{\varepsilon}{\tau}SA^TAS|.\end{aligned}$$

Remark 5.2: Based on Sylvester's determinant theorem, we have

$$det(I_m+A_{m\times n}B_{n\times m}) = det(I_n+BA)$$

Therefore,

$$\log\{|I_N+\frac{\varepsilon}{\tau}SA^TAS|\} = \log\{|I_M+\frac{\varepsilon}{\tau}AS^2A^T|\}.$$

Thus,

$$\begin{aligned}L :&= \log p(\boldsymbol{y}|\boldsymbol{s},\varepsilon,\tau)\\ &= -\frac{M}{2}\log\{2\pi\}+\frac{M}{2}\log\varepsilon-\frac{1}{2}\log\{|I_M+\frac{\varepsilon}{\tau}AS^2A^T|\}-\\ &\frac{1}{2}\varepsilon\boldsymbol{y}^T\big(I_M-\varepsilon AS(\tau I_N+\varepsilon SA^TAS)^{-1}SA^T\big)\boldsymbol{y}.\end{aligned}$$

Then, for comparison purposes of $L^{[t+1]}$ with $L^{[t]}$ in the updating process, we have

$$L \propto \frac{M}{2}\log\{\varepsilon\}+\frac{1}{2}\log\{|(I_M+\frac{\varepsilon}{\tau}AS^2A^T)^{-1}|\}-\frac{1}{2}\varepsilon\boldsymbol{y}^T\big(I_M-\varepsilon AS(\tau I_N+\varepsilon SA^TAS)^{-1}SA^T\big)\boldsymbol{y}.$$

Remark 5.3: According to the matrix inverse lemma, we have

$$(A+BCD)^{-1}=A^{-1}-A^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1}.$$

Therefore,

$$I_M-\varepsilon AS(\tau I_N+\varepsilon SA^TS)^{-1}SA^T=(I+\varepsilon AS\tau^{-1}SA^T)^{-1}=(I_M+\frac{\varepsilon}{\tau}AS^2A^T)^{-1}.$$

Thus,

$$\begin{aligned}L &\propto \frac{M}{2}\log\{\varepsilon\}+\frac{1}{2}\log\{|(I_M+\frac{\varepsilon}{\tau}AS^2A^T)^{-1}|\}-\frac{1}{2}\varepsilon\boldsymbol{y}^T(I_M+\frac{\varepsilon}{\tau}AS^2A^T)^{-1}\boldsymbol{y}\\ &\propto \frac{1}{2}\big(\log\{|\varepsilon I_M|\}+\log\{|(I_M+\frac{\varepsilon}{\tau}AS^2A^T)^{-1}|\}\big)-\frac{1}{2}\boldsymbol{y}^T(\varepsilon^{-1}I_M+\frac{1}{\tau}AS^2A^T)^{-1}\boldsymbol{y}\\ &\propto \frac{1}{2}\log\{|\varepsilon^{-1}I_M|^{-1}|I_M+\frac{\varepsilon}{\tau}AS^2A^T|^{-1}\}-\frac{1}{2}\boldsymbol{y}^T(\varepsilon^{-1}I_M+\frac{1}{\tau}AS^2A^T)^{-1}\boldsymbol{y}\\ &\propto \frac{1}{2}\log\{|\varepsilon^{-1}I_M(I_M+\frac{\varepsilon}{\tau}AS^2A^T)^{-1}|\}-\frac{1}{2}\boldsymbol{y}^T(\varepsilon^{-1}I_M+\frac{1}{\tau}AS^2A^T)^{-1}\boldsymbol{y}\\ &\propto \log\{|\varepsilon^{-1}I_M+\frac{1}{\tau}AS^2A^T|^{-1}\}-\boldsymbol{y}^T(\varepsilon^{-1}I_M+\frac{1}{\tau}AS^2A^T)^{-1}\boldsymbol{y}.\end{aligned}$$

Therefore, $L \propto \log\{|\Sigma_0|\}-\boldsymbol{y}^T\Sigma_0\boldsymbol{y}$, where $\Sigma_0 := (\varepsilon^{-1}I_M+\tau^{-1}AS^2A^T)^{-1}$, which yields to

$$-L \propto \log\{|\Sigma_0^{-1}|\}+\boldsymbol{y}^T\Sigma_0\boldsymbol{y}.$$

This means that

$$p(\boldsymbol{y}|\mathbf{s},\varepsilon,\tau)=\frac{1}{(2\pi)^{\frac{M}{2}}}\frac{1}{|\Sigma_0^{-1}|^{\frac{1}{2}}}\exp\{-\frac{1}{2}\boldsymbol{y}^T\Sigma_0\boldsymbol{y}\}$$

or equivalently, $p(\boldsymbol{y}|\boldsymbol{s},\varepsilon,\tau)\sim\mathcal{N}(\mathbf{0},\Sigma_0^{-1})$.

5.7.2 Gaussians-Inverse Gammas Modeling and the SBL-VB(GiG) Algorithm

- Update rule for the precision τ_n of the nth component of the solution vector $\boldsymbol{x}$ using VB

$$\begin{aligned}
q(\tau_n) &\propto p(\tau_n; a_0, b_0) \exp\{\langle \log p(\boldsymbol{x}|T)\rangle_{q_{x_n}}\} \\
&\propto \tau_n^{a_0-1} e^{-b_0\tau_n} \exp\{\langle \log\{\prod_{n=1}^{N} p(x_n; \tau_n^{-1})\}\rangle_{q_{x_n}}\} \\
&\propto \tau_n^{a_0-1} e^{-b_0\tau_n} \exp\{\langle \log\{\prod_{n=1}^{N} (\tau_n^{\frac{1}{2}} \exp\{-\frac{\tau_n}{2} x_n^2\})\}\rangle_{q_{x_n}}\} \\
&\propto \tau_n^{a_0-1} e^{-b_0\tau_n} \exp\{\langle \log\{\tau_n^{\frac{1}{2}} e^{-\frac{\tau_n}{2} x_n^2}\}\rangle_{q_{x_n}}\} \\
&\propto \tau_n^{a_0+\frac{1}{2}-1} e^{-b_0\tau_n} \exp\{-\frac{\tau_n}{2} \langle x_n^2\rangle_{q_{x_n}}\} \\
&\propto \tau_n^{(a_0+\frac{1}{2})-1} e^{-b_0\tau_n} e^{-\frac{\tau_n}{2}(\tilde{x}_n^2+\sigma_{\tilde{x}_n}^2)} \\
&\propto \tau^{(a_0+\frac{1}{2})-1} e^{-\left(b_0+\frac{1}{2}(\tilde{x}_n^2+\sigma_{\tilde{x}_n}^2)\right)\tau_n},
\end{aligned}$$

where $T := \mathrm{diag}\{\tau_1, \ldots, \tau_N\}$. Therefore,

$$q(\tau_n) \sim \mathrm{Gamma}\big(a_0 + \frac{1}{2}, b_0 + \frac{1}{2}(\tilde{x}_n^2 + \sigma_{\tilde{x}_n}^2)\big).$$

The update rule for τ_n can be then defined as follows

$$\tilde{\tau}_n = \frac{a_0 + \frac{1}{2}}{b_0 + \frac{1}{2}(\tilde{x}_n^2 + \sigma_{\tilde{x}_n}^2)}, \ \forall n = 1, 2, \ldots, N.$$

- Update rule for the noise precision ε using VB

$$\begin{aligned}
q(\varepsilon) &\propto p(\varepsilon; \theta_0, \theta_1) \exp\{\langle \log p(\boldsymbol{y}|\boldsymbol{x}, \varepsilon)\rangle_{q_x}\} \\
&\propto \varepsilon^{\theta_0-1} e^{-\theta_1\varepsilon} \exp\{\langle \log\{\varepsilon^{\frac{M}{2}\exp(-\frac{1}{2}\varepsilon\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2)}\}\rangle_{q_x}\} \\
&\propto \varepsilon^{\theta_0+\frac{M}{2}-1} e^{-\varepsilon(\theta_1+\frac{1}{2}\langle\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2\rangle_{q_x})}.
\end{aligned}$$

Therefore,

$$q(\varepsilon) \sim \mathrm{Gamma}\big(\theta_0 + \frac{M}{2}, \theta_1 + \frac{1}{2}\langle\|\boldsymbol{y} - A\boldsymbol{x}\|_2^2\rangle_{q_x}\big).$$

The update rule for ε can be then defined as follows

$$\tilde{\varepsilon} = \frac{\theta_0 + \frac{M}{2}}{\theta_1 + \frac{1}{2}\langle\!\langle \|\boldsymbol{y} - A\boldsymbol{x}\|_2^2 \rangle\!\rangle_{q_x}},$$

where $\langle\!\langle \|\boldsymbol{y} - A\boldsymbol{x}\|_2^2 \rangle\!\rangle_{q_x}$ is obtained as follows

$$\begin{aligned}\langle \|\boldsymbol{y} - A\boldsymbol{x}\|_2^2 \rangle_{q_x} &= \boldsymbol{y}^T\boldsymbol{y} - 2\langle \boldsymbol{x}\rangle_{q_x}^T A^T \boldsymbol{y} + \langle \boldsymbol{x}^T A^T A \boldsymbol{x}\rangle_x \\ &= \boldsymbol{y}^T\boldsymbol{y} - 2\tilde{\boldsymbol{x}}^T A^T \boldsymbol{y} + \langle \boldsymbol{x}^T A^T A \boldsymbol{x}\rangle_{q_x},\end{aligned}$$

Also,

$$\begin{aligned}\langle \boldsymbol{x}^T A^T A \boldsymbol{x}\rangle_{q_x} &\propto \operatorname{tr}\left(\langle \boldsymbol{x}^T A^T A \boldsymbol{x}\rangle_{q_x}\right) \\ &= \operatorname{tr}\left(\langle \boldsymbol{x}\boldsymbol{x}^T\rangle_{q_x} A^T A\right) \\ &= \operatorname{tr}\big((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}})A^T A\big).\end{aligned}$$

Therefore,

$$\langle \|\boldsymbol{y} - A\boldsymbol{x}\|_2^2 \rangle_{q_x} = \boldsymbol{y}^T\boldsymbol{y} - 2\tilde{\boldsymbol{x}}^T A^T \boldsymbol{y} + \operatorname{tr}\big((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}})A^T A\big).$$

The update rule for ε can be then written as

$$\tilde{\varepsilon} = \frac{\theta_0 + \frac{M}{2}}{\theta_1 + \frac{1}{2}\tilde{\Psi}},$$

Where

$$\tilde{\Psi} := \boldsymbol{y}^T\boldsymbol{y} - 2\tilde{\boldsymbol{x}}^T A^T \boldsymbol{y} + \operatorname{tr}\big((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}})A^T A\big)$$

- Update rule for the solution vector $\boldsymbol{x}$ using VB

$$\begin{aligned}q_x(\boldsymbol{x}) &\propto \exp\{\langle \log\{p(\boldsymbol{x}, \boldsymbol{y}|\theta, \mathbb{M})\}\rangle_{q_\theta}\} \\ &\propto \exp\{\langle \log\{p(\boldsymbol{x}|\theta, \mathbb{M})p(\boldsymbol{y}|\boldsymbol{x}, \theta, \mathbb{M})\}\rangle_{q_\theta}\} \\ &\propto \exp\{\langle \log\{p(\boldsymbol{x}|\theta)\}\rangle_{q_\theta}\}\exp\{\langle \log\{p(\boldsymbol{y}|\boldsymbol{x}, \theta, \mathbb{M})\}\rangle_{q_\theta}\} \\ &\propto \exp\{\langle \log\{p(\boldsymbol{x}|T)\}\rangle_{q_\tau}\}\exp\{\langle \log\{p(\boldsymbol{y}|\boldsymbol{x}, \varepsilon)\}\rangle_{q_\varepsilon}\}\end{aligned}$$

For the purpose of updating the elements of $\boldsymbol{x}$, we have

$$\begin{aligned} p(\boldsymbol{y},\boldsymbol{x},\varepsilon) &\propto \varepsilon^{\frac{M}{2}} \exp\{-\frac{1}{2}\varepsilon\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2\} \\ &\propto \exp\{-\frac{1}{2}\varepsilon\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2\}. \end{aligned}$$

Therefore,

$$\begin{aligned} \langle \log\{p(\boldsymbol{y}|\boldsymbol{x},\varepsilon)\}\rangle_{q_\varepsilon} &\propto -\frac{1}{2}\langle\varepsilon\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2\rangle_{q_x} \\ &\propto -\frac{1}{2}\langle\varepsilon\rangle_{q_\varepsilon}\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2 \\ &\propto -\frac{1}{2}\tilde{\varepsilon}\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2. \end{aligned}$$

Now, we can write $q_x(\boldsymbol{x})$ as

$$\begin{aligned} q_x(\boldsymbol{x}) &\propto e^{\langle \log\{p(\boldsymbol{x}|T)\}\rangle_{q_T}} e^{\langle \log\{p(\boldsymbol{y}|\boldsymbol{x},\theta,\mathbb{M})\}\rangle_{q_\theta}} \\ &\propto \exp\{-\frac{1}{2}\boldsymbol{x}^T\tilde{T}\boldsymbol{x}\}\exp\{-\frac{1}{2}\tilde{\varepsilon}(\boldsymbol{x}^TA^TA\boldsymbol{x}-2\boldsymbol{x}^TA^T\boldsymbol{y})\} \\ &\propto \exp\{-\frac{1}{2}\big(\boldsymbol{x}^T(\tilde{T}+\tilde{\varepsilon}A^TA\boldsymbol{x}-2\tilde{\varepsilon}\boldsymbol{x}^TA^T\boldsymbol{y})\big)\}, \end{aligned}$$

where $\tilde{T}:=\operatorname{diag}\{\tilde{\tau}_1,\dots,\tilde{\tau}_N\}$. Therefore, $q_x(\boldsymbol{x})\sim\mathcal{N}(\tilde{\boldsymbol{x}},\Sigma_{\tilde{x}})$, where

$$\Sigma_{\tilde{x}} := (\tilde{T}+\tilde{\varepsilon}A^TA)^{-1}$$

and

$$\tilde{\boldsymbol{x}} := \tilde{\varepsilon}\Sigma_{\tilde{x}}A^T\boldsymbol{y}.$$

- Stopping rule of the algorithm

We set the stopping rule of the algorithm based on the marginalized log-likelihood (evidence).

$$\begin{aligned}
p(\boldsymbol{y}|\varepsilon,T) &= \int p(\boldsymbol{y},\boldsymbol{x}|\varepsilon,T)d\boldsymbol{x} \\
&\int p(\boldsymbol{y}|\boldsymbol{x},\varepsilon,T)p(\boldsymbol{x}|T)d\boldsymbol{x} \\
&= \int \frac{1}{(2\pi\varepsilon^{-1})^{\frac{M}{2}}} e^{-\frac{1}{2}\varepsilon\|\boldsymbol{y}-A\boldsymbol{x}\|_2^2} \frac{1}{(2\pi|\tilde{T}|)^{\frac{1}{2}}} e^{-\frac{1}{2}\boldsymbol{x}^T\tilde{T}\boldsymbol{x}} d\boldsymbol{x} \\
&= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}|T|^{\frac{1}{2}} \int \frac{1}{(2\pi)^{\frac{N}{2}}} e^{-\frac{1}{2}\left(\varepsilon(\boldsymbol{y}^T\boldsymbol{y}-2\boldsymbol{x}^TA^T\boldsymbol{y}+\boldsymbol{x}^TA^TA\boldsymbol{x})+\boldsymbol{x}^T\tilde{T}\boldsymbol{x}\right)} d\boldsymbol{x}
\end{aligned}$$

Therefore,

$$\begin{aligned}
p(\boldsymbol{y}|\varepsilon,T) &= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}|T|^{\frac{1}{2}} \int \frac{1}{(2\pi)^{\frac{N}{2}}} e^{-\frac{1}{2}\left(\varepsilon(\boldsymbol{y}^T\boldsymbol{y}-2\boldsymbol{x}^TA^T\boldsymbol{y}+\boldsymbol{x}^TA^TA\boldsymbol{x})+\boldsymbol{x}^TT\boldsymbol{x}\right)} d\boldsymbol{x} \\
&= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}|T|^{\frac{1}{2}} e^{-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y}} \int \frac{1}{(2\pi)^{\frac{N}{2}}} e^{-\frac{1}{2}\left(\boldsymbol{x}^T(\varepsilon A^TA+T)\boldsymbol{x}-2\varepsilon\boldsymbol{x}^TA^T\boldsymbol{y}\right)} d\boldsymbol{x} \\
&= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}|T|^{\frac{1}{2}} e^{-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y}} |(T+\varepsilon A^TA)^{-1}|^{\frac{1}{2}} \times \\
&\int \frac{1}{(2\pi)^{\frac{N}{2}}} \frac{1}{|(T+\varepsilon A^TA)^{-1}|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\boldsymbol{x}^T(\varepsilon A^TA+T)\boldsymbol{x}-2\varepsilon\boldsymbol{x}^TA^T\boldsymbol{y}\right)} d\boldsymbol{x} \\
&= \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}|T|^{\frac{1}{2}} e^{-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y}} |(T+\varepsilon A^TA)^{-1}|^{\frac{1}{2}} \int \frac{1}{(2\pi)^{\frac{N}{2}}} \frac{1}{|(T+\varepsilon A^TA)^{-1}|^{\frac{1}{2}}} \times \dots \\
&e^{-\frac{1}{2}\left(\left(\boldsymbol{x}-(T+\varepsilon A^TA)^{-1}\varepsilon A^T\boldsymbol{y}\right)^T(T+\varepsilon A^TA)\left(\boldsymbol{x}-(T+\varepsilon A^TA)^{-1}\varepsilon A^T\boldsymbol{y}\right)-\varepsilon^2\boldsymbol{y}^TA(T+\varepsilon A^TA)^{-1}A^T\boldsymbol{y}\right)} d\boldsymbol{x}.
\end{aligned}$$

Notice that

$$p(\boldsymbol{y}|\boldsymbol{x},\varepsilon,T) = \frac{1}{(2\pi)^{\frac{M}{2}}}\varepsilon^{\frac{M}{2}}|T|^{\frac{1}{2}} e^{-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y}} |(T+\varepsilon A^TA)^{-1}|^{\frac{1}{2}} e^{\frac{1}{2}\varepsilon^2\boldsymbol{y}^TA(T+\varepsilon A^TA)^{-1}A^T\boldsymbol{y}}.$$

Thus,

$$\begin{aligned}
&\log\{p(\boldsymbol{y}|\varepsilon,T)\} = \\
&-\frac{M}{2}\log\{2\pi\} + \frac{M}{2}\log\varepsilon + \frac{1}{2}\log\{|T|\} - \frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y} + \frac{1}{2}\log\{|(T+\varepsilon A^TA)^{-1}|\} + \\
&\frac{1}{2}\varepsilon^2\boldsymbol{y}^TA(T+\varepsilon A^TA)^{-1}A^T\boldsymbol{y}.
\end{aligned}$$

Also,

$$-\frac{1}{2}\varepsilon\boldsymbol{y}^T\boldsymbol{y} + \frac{1}{2}\varepsilon^2\boldsymbol{y}^T A(T + \varepsilon A^T A)^{-1} A^T\boldsymbol{y} = -\frac{1}{2}\boldsymbol{y}^T(I_M - \varepsilon A(T + \varepsilon A^T A)^{-1} A^T)\boldsymbol{y}$$

and,

$$\begin{aligned}\frac{1}{2}\log\{|T|\} + \frac{1}{2}\log\{|(T + \varepsilon A^T A)^{-1}|\} &= \frac{1}{2}\big(\log\{|T|\} + \log\{|(T + \varepsilon A^T A)^{-1}|\}\big)\\ &= \frac{1}{2}\log\{|T(T + \varepsilon A^T A)^{-1}|\}\\ &= -\frac{1}{2}\log\{|T^{-1}(T + \varepsilon A^T A)|\}\\ &= -\frac{1}{2}\log\{|T + \varepsilon T^{-1} A^T A|\}.\end{aligned}$$

Remark 5.4: Based on Sylvester's determinant theorem, we have

$$\det\{(I_m + A_{m\times n}B_{n\times m})\} = \det(I_n + BA).$$

Therefore,

$$\log\{|I_N + \varepsilon A^T T^{-1} A|\} = \log\{|I_M + \varepsilon A T^{-1} A^T|\}.$$

Thus,

$$\begin{aligned}L :&= \log\{p(\boldsymbol{y}|\varepsilon, T)\} =\\ &-\frac{M}{2}\log\{2\pi\} + \frac{M}{2}\log\{\varepsilon\} - \frac{1}{2}\log\{|I_M + \varepsilon A T^{-1} A^T|\} -\\ &\frac{1}{2}\varepsilon\boldsymbol{y}^T\big(I_M - \varepsilon A(T + \varepsilon A^T A)^{-1} A^T\big)\boldsymbol{y}.\end{aligned}$$

Then, for the comparison purposes of $L^{[t+1]}$ with $L^{[t]}$ in the updating process, we have

$$L \propto \frac{M}{2}\log\{\varepsilon\} + \frac{1}{2}\log\{|(I_M + \varepsilon A T^{-1} A^T)^{-1}|\} - \frac{1}{2}\varepsilon\boldsymbol{y}^T\big(I_M - \varepsilon A(T + \varepsilon A^T A)^{-1} A^T\big)\boldsymbol{y}.$$

Remark 5.5: According to the matrix inversion lemma, we have

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}.$$

Therefore,

$$I_M - \varepsilon A(T + \varepsilon A^T)^{-1}A^T = (I + \varepsilon AT^{-1}A^T)^{-1} = (I_M + \varepsilon AT^{-1}A^T)^{-1}.$$

Thus,

$$\begin{aligned}
L &\propto \frac{M}{2}\log\varepsilon + \frac{1}{2}\log\{|(I_M + +\varepsilon AT^{-1}A^T)^{-1}|\} - \frac{1}{2}\varepsilon \boldsymbol{y}^T(I_M + \varepsilon AT^{-1}A^T)^{-1}\boldsymbol{y} \\
&\propto \frac{1}{2}(\log\{|\varepsilon I_M|\} + \log\{|(I_M + \varepsilon AT^{-1}A^T)^{-1}|\}) - \frac{1}{2}\boldsymbol{y}^T(I_M + \varepsilon AT^{-1}A^T)^{-1}\boldsymbol{y} \\
&\propto \frac{1}{2}\log\{|\varepsilon^{-1}I_M|^{-1}|I_M + \varepsilon AT^{-1}A^T|^{-1}\} - \frac{1}{2}\boldsymbol{y}^T(\varepsilon^{-1}I_M + AT^{-1}A^T)^{-1}\boldsymbol{y} \\
&\propto \frac{1}{2}\log\{|\varepsilon^{-1}I_M(I_M + \varepsilon AT^{-1}A^T)^{-1}|\} - \frac{1}{2}\boldsymbol{y}^T(\varepsilon^{-1}I_M + AT^{-1}A^T)^{-1}\boldsymbol{y} \\
&\propto \log\{|\varepsilon^{-1}I_M + AT^{-1}A^T|^{-1}\} - \boldsymbol{y}^T(\varepsilon^{-1}I_M + AT^{-1}A^T)^{-1}\boldsymbol{y}.
\end{aligned}$$

Therefore, $L \propto \log\{|\Sigma_0| - \boldsymbol{y}^T\Sigma_0\boldsymbol{y}\}$, where $\Sigma_0 := (\varepsilon^{-1}I_M + AT^{-1}A^T)^{-1}$, which yields to

$$-L \propto \log\{|\Sigma_0^{-1}| + \boldsymbol{y}^T\Sigma_0\boldsymbol{y}\}.$$

This means that

$$p(\boldsymbol{y}|\varepsilon, T) = \frac{1}{(2\pi)^{\frac{M}{2}}}\frac{1}{|\Sigma_0^{-1}|^{\frac{1}{2}}}\exp\{-\frac{1}{2}\boldsymbol{y}^T\Sigma_0\boldsymbol{y}\}$$

or equivalently, $p(\boldsymbol{y}|\varepsilon, T) \sim \mathcal{N}(\mathbf{0}, \Sigma_0^{-1})$.

CHAPTER 6
EXPLORATION *VS.* DATA REFINEMENT VIA MULTIPLE MOBILE SENSORS

6.1 Introduction

In this chapter, we examine the configuration of multiple mobile sensors to explore an unknown region, where the exploration trades off between two desiderata: to continue taking data in a region known to be interesting with the intent of refining the measurements *vs.* taking data in unobserved areas to attempt to discover new interesting regions. Making reasonable and practical decisions to simultaneously fulfill both goals of exploration and data refinement seem to be hard and contradictory. For this purpose, we propose a general framework that makes value-laden decisions for the trajectory of mobile sensors. The proposed framework employs a Gaussian process regression model to predict the interesting phenomena at unseen locations. Then, the decision making on the trajectories of sensors is performed using an epistemic utility controller. An example is provided to illustrate the importance of the proposed framework.

We consider the general problem of exploration using multiple mobile sensors, or fixed sensors whose field of view is reconfigurable. The problem is to *locate* regions where interesting phenomena occur and then, having found such regions, to expend sensor capability to *refine* data in there while continuing to search for new interesting regions. That is, after an initial discovery there is a tradeoff between increasing knowledge by taking more measurements in regions already known to be of interest, and increasing knowledge by exploring in regions where interesting phenomena may exist but are not known to be present. Due to the uncertainties of exploration, the problem is not posed as one of optimal path (or resource) planning, but as a problem which balances the competing imperatives of refining measurements while exploring new territory.

The paradigm developed here is applicable to general exploration problems, but to illustrate how the principles apply we consider a surrogate problem of space exploration in the vicinity of a planet using a constellation of satellites. In general, the design of a satellite constellation can be divided into three main categories, depending on the mission objectives. In the first category there is a single launch event with a fixed and predetermined constellation. Here, the orbit design process may turn into solving a global optimization problem. The mission objectives in this category could be for global, zonal or regional coverage of the Earth [129–136]. The second category involves multiple launch events, in which a small constellation is initially deployed and then is gradually expanded based on the increase in the future demand or market change [137, 138]. In this case, as additional satellites are launched, it may be necessary to reconfigure the existing constellation. The third category considers a single launch event but with an adaptive constellation, also referred to as operationally responsive constellation [139]. Besides the required energy for keeping the satellites in the operating constellation, this category further assumes that the satellites are capable of performing some other orbital changes. An application of this category can be found in the surveillance satellites used to monitor ground targets [139]. This category can also be used for exploration problems in which the satellites are deployed and then the goal is to either change or tune their orbital planes in order to find, capture, and track the most interesting features of the phenomenon of interest (PoI). The exploration problem with multiple mobile sensors that we investigate here falls most closely within this third class.

There exist many approaches for sensor array configuration or placement [26–31, 140–143]. For example, Zhang provide a necessary condition for optimal sensor placement in two-dimensional space using the algebric structure of sensors [141]. The problem of array configuration in a remote sensing formulation is addressed in [144], in which a statistical optimality criterion is used to identify a solution. By contrast, here a more dynamic, exploratory stance is taken, to trade off typically repeated (or related) measurements against measurements in new areas. Here, we focus on approaches that use Gaussian processes (GPs) modeling to formulate the sensor placement problem [26–31]. A Gaussian process model is useful for modeling spatiotemporal data. GPs have been used for decades as a supervised learning tool for regression problems known as Gaussian process regression (GPR) models [21, 22], and are also referred to as kriging, named after the mining engineer D.G. Krige in the geostatistics literature [23–25]. GPR models are used here as a spatiotemporal interpolator/extrapolator tool to predict the behavior of the PoI at unsampled data points. GPR models and kriging methods are applicable to a wide variety of problems such as the prediction and estimation of temperature, precipitation, missing pixel and un-mixing of pixels in hyperspectral imaging (HSI), human head pose estimation, concentration of carbon dioxide in the atmosphere, etc. [22, 145–150]. As an example in HSI, one main objective is to un-mix the spectral information to make inference of the composing materials in the scene. Imbiriba et al. in [145] consider a nonlinear model where the underlying function is governed by a Gaussian process model to detect the nonlinearly mixed pixels. Xing et al. in [148] introduce an algorithm for dictionary learning based on a GP prior to remove the noise and infer the missing data in HSI. Another example is to predict the temperature based on the available data collected from the meteorological stations. For instance, Wu and Li [146] apply the residual kriging method to predict the average monthly temperature at over 500 unknown locations in the United States.

GPs have also found been applied in sensor placement problems [26–31]. In [30], Garnett et al. propose a Bayesian optimization algorithm for sensor placement based on GP modeling. As an experiment, they applied their approach to place 50 sensors around the UK to measure the air temperature. A typical sensor placement technique is to use the variances associated with the maximum a *posteriori* (MAP) estimates of GP as a measure representing the amount of uncertainty in the region. This leads to placing the sensors at locations with the highest variance (entropy) [27, 28], to reduce the overall entropy. This characterization of the quality of sensor placements seems to be naive due to the following reasons. As observed by [28,29], sensor placement using only the measure of variance usually forces the sensors to be placed at the borders of the region under study. This is due to the fact that there is no measurement outside of the region and as a result the borders tend to have very few measurements in their neighborhood to be used in the training set of GPs. To tackle this issue, in [26] a mutual information criterion is proposed in order to place the sensors at the locations most informative about the unseen locations. However, the optimization turns out to be an NP-hard problem because it involves solving for a combinatorial optimization problem of maximizing the mutual information between the chosen locations and the locations which are not selected [29]. In order to rackle this issue, an approximate form of mutual information optimization approach is considered to select informative sensor locations via exploiting the sub-modularity property of mutual information in [29]. However, if we continue to perform sensor placement successively using either the entropy of the region or the mutual information criterion, there is a high chance ending up with some sort of uniform sampling (equally spaced sensor placements) in the region. This is because the placements tend to occur at the locations far away from the visited locations of the sensors. These criteria only fulfill the exploration goals without taking into account refining measurements of the interesting features about the underlying phenomenon. Also, the available budget on sensors may not allow us to have widely scattered sensor placements (or place the sensors far away from their previous positions) in applications such as the space missions that exemplify our work.

In this chapter we devise a general framework to specify the trajectory of mobile sensors (e.g., sensor-bearing satellites) in order to find and characterize the important features of phenomena in a region of interest. Although the suggested framework is applicable to the mobile sensor placement problems, it is different than the other studies on typical mobile sensors due to following reasons. First, our set of mobile sensors are expected to follow specific paths, once the trajectories are determined. This means that even if we are inclined to favor only the exploration goal, we cannot arbitrarily place the sensors at particlar *locations*, but rather on particular *trajectories*. Thus, it is not possible to arbitrarily scatter the sensors in the region under study. Secondly, we do not only seek the exploration goal, but rather to fulfill both exploration and data refinement. Here, we assume that there are some interesting phenomena occurring in the region under study, probably scattered sparsely over the region. Therefore, the measure of mutual information or entropy is not enough to reach a single objective. The proposed framework adaptively makes value-laden decisions on the trajectories. This framework consists of two main stages. The first stage is the prediction stage, where we apply a Gaussian process regression model to predict the PoI at the unsampled locations. The GPR not only provides interpolated/extrapolated values, but also the variance of the estimated values. After collecting the initial measurements over the region, the GPR model predicts the behavior of the PoI at the unseen locations. Then, the question is how to decide on the next set of trajectories, based on the information obtained from the previous (GPR) stage. There is a tradeoff between repeating measurements in the same (or nearby) locations in order to refine information about that region where it is known that interesting things are happening *vs.* making measurements in new locations (exploration) with the possibility of discovering additional interesting information. There is not sufficient information to obtain an optimal solution *a priori*, since the available information is local (the result of previous measurements) and may change over time. The decision between improving information in the neighborhood of known PoI and exploring new territory in the hope of discovering additional interesting locations can be hard or contradictory. In order to deal with the above issues, we set up the decision making based on epistemic utility theory [32–35], which forms the second stage of our framework. Epistemic utility is able to provide decisions, called *satisficing* decisions, based on the local rather than global information in a way that specifically trades off between two different goals [151].

The remainder of this chapter is organized as follows. In Section 6.2, some background of the GPR model is presented. Then, decision making based on the epistemic utility controller is described. In Section 6.3, these tools are applied to the problem of trajectory determination of mobile sensors. Section 6.4 contains an example illustrating the effectiveness of the proposed approach.

6.2 Theoretical Preliminaries

This section is divided into two parts. The first part is devoted to a review on Gaussian process regression models, and the second part describes epistemic utility theory. These two tools will serve as the backbone of our proposed framework.

6.2.1 Gaussian Process Regression

Gaussian processes (GPs) are widely used for modeling a feature or PoI based on observed spatiotemporal data. A GP can be used as a tool for either classification or regression, where the regression application of GP is called Gaussian process regression (GPR) [21,22]. It is assumed that the PoI can be evaluated via an unknown and probably nonlinear function, which we denote by $f(\cdot)$. The arguments of the function comprise a variable set $\boldsymbol{u}$ referred to as the input data. For example, $\boldsymbol{u}$ can be defined as $\boldsymbol{u} = [u_x, u_y, u_z, t]^T$, where (u_x, u_y, u_z) and t denote the spatial and temporal information about the measurements, respectively. Unlike parametric models such as linear regression, GP is non-parametric. In GP one defines a probability distribution function as a prior over the unknown function $f(\cdot)$, directly. In other words, GP defines a distribution over functions in the function space and the inference is performed directly in this space [22]. This is more general than a parametric model such as Bayesian linear regression, where the prior distribution is defined over the space of parameters. The GP model treats any observation as an outcome of a Gaussian random variable, and all of these random variables are jointly Gaussian. With this setting, any well-defined GP model only needs a mean accompanied with a positive definite covariance function. Under this assumption, GP provides a posterior distribution over the unknown function f once data are observed. Therefore, for any set of N observations with the input data set $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N\}$, GP assumes that the distribution $p(f(\boldsymbol{u}_1), \ldots, f(\boldsymbol{u}_N))$ is jointly Gaussian with some mean $\boldsymbol{\mu}(U)$ and a covariance matrix $K(U)$, where $U := [\boldsymbol{u}_1, ..., \boldsymbol{u}_N]$. The entry in row i and column j of $K(U)$ is denoted by $[K(U)]_{ij} = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$, where $\kappa(.,.)$ is a positive definite kernel function. The kernel function specifies the covariance between pairs of random variables at the corresponding data points. The GP model is defined as follows [152]:

$$\boldsymbol{f}(U) \sim \mathcal{GP}(\boldsymbol{\mu}(U), K(U)), \tag{6.1}$$

where $\boldsymbol{f}(U) := [f(\boldsymbol{u}_1), \ldots, f(\boldsymbol{u}_N)]^T$ and $\boldsymbol{\mu}(U) := [\mu(\boldsymbol{u}_1), \ldots, \mu(\boldsymbol{u}_N)]^T$. For the regression purposes, GPR predicts the behavior of the PoI at unseen data points using the available training data set.

The GPR can also handle noisy observations. Suppose we have access to a set of N noisy observations $\boldsymbol{y}=\boldsymbol{f}(U)+\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\sigma_n^2 I_N)$ and $y_n=f(\boldsymbol{u}_n)+\epsilon_n, \forall n=1,\ldots,N$. The pair $(\boldsymbol{u}_n, y_n)$ is the nth training data. The goal is to predict the underlying function f evaluated at some other input data set $U_\star$ i.e., inferring $\boldsymbol{f}(U_\star)$, where $U_\star := [\boldsymbol{u}_{\star,1},\ldots,\boldsymbol{u}_{\star,M}]^T$. The set $U_\star$ serves as the input test data set. Based on GP modeling, the prior joint distribution between the training and test data can be expressed as [152]

$$\begin{bmatrix}\boldsymbol{y}\\ \boldsymbol{f}_\star\end{bmatrix} = \mathcal{N}\Big(\begin{bmatrix}\boldsymbol{\mu}_y\\ \boldsymbol{\mu}_{f_\star}\end{bmatrix}, \begin{bmatrix}K(U,U) & K(U_\star,U)\\ K(U,U_\star) & K(U_\star,U_\star)\end{bmatrix}\Big), \tag{6.2}$$

where $\boldsymbol{f}_\star$ denotes $\boldsymbol{f}(U_\star)$ and $[K(U,U_\star)]_{nm}=\kappa(\boldsymbol{u}_n,\boldsymbol{u}_{\star,m})$. The predictive distribution over the test data, using the existing rules for conditioning Gaussian distributions, is expressed as follows

$$\boldsymbol{f}_\star | U, \boldsymbol{y}, U_\star \sim \mathcal{N}(\boldsymbol{\mu}_{f_\star}, \Sigma_{f_\star}), \tag{6.3}$$

where

$$\begin{aligned}\boldsymbol{\mu}_{f_\star} &= \boldsymbol{\mu}_y + K(U_\star,U)\big(K(U,U)+\sigma_n^2 I\big)^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_y)\\ \Sigma_{f_\star} &= K(U_\star,U_\star) - K(U_\star,U)\big(K(U,U)+\sigma_n^2 I\big)^{-1}K(U,U_\star).\end{aligned} \tag{6.4}$$

Therefore, the point estimate for $\boldsymbol{f}(U_\star)$ is the mean $\boldsymbol{\mu}_{f_\star}$ and the amount of uncertainty in the estimation is represented by the variance $\Sigma_{f_\star}$ in (6.4). Design of the covariance function requires incorporating some prior knowledge about the behavior of the PoI as it determines the amount of correlation between any pair of data points [21]. Some of the most widely used covariance functions are squared exponential kernel ($\kappa_{SE}(u,u') = \exp\{-\frac{(u-u')^2}{2l^2}\}$) and rational quadratic ($\kappa_{RQ}(u,u') = (1+\frac{(u-u')^2}{2\alpha l^2})^{-\alpha}$), where l and α are hyperparameters [22]. These kernels fall in the category of stationary covariance functions. Once the structure of the covariance function is selected, the corresponding hyperparameters in the model can be chosen either empirically or using some quantified statistical methods. In the empirical approach, the selection of hyperparameters is usually achieved using the empirical features obtained from the observed data such as the smoothness or periodic behavior of the samples.

Remark 6.1: Although GPs are powerful tools for regression and classification problems, they suffer from high computational complexity as the sample size of the training data set increases. This problem occurs because the estimation of the test data involves inverting the covariance matrix of the training data which grows as more data are collected. The focus of this work is not on the computational complexity of the GPs but rather on providing a general framework for mobile sensor configuration. Regarding the complexity of GPs, there exist some approaches such as the one for truncated covariance matrices in GPs [153], online sparse matrix GPs (OSMGP) algorithm [147], sparse greedy GP (SGGP) approximation method [154], and reduced rank GP (RRGP) [155]. One can use these approaches for the GPR estimation in the first stage of the proposed framework. Furthermore, there exist some studies on estimating the covariance matrix instead of an experimentally designed kernel function. For instance, Xu and Choi provide an approach to estimate and improve the quality of covariance function for anisotropic spatio-temporal GP using mobile sensor networks [31]. The suggested sampling method for such problem is based on minimizing the information-theoretic cost function of the Fisher information [31]. This can also be incorporated in the proposed framework.

6.2.2 Epistemic Utility Framework

Epistemic utility theory provides a framework that makes satisficing decisions. A satisficing decision-making strategy looks for a satisfactory and sufficient solution rather than an optimal solution. This is useful when either the available information is local or insufficient, or the problem is complex enough that the optimal solution is intractable [33, 151]. In epistemic utility theory, satisficing decisions are made via putting more emphasis on avoiding wrong decisions and favoring informationally valuable decisions rather than the more restrictive (and sometimes unachievable) task of finding the best solution. This theory employs two utility functions [33,35,156] and seeks a tradeoff between them. This is different than a more conventional approach with a single utility for which a maximizing point is sought. The first utility function in the epistemic utility characterizes the *truth value* of propositions being evaluated, and the second utility function characterizes the *informational value* of rejecting propositions. The two utilities are established independently. As an example [151], the truth value for rival scientific theories can be assessed by compatibility with observations, while the informational value can be assessed by parsimony or predictive power.

To apply epistemic utility theory, each of these utility functions is equipped with the mathematical structure of a probability (through normalization if necessary), which allows these two utilities to be compared. The truth value utility function, also known as the *credal probability function*, is denoted by $q(\cdot)$. The informational value utility function, also known as the rejectability probability function, is denoted by $m(\cdot)$. The rejectability defines the amount of information the agent gains once some propositions are rejected from the decision space [32]. A parameter denoted as *boldness*, b, is used as a weighting factor on the informational value function. The boldness parameter represents the agent's willingness to reject propositions. Among the set of possible options, those propositions whose truth value does not outweigh the weighted information value of rejection are rejected. That is, a proposition p is not rejected if $q(p) > bm(p)$. By this means, propositions are retained that are "good enough," ensuring adequate performance without the imposition of an overly restrictive unique "optimal" solution.

For a decision space containing all the possible propositions P at some time, the set of options which are informationally valuable and have a probability of being correct is

$$P_l = \{p \in P : q(p) \geq bm(p)\}. \tag{6.5}$$

P_l contains all the surviving, satisficing, proposition(s) and thus it may not be a singleton set. If the cardinality of the set P_l is greater than 1, then the rejectability and credal probability functions of the surviving propositions are renormalized and the test in (6.5) is repeated.

This procedure is referred to as "deliberation", and is repeated until the cardinality of P_l cannot be reduced further [33]. We denote the surviving proposition(s) of the deliberation stage as $P_l^{\star}$. When there is still more than one decision acceptable and a single decision is needed, $P_l^{\star}$ is then subjected to some "tie-breaking". One way of doing this is decision rule [33]

$$\hat{p}_B = \max_{p \in P_{l^{\star}}} \{q(p) - bm(p)\}. \tag{6.6}$$

An example of making satisficing decisions for a difficult problem can be found in [33], where a driver wishes to reach a destination through some road segments with the purpose of finding a good route that simultaneously conserves both fuel cost and distance. Though being provided with the road map, no global information is assumed to be available to the driver on fuel costs (fuel cost may change over time).

6.3 Trajectory Determination of Mobile Sensors

In this section, we propose a general framework to determine the trajectory of mobile sensors in order to explore the important features of the PoI over the region under study. Assume that an initial trajectory of the sensors has already been determined. As may occur in real situations, these initial trajectories may not be necessarily very informative. Once some measurements are collected over the region, the goal becomes using the capability of sensors in follow-on trajectories to both refine the data and explore for interesting regions. Our proposed framework contains two main stages: the prediction stage and the decision stage. A single pass of the sensors only samples a small fraction of the entire region under study, so that decisions about whether to explore other unseen areas must be based on some predictions of the behavior of the interesting phenomenon. Such predictions will be conducive to decide on how to replace the mobile sensors or equivalently to determine the next set of trajectories of the existing sensors.

The prediction stage employs GPR modeling to estimate the PoI in unseen locations. In this setting, the collected data are treated as the training set, where the spatiotemporal location of the measurements determines the input training data and the corresponding

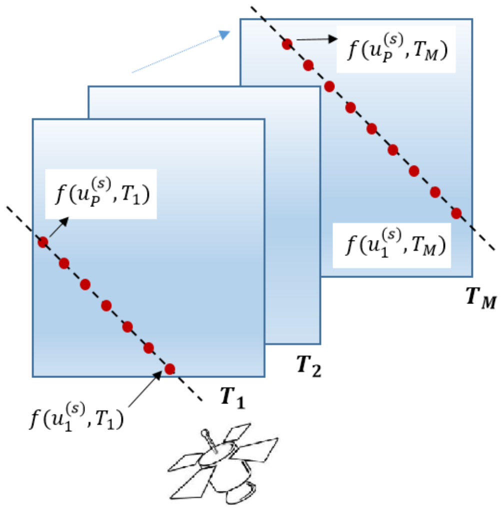


Fig. 6.1: Example showing a satellite (or a mobile sensor) trajectories in the region under study.

measurements are the output training data. The input test data are the other unseen locations over the region under study and the output test data are unknown and are predicted using the GPR. Without loss of generality, we assume that input training data are collected into the set $U_s \in R^{N_1 \times d}$, where $U_s = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N_1}\}$ and d is the dimension of the input data i.e., $\boldsymbol{u}_n \in R^d$. For example, $d = 4$ for the spatiotemporal input data. The output training data, for the scalar output case, are accumulated in $\boldsymbol{y} = [y_1, \ldots, y_{N_1}]^T$, where y_n is the output corresponding to the input $\boldsymbol{u}_n$. The spatiotemporal information of the test data are collected into the set $U_\star \in R^{N_2 \times d}$, where $U_\star = \{\boldsymbol{u}_{\star,1}, \ldots, \boldsymbol{u}_{\star,N_2}\}$. The unknown outputs evaluated at the input test data are defined as $\boldsymbol{f}_\star = [f(\boldsymbol{u}_{\star,1}), \ldots, f(\boldsymbol{u}_{\star,N_2})]^T$. As prior knowledge, we assume that the joint density function between the training and test data is zero-mean Gaussian, meaning that on the average we expect interesting phenomena occur rarely. The GPR model was defined in (6.2). The kernel function for constructing the covariance matrices $K(\cdot,\cdot)$ in (6.2) will be defined later. The predictive posterior distribution over $\boldsymbol{f}_\star$ for the noisy observation case was given in (6.3) and (6.4).

In Fig. 6.1 we show an example of sampling over the region under study. In Fig. 6.1, the dashed lines represent the determined trajectory of a satellite, the red circles denote the locations where the samples are taken, and the rectangular shape is the region under study.

The behavior of the PoI is represented by $f(\cdot)$, and time frame T_m is the mth time the satellite visits the region. Here, we assume that the measurements during the time frame T_m are taken much faster than the changes in the PoI. Also, we assume that the PoI exhibits some sort of smooth behavior in the region under study. Therefore, as the satellite is within a specific time frame T_m, we may expect to see high correlation between the nearby samples. We further assume (here, for simplicity) that the time it takes for the satellite to revisit the region is less than the time for changes to occur in the PoI. Therefore, if it happens to have the same trajectory for time frames T_i and T_j, then high correlation between the collected data at such time frames is expected for the case where T_i is close to T_j.

Under the smoothness assumption for the PoI and in order to account for the correlation that may exist between the nearby measurements, we define the following squared exponential covariance function

$$\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp\{\frac{-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2}{2l_1} + \frac{-\|T_{m_1} - T_{m_2}\|_2^2}{2l_2}\}, \tag{6.7}$$

where l_1 and l_2 are scaling factors, $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ are the coordinates of any two arbitrary locations in the region that is under investigation, and T_{m_1} and T_{m_2} are the time frames at which the measurements $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ are collected, respectively. As the available prior knowledge changes, one may define a different kernel function from (6.7). Suppose that $\mathcal{S} = \{s^{(1)}, s^{(2)}, \ldots, s^{(K)}\}$ is a set of all feasible trajectories of the sensors through the region under study. We denote the locations along the trajectory $s^{(k)}$ at which measurements are collected by the set $U_{s^{(k)}} = \{\boldsymbol{u}^1_{s^{(k)}}, \ldots, \boldsymbol{u}^P_{s^{(k)}}\}$. For the case where the trajectory along $s^{(k)}$ has not taken yet, the locations in the set $U_{s^{(k)}}$ belong to the set of test data i.e., $U_{s^{(k)}} \in U_\star$.

Using the GPR model, we obtain a prediction of the PoI throughout the region and the amount of uncertainty (variance) associated with the predictions. The amount of uncertainty can be evaluated via the kernel function defined in (6.7). We denote $\hat{f}(U^p_{s^{(k)}})$ and $\hat{\Sigma}(U^p_{s^{(k)}})$ as the estimate of the PoI and the associated measure of variance for the pth location along the trajectory $s^{(k)}$, respectively. Then, the goal is to decide on the next trajectory of the sensors based on the available data, the estimated data, and also the amount of uncertainty over the region in order to explore the interesting phenomenon. Although the GPR provides us with an estimate of the behavior of the phenomenon over the whole region, we get large uncertainty at locations where there exist almost no measurements; the farther away the sensors are from the measurements, the higher the variance becomes. Therefore, if we only emphasize refining existing measurements, the sensors lose the inclination to explore. In contrast, if we put more emphasis on the variance, then the sensors are more encouraged to choose the trajectories which are far away from the previous trajectories to fulfill the exploration objective of the mission. In this case, even if interesting phenomena are found by the past trajectories, the sensors are reluctant to pass nearby again. One way of taking into account both data refinement and exploration desires can be achieved by constructing the following optimization problem to decide on the trajectories of the sensors:

$$k^\star = \arg\max_{k} \quad \hat{f}_{ave}(U_{s(k)}) + \lambda \hat{\Sigma}_{ave}(U_{s(k)}), \tag{6.8}$$

where k denotes the kth trajectory from the dictionary of feasible trajectories $\mathcal{S}$. The parameter λ is a tuning parameter that can be set based on the desire to emphasize either interesting phenomenon or the exploration. In (6.8), $\hat{f}_{ave}(U_{s(k)})$ and $\hat{\Sigma}_{ave}(U_{s(k)})$ are defined as

$$\begin{aligned} \hat{f}_{ave}(U_{s(k)}) &= \frac{1}{P}\sum_{p=1}^{P} \hat{f}(\boldsymbol{u}^p_{s(k)}) \\ \hat{\Sigma}_{ave}(U_{s(k)}) &= \frac{1}{P}\sum_{p=1}^{P} \hat{\Sigma}(\boldsymbol{u}^p_{s(k)}). \end{aligned} \tag{6.9}$$

There can be a case to make λ depend on time. For example, one can initially set λ to a large value to favor exploration. Then after some discovery, reducing λ increases the inclination to refine the data on the interesting phenomena that may have been obtained. This approach is applicable to situations where the capability of changing the trajectories of the sensors is limited by the available energy resources. Therefore, we can relate λ to the available energy (e.g., thrust) resources as time elapses. For the case of satellite constellation problem, λ can be related to the required ΔV-budget for the orbit transfer and the available thrust on the satellites.

The difference between the boldness factor b in the epistemic utility defined in Section 6.2.2 and the tuning parameter λ in (6.8) is as follows. The parameter λ in (6.8) only balances between $\hat{f}_{ave}(U_{s^{(k)}})$ and $\hat{\Sigma}_{ave}(U_{s^{(k)}})$ by searching the whole solution space. In contrast, the boldness parameter takes care of such balance by reducing the solution space such that the remaining solutions in the decision space are all informationally valuable. There exist no optimal solution for the optimization problem (6.8) when the PoI changes over the sampling period. Therefore, we need a tool to be able to make value-laden decisions on the trajectory determination of the mobile sensors. For this purpose, we consider the epistemic utility controller discussed in Section 6.2.2 using the outputs of the GPR model. Specifically, we consider the two following cases with their corresponding credal and rejection probability functions. In the first case, the rejectability probability is defined such that it emphasizes rejecting the trajectories from $\mathcal{S}$ that contain lower uncertainty in order to favor the exploration desire. Since the kernel function that we considered for our GPR, (6.7), has a smoothing feature, the farthest the trajectory $s^{(k)}$ becomes from the previous trajectories the higher value the measure of uncertainty $s^{(k)}$ will contain. Below, we define the probability functions constructing the epistemic utility controller for the first case.

$$\begin{aligned} q(U_{s^{(k)}}) &= \hat{f}_n(U_{s^{(k)}}),\ \forall k = 1, 2, ..., K \\ m(U_{s^{(k)}}) &= \frac{1}{\hat{\Sigma}_n(U_{s^{(k)}})},\ \forall k = 1, 2, ..., K, \end{aligned} \tag{6.10}$$

where $\hat{f}_n(U_{s^{(k)}})$ and $\hat{\Sigma}_n(U_{s^{(k)}})$ are defined as

$$\begin{aligned} \hat{f}_n(U_{s^{(k)}}) &= \frac{\hat{f}_{ave}(U_{s^{(k)}})}{\sum_{k=1}^{K} \hat{f}_{ave}(U_{s^{(k)}})},\ \forall k = 1, 2, ..., K \\ \hat{\Sigma}_n(U_{s^{(k)}}) &= \frac{\hat{\Sigma}_{ave}(U_{s^{(k)}})}{\sum_{k=1}^{K} \hat{\Sigma}_{ave}(U_{s^{(k)}})},\ \forall k = 1, 2, ..., K, \end{aligned} \tag{6.11}$$

where $\hat{f}_{ave}(U_{s^{(k)}})$ and $\hat{\Sigma}_{ave}(U_{s^{(k)}})$ were defined in (6.9). The subscript n denotes that the

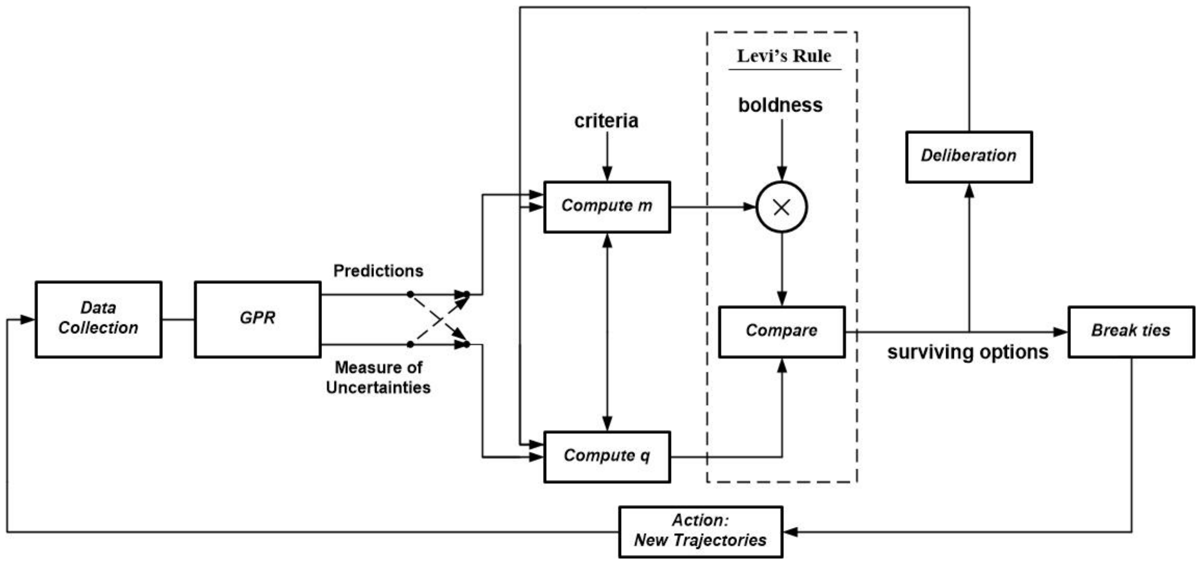


Fig. 6.2: Block diagram of exploration using mobile sensors based on epistemic utility controller.

functions in (6.11) are normalized to act like probabilities. According to (6.10), for the first case we assign credal probabilities to the estimates of each possible trajectory while the measure of uncertainties determine the rejection probabilities. Also, without loss of generality, we assume that the more interesting behavior the phenomenon at location $\boldsymbol{u}^p_{s(k)}$ becomes, the higher value the underlying function $\hat{f}(\boldsymbol{u}^p_{s(k)})$ will possess.

In the second case, we relate the less interesting phenomenon to the rejection probability function. Particularly, the less interesting the predicted phenomenon behaves along the possible trajectory $U_{s(k)}$, the higher the rejection of the corresponding hypothesis becomes. In this case, the credal probability is evaluated via the amount of uncertainty each possible trajectory may possess. Each of the possible trajectories of sensors is considered as a hypothesis. Below, the credal and rejection probabilities for case 2 are represented.

$$\begin{aligned} q(U_{s(k)}) &= \hat{\Sigma}_n(U_{s(k)}),\ \forall k = 1, 2, ..., K \\ m(U_{s(k)}) &= \frac{1}{\hat{f}_n(U_{s(k)})},\ \forall k = 1, 2, ..., K, \end{aligned} \tag{6.12}$$

where $\hat{f}_n(U_{s(k)})$ and $\hat{\Sigma}_n(U_{s(k)})$ were defined in (6.11). Once the probability functions are computed for either of the two above cases, we apply the Levi's rule of epistemic utility. As a result, only those trajectories that satisfy $q(U_{s(k)}) \geq bm(U_{s(k)})$ are surviving hypotheses. The surviving options are defined by the following set

$$U_{srv} = \{u \in U_{\star} \ : \ q(u) \geq bm(u)\}. \tag{6.13}$$

Remark 6.2: In (6.10) and (6.12), we used the normalized version of estimates and the associated variance along each trajectory in order to make $q(\cdot)$ and $m(\cdot)$ follow the "sum to one" property of probability. However, one can remove the normalization step and define the boldness factor of $b \geq 0$ instead of $b \in [0, 1]$.

After applying rule (6.13), the number of surviving options may not be necessarily unique i.e., the cardinality of U_{srv} could be greater than one. Each of the elements of U_{srv} is a satisficing hypothesis meaning that is both likely to be correct and possesses high informational value. In order to take an action we seek to accept only one trajectory (one hypothesis). Reducing the number of elements in the set U_{srv} is accomplished in the deliberation stage described in Section 6.2.2, results in reducing the number of surviving hypotheses. Once the cardinality of the set U_{srv} reduces to a reasonable number, the tie breaking stage comes into play to force the set U_{srv} to a unique element in order to take an action. For this purpose, one can apply the approach (6.6) as described in [33], which selects one hypothesis out of the survived hypotheses as a next trajectory. We refer to this trajectory as $U_{s(k_{\star})}$. Finally, after measurement, the data obtained from $U_{s(k_{\star})}$ is added to the training set U, and the whole process starts again. Fig. 6.2 illustrates the block diagram of the proposed framework.

In order to restrain the increase in the amount of training data fed to the GPR (to reduce complexity), in the data collection block of Fig. 6.2, we only retain the measurements obtained from the last M visited trajectories of sensors. Once a new trajectory is determined and the corresponding measurements are collected, the new information is added to the training set and the oldest set of measurements are discarded. The reason is due to the assumption that the oldest set of measurements may have very low correlation with the new

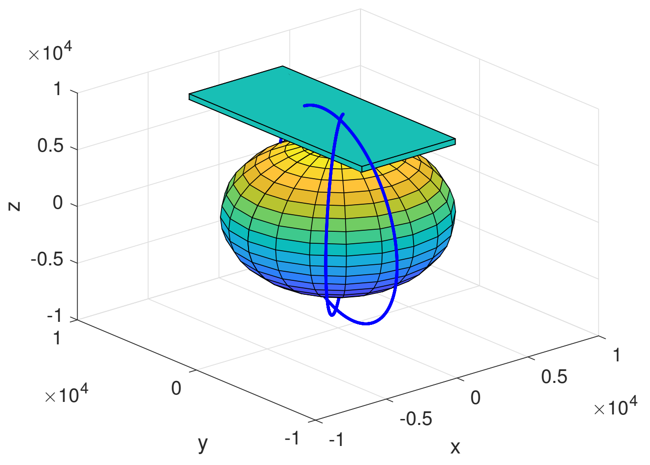


Fig. 6.3: The satellites orbits and the region under study.

measurements and the PoI may have been changed. This avoids dealing with the inverse of a big covariance matrix of the training data as we continue collecting new measurements.

6.4 Simulation Results

We demonstrate how the proposed framework is applied via a problem of a constellation of two satellites at the low Earth orbit (LEO). In this problem, we are incapable of performing random sampling over the region. Instead, we are restricted to follow specific trajectories once the orbits of the satellites (or trajectories of the mobile sensors) are determined. Initially, the satellites move in predetermined orbital planes over the region of interest. Assume that the PoI remains essentially unchanged during the sampling period. Fig. 6.3 illustrates an example including the orbital planes of satellites, where the rectangular slab indicates the region under study. In Fig. 6.3, the constellation is defined based on Keplerian orbital elements with semi-major axis $a = \{7700, 8500\}$ (km), eccentricity $e = 0$, inclination of $i = \{\pi/2, \pi/2\}$ (rad), and the right ascension of ascending node (R.A.A.N.) $\Omega = \{\pi/6, \pi/3\}$ (rad). Since the PoI is assumed to be unchanged during the sampling period, the true (mean) anomaly θ and the argument of perigee are dismissed. The true behavior of the region of under exploration is shown via the image in Fig. 6.4, which is from [144] illustrating a sense of the total electron content (TEC) in the ionosphere.

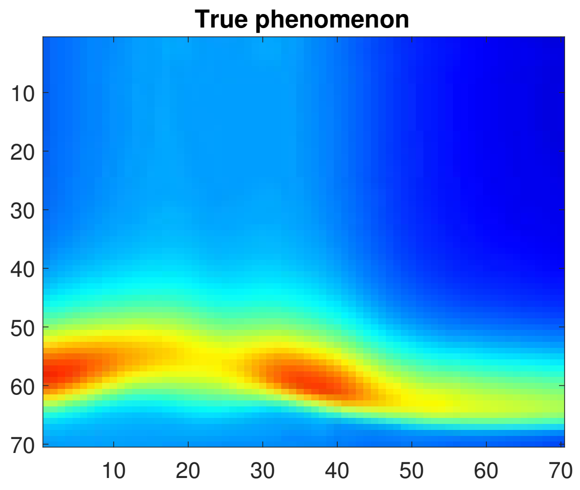


Fig. 6.4: True behavior of the PoI. This is an image, representing the PoI in the region of under study for some range of latitude and longitude.

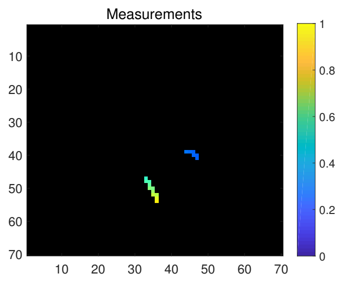
In Fig. 6.4, the most and the least interesting phenomenon corresponding to the PoI are shown with red and blue colors, respectively. We further assume that the PoI has the same profile as shown in Fig. 6.4 along the z-axis of the rectangular region shown in Fig. 6.3. This image can be thought of as a discretized 2-D version of the region of interest defined by the pixel values, where the y- and x-axis correspond to the longitude and latitude of the region of interest. In the simulations, it is assumed that it is possible to get direct measurements about the PoI along the current trajectories of satellites. Since the PoI is assumed to be unchanged during the sampling period, the kernel function (6.7) simplifies into

$$\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp\{\frac{-\|\boldsymbol{u}_i - \boldsymbol{u}_j)\|_2}{2l}\}, \tag{6.14}$$

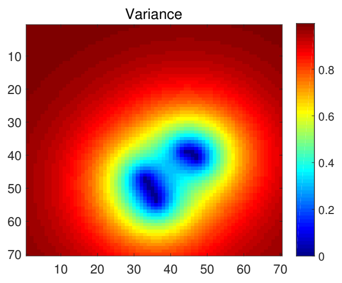
where we set $l = 10$ and $\boldsymbol{u}_i$ is defined by the pixel location of the image shown in Fig. 6.4.

From the initial trajectories, shown in Fig. 6.3, the corresponding set of measurements of the region under study is shown in Fig. 6.5(a). From the initial measurements, the GPR model defined in (6.4) and (6.14) is applied to measure the uncertainty and the estimate of the PoI throughout the region of interest. The results are illustrated in Fig. 6.5(b) and Fig. 6.5(c).

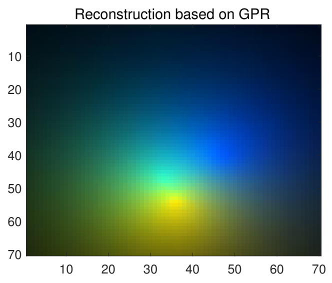
The second stage of the proposed framework is applied to decide on the next trajectories



(a)



(b)



(c)

Fig. 6.5: From (a)–(c): Initial measurements, measure of variance, and reconstruction of PoI based on the initial observation.

of the satellites. Although the initial orbital planes where constructed directly from the Keplerian orbital planes, below we assume (for simplicity in this example) that each possible trajectory can pass through the region under study such that it fully measures either a column or a row of the image shown in Fig. 6.4. These trajectories can be generated from orbital planes with high eccentricity.

The epistemic utility is set with the agent's index of boldness $b = 1$ and the credal and rejection probability functions of (6.12). In other words, the rejection probability function is defined such that it tends to remove the trajectories corresponding to uninteresting phenomenon. The credal probability function is set to encourage trajectories willing to visit regions with higher uncertainty. Fig. 6.6 illustrates some of the results for the measurements, selected trajectories, measure of uncertainty, and the reconstruction. According to Fig. 6.6, the selected trajectories are not willing to revisit the vicinity of the areas where interesting phenomenon seems not to exist. Simultaneously, the decision maker does not allow accepting a trajectory at the very vicinity of the already chosen trajectories even if some interesting phenomenon has been detected at their neighborhood. This is shown by the measure of variance in the third row of Fig. 6.6. More specifically, very few trajectories are selected at the uninteresting subregions, and such trajectories demonstrate some sort of uniform sampling. In contrast, more trajectories are chosen at the interesting subregions and yet these trajectories do not tend to be fully next to each other.

In Fig. 6.7, we compare the performance of the proposed framework for this example for

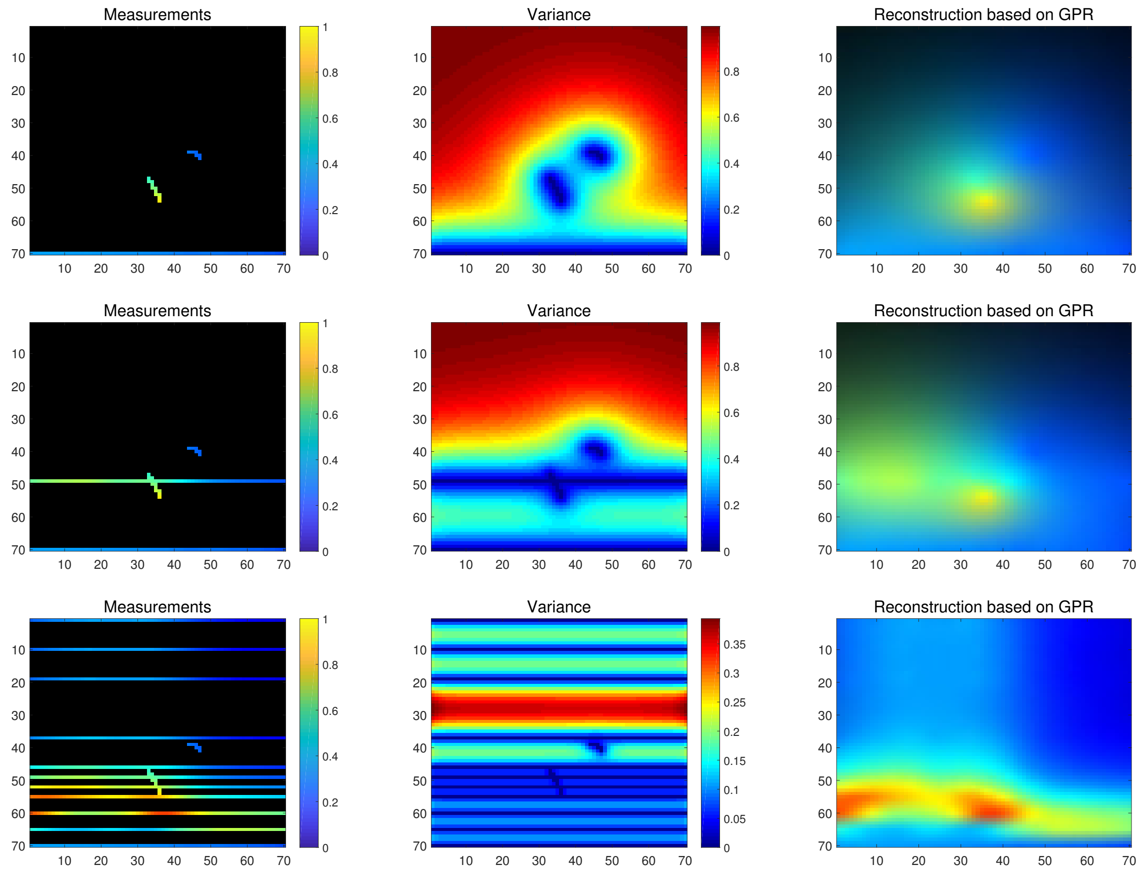
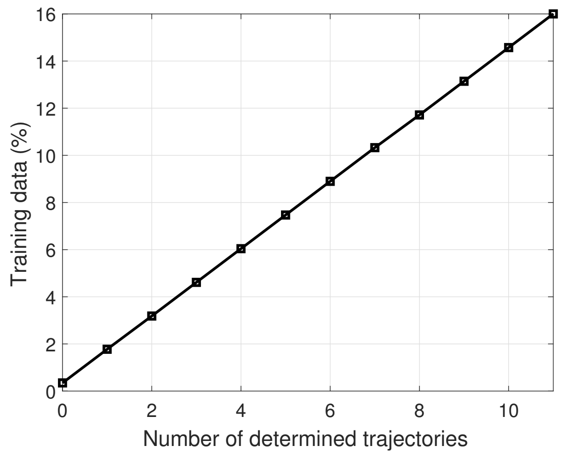


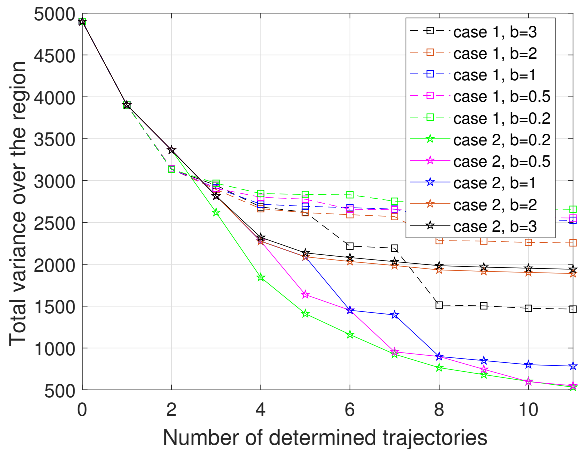
Fig. 6.6: From left to right, the measurements, measure of uncertainty, and the reconstruction of phenomenon based on the initial and one, two, and eleven successive trajectories are shown for $b = 1$ when using (6.12).

cases 1 and 2 defined in (6.10) and (6.12), respectively. In Fig. 6.7(a) we show the percentage of the measurements (training data) with respect to the total number of possible data if we were able to cover all the region under study. Even after accumulating the measurements obtained from the initial and the eleven successive trajectories, we still cover around 16% of the complete data over the region. In Fig. 6.7(b), the total variance over the region *vs.* the increase in the number of trajectories is illustrated. Here, the variance of visited locations are set to zero and the variance of unvisited locations are measured via the kernel function defined in (6.14). Then, we sum over all the variances associated with the pixels. Finally, in Fig. 6.7(c) the peak-SNR evaluation between the true and the reconstructed image is demonstrated. According to Fig. 6.7(b), the increase of boldness factor for case 1 results in the decrease of the overall uncertainty in the region under study. This is because the rejection probability function is corresponded to the inverse of overall variance along the possible trajectories. Therefore, the increase of the boldness factor promotes selecting the trajectories which are estimated to have higher uncertainty. In contrast, case 2 shows a different behavior and that is the increase of the boldness factor forces to discard the trajectories that are believed to be uninteresting. The PSNR evaluation depends not only on how to construct the probabilities in the epistemic utility but also on the true behavior of the PoI. Since the PoI has a smooth behavior and we have already taken this fact into account, the PSNR increases as we increase the boldness factor for case 1. For case 2, the reduction of the boldness factor usually provides better performance in terms of PSNR but it also depends on where we sample. For example, setting $b = 0.2$ does not show better performance compared to $b = 0.5$ and $b = 1$. The reason is because the controller decided on a trajectory that is in the interesting region but close to the edge of such phenomenon. Since the kernel function in the GPR stage assumes the smooth behavior, it expands the interesting phenomenon at the neighborhood of the selected trajectory and thus the estimated PoI exceeds the edges of the true PoI. This yields to the decrease in the PSNR.

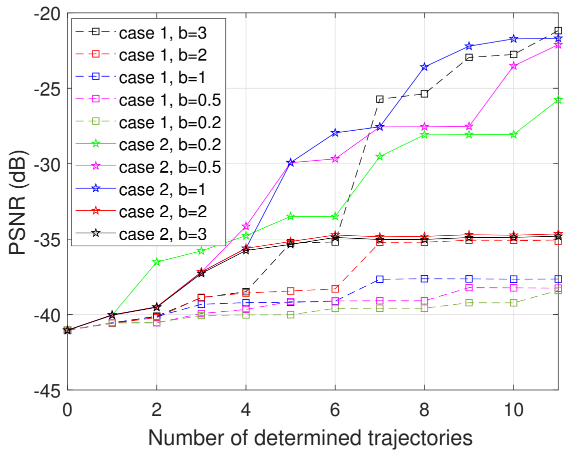
Finally, we consider two more cases to highlight the advantage of the proposed framework. In cases 3 and 4, the trajectories are selected only based on either the variance or the



(a)

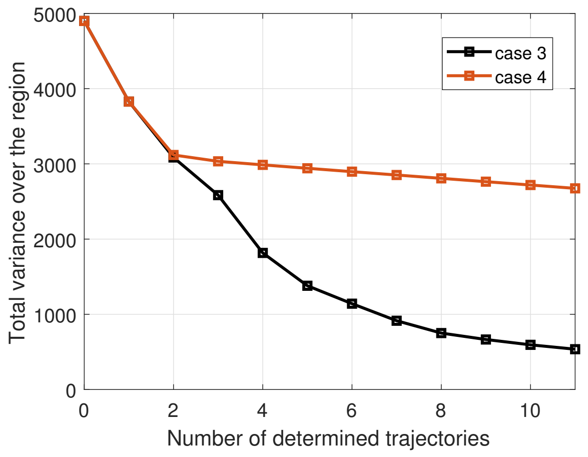


(b)



(c)

Fig. 6.7: Comparison of the performance for cases 1 and 2 with $b = 0.2, 0.5, 1, 2$, and $b = 3$.



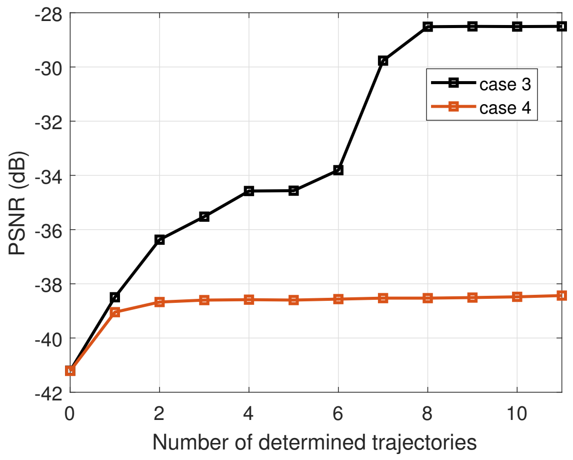


Fig. 6.8: Comparison of the performance for cases 3 and 4.

interesting phenomenon, respectively. Fig. 6.8 and Fig. 6.9 illustrate the obtained results.

As expected, Fig. 6.8 shows that case 3 outperforms case 4 in terms of reducing the overall uncertainty in the region. The reason is because case 3 only favors the trajectories with highest variance. Also, due to the smoothness of the PoI, still case 3 shows higher PSNR than case 4. Comparing the total variance and the PSNR of all cases 1–4, it is clear that cases 1 and 2 result in better performance in reducing the overall variance and the increase of PSNR, collectively. The trajectory selection of cases 3 and 4 is also shown in Fig. 6.9 for eleven successive set of measurements added to the initial measurements. Comparing cases 3 and 4, it is obvious that they both tend to uniformly sample the region. However, the difference is that case 3 uniformly samples the whole region, while case 4 uniformly samples

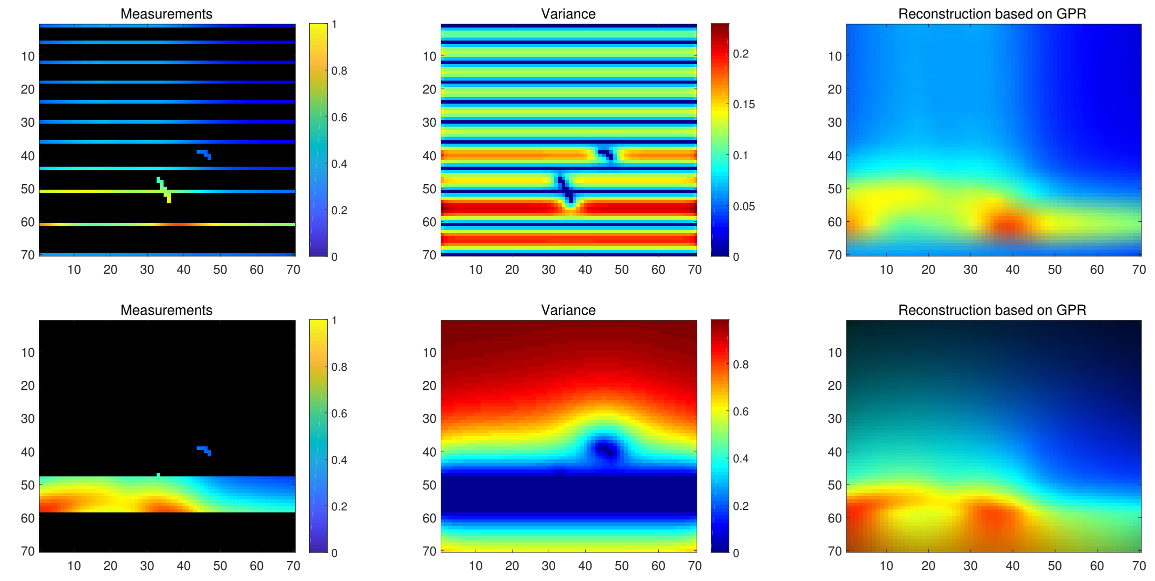


Fig. 6.9: The rows from top to bottom illustrate the measurements, measure of uncertainty, and the reconstruction of phenomenon based on the initial and eleven successive trajectories are shown for cases 3 and 4, respectively.

the vicinity of the interesting phenomenon once some is observed. Therefore, if there was any other interesting subregion, we would not have a chance to observe it via case 4 for low number of trajectories. In contrast, cases 1 and 2 of our proposed framework apply an adaptive sampling structure to balance between the reduction of the amount of uncertainty in the region and refining data in the vicinity of interesting phenomenon.

6.5 Summary

We provided a new framework for placing mobile sensors using Gaussian process regression and epistemic utility controller. The proposed framework considers both desires of exploration and data refinement once some interesting phenomenon is observed. This approach is useful for real world problems where very little local information is available and no optimal solution exists for the sensor placement. The proposed framework can also handle the constraints that may exist on the budget for sensor placement.

6.6 Appendix

In this appendix, we provide some more detail on the GPR modeling and the associated proofs for the predictions using GRP for both non-parametric GP and semi-parametric GP modeling. Although the derived predictions can be found in references such as [22, 152], the derivation of the equations were not included in these references.

6.6.1 Appendix (a): Prediction Using Non-Parametric GPR for Noise-Free Observations

Suppose we observe a training data set $D = \left\{(\mathbf{x}_i, f_i), i = 1, ..., N\right\}$ and $f_i = f(\mathbf{x}_i)$, where $\mathbf{x}_i$ and f_i denote the $i$the set of inputs and the corresponding output, respectively. Given a test set X_* of size $N_* \times D$, the goal is to predict outputs $\mathbf{f}_*$. By definition of the GP, the joint distribution has the following form

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} K & K_* \\ K_*^T & K_{**} \end{bmatrix}\right),$$

where $K = \kappa(X, X)$ is $N \times N$, $K_* = \kappa(X, X_*)$ is $N \times N_*$, and $K_{**} = \kappa(X_*, X_*)$ is $N_* \times N_*$.

Then, the posterior distribution over $\mathbf{f}_*$ becomes [22, 152]

$$p(\mathbf{f}_* | X_*, X, \mathbf{f}) = \mathcal{N}(\mathbf{f}_* | \boldsymbol{\mu}_{f_*}, \Sigma_{f_*}), \tag{6.15}$$

where

$$\begin{cases} \boldsymbol{\mu}_{f_*} = \boldsymbol{\mu}_* + K_*^T K^{-1}(\mathbf{f} - \boldsymbol{\mu}(X)), \quad \boldsymbol{\mu}_* = \boldsymbol{\mu}(X_*) \\ \Sigma_{f_*} = K_{**} - K_*^T K^{-1} K_*. \end{cases}$$

Below, the details on how to derive (6.15) are provided.

Remark 6.3: Notice that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} E & F \\ G & H \end{bmatrix},$$

where

$$\begin{cases} E = A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} \\ F = -A^{-1}B(D - CA^{-1}B)^{-1} \\ G = -(D - CA^{-1}B)^{-1}CA^{-1} \\ H = (D - CA^{-1}B)^{-1}. \end{cases}$$

Using Remark 6.3, the posterior distribution over $\mathbf{f}_*$ is proportional to

$$\log\{p(\mathbf{f}_*|X_*, X, \mathbf{y})\} \propto -\Big(\begin{bmatrix} (\mathbf{f} - \boldsymbol{\mu})^T & (\mathbf{f}_* - \boldsymbol{\mu}_*)^T \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} \begin{bmatrix} \mathbf{f} - \boldsymbol{\mu} \\ \mathbf{f}_* - \boldsymbol{\mu}_* \end{bmatrix}\Big),$$

where

$$\begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} K & K_* \\ K_*^T & K_{**} \end{bmatrix}^{-1}.$$

Therefore

$$\begin{aligned}
\log\{p(\mathbf{f}_*|X_*,X,\mathbf{y})\} \propto \\
&-\Big((\mathbf{f}_*-\boldsymbol{\mu}_*)^T H(\mathbf{f}_*-\boldsymbol{\mu}_*) + (\mathbf{f}_*-\boldsymbol{\mu}_*)^T\big(G(\mathbf{f}-\boldsymbol{\mu}) + F^T(\mathbf{f}-\boldsymbol{\mu})\big)\Big) \\
&\propto -\Big((\mathbf{f}_*-\boldsymbol{\mu}_*)^T H(\mathbf{f}_*-\boldsymbol{\mu}_*) + (\mathbf{f}_*-\boldsymbol{\mu}_*)^T\big((F^T+G)(\mathbf{f}-\boldsymbol{\mu})\big)\Big) \\
&\propto -\Big(\mathbf{f}_*^T H\mathbf{f}_* + \mathbf{f}_*^T\big(-2H\boldsymbol{\mu}_* + (F^T+G)(\mathbf{f}-\boldsymbol{\mu})\big)\Big) \\
&\propto -\Big(\Big(\mathbf{f}_* - H^{-1}\big(H\boldsymbol{\mu}_* - \frac{1}{2}(F^T+G)(\mathbf{f}-\boldsymbol{\mu})\big)\Big)^T H\Big(\star\Big)\Big).
\end{aligned} \tag{6.16}$$

According to (6.16), the covariance Σ_{f_*} becomes

$$\Sigma_{f_*} = H^{-1}, \quad H^{-1} = K_{**} - K_*^T K^{-1} K_*$$

and the mean $\boldsymbol{\mu}_{f_*}$ can be found from

$$\begin{aligned}
\boldsymbol{\mu}_{f_*} &= \boldsymbol{\mu}_* - \frac{1}{2}H^{-1}(-H^T K_*^T K^{-T} - HK_*^T K^{-1})(\mathbf{f}-\boldsymbol{\mu}) \\
&= \boldsymbol{\mu}_* + \frac{1}{2}(H^{-1}H^T K_*^T K^{-T} + K_*^T K^{-1})(\mathbf{f}-\boldsymbol{\mu}) \\
&= \boldsymbol{\mu}_* + \frac{1}{2}(K_*^T K^{-T} + K_*^T K^{-1})(\mathbf{f}-\boldsymbol{\mu}).
\end{aligned}$$

After some simplification, the mean $\boldsymbol{\mu}_{f_*}$ becomes

$$\boldsymbol{\mu}_{f_*} = \boldsymbol{\mu}_* + K_*^T K^{-1}(\mathbf{f}-\boldsymbol{\mu}).$$

Therefore, in summary we have

$$\begin{aligned}
&p(\mathbf{f}_*|X_*,X,\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_{f_*}, \Sigma_{f_*}), \\
&\begin{cases}
\boldsymbol{\mu}_{f_*} = \boldsymbol{\mu}_* + K_*^T K^{-1}(\mathbf{f}-\boldsymbol{\mu}) \\
\Sigma_{f_*} = K_{**} - K_*^T K^{-1} K_*.
\end{cases}
\end{aligned} \tag{6.17}$$

6.6.2 Appendix (b): Prediction Using Semi-Parametric GPR for Noisy Observations

Consider the following model

$$f(\mathbf{x}) = \boldsymbol{\beta}^T \Phi(\mathbf{x}) + r(\mathbf{x}),$$

where the linear model $\boldsymbol{\beta}^T \Phi(\mathbf{x})$ is used for the mean and GP modeling is considered over $r(\mathbf{x})$ for the residual of the process defined as

$$r(\mathbf{x}) \sim \mathcal{GP}\big(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}')\big).$$

In fact, semi-parametric modeling combines a parametric model of the mean and non-parametric model for the residual of the process. In this case, one can define the following prior for $\boldsymbol{\beta}$

$$\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{b}, B). \tag{6.18}$$

Then, the posterior distribution over $\mathbf{f}_*$ for GPR with semi-parametric model becomes

$$\begin{aligned} & p(\mathbf{f}_* | X_*, X, \mathbf{y}) = \mathcal{N}\big(\bar{\mathbf{f}}_*, \mathrm{Cov}(\mathbf{f}_*)\big), \\ & \begin{cases} \bar{\mathbf{f}}_* = \Phi_*^T \bar{\boldsymbol{\beta}} + K_*^T K_y^{-1} (\mathbf{y} - \Phi^T \bar{\boldsymbol{\beta}}) \\ \bar{\boldsymbol{\beta}} = (\Phi K_y^{-1} \Phi^T + B^{-1})^{-1} (\Phi K_y^{-1} \mathbf{y} + B^{-1} \mathbf{b}) \\ \mathrm{Cov}(\mathbf{f}_*) = K_{**} - K_*^T K_y^{-1} K_* + R^T (B^{-1} + \Phi K_y^{-1} \Phi^T)^{-1} R \\ R = \Phi_* - \Phi K_y^{-1} K_*. \end{cases} \end{aligned} \tag{6.19}$$

Below the details on how to derive (6.19) is represented. The set of priors considered here are

$$\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{b}, B), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_N^2 I),$$

where the noise is denoted by $\boldsymbol{\epsilon}$ as is modeled as a zero-mean Gaussian distribution. Therefore, the posterior distribution over $\boldsymbol{\beta}$ is proportional to

$$p(\boldsymbol{\beta}|X,\mathbf{y}) \propto p(\mathbf{y}|X,\boldsymbol{\beta})p(\boldsymbol{\beta}|\mathbf{b},B),$$

and by taking the logarithm from the above equation, we have

$$\begin{aligned}\log\{p(\boldsymbol{\beta}|X,\mathbf{y})\} &\propto -\big((\mathbf{y}-\Phi^T\boldsymbol{\beta})^T K_y^{-1}(\star) + (\boldsymbol{\beta}-\mathbf{b})^T B^{-1}(\star)\big)\\ &\propto -\big(\boldsymbol{\beta}^T(\Phi K_y^{-1}\Phi^T + B^{-1})\boldsymbol{\beta} - 2\boldsymbol{\beta}^T(\Phi K_y^{-1}\mathbf{y} + B^{-1}\mathbf{b})\big).\end{aligned}$$

Therefore, the posterior distribution over $\boldsymbol{\beta}$ becomes

$$p(\boldsymbol{\beta}|X,\mathbf{y}) = \mathcal{N}(\bar{\boldsymbol{\beta}},\hat{\beta}),$$

where

$$\begin{cases}\bar{\boldsymbol{\beta}} = (\Phi K_y^{-1}\Phi^T + B^{-1})^{-1}(\Phi K_y^{-1}\mathbf{y} + B^{-1}\mathbf{b}) = \hat{\beta}(\Phi K_y^{-1}\mathbf{y} + B^{-1}\mathbf{b})\\ \hat{\beta} = (\Phi K_y^{-1}\Phi^T + B^{-1})^{-1}\end{cases} \tag{6.20}$$

yielding to

$$p(\mathbf{f}_*|X_*,X,\mathbf{y}) = \mathcal{N}(\mathbf{f}_*;\boldsymbol{\mu}_*,\Sigma_*),$$
$$\begin{cases}\boldsymbol{\mu}_* = \boldsymbol{\mu}(X_*) + K_*^T K_y^{-1}\big(\mathbf{y} - \boldsymbol{\mu}(X)\big)\\ \Sigma_* = K_{**} - K_*^T K_y^{-1} K_*\end{cases}$$

Therefore,

$$\begin{cases}\boldsymbol{\mu}_* = \Phi_*^T\boldsymbol{\beta} + K_*^T K_y^{-1}(\mathbf{y} - \Phi^T\boldsymbol{\beta})\\ \Sigma_* = K_{**} - K_*^T K_y^{-1} K_*\end{cases} \tag{6.21}$$

Since the above set of equations are dependent on the parameter $\boldsymbol{\beta}$, we integrate out $\boldsymbol{\beta}$ in order to have a non-parametric model, resulting in

$$
\begin{aligned}
p(\mathbf{f}_*|X_*, X, \mathbf{y}) &= \int_{\boldsymbol{\beta}} p(\mathbf{f}_*|X_*, X, \mathbf{y}, \boldsymbol{\beta}) p(\boldsymbol{\beta}|X, \mathbf{y}, X_*, \mathbf{y}_*) d\boldsymbol{\beta} = \int_{\boldsymbol{\beta}} p(\mathbf{f}_*|X_*, X, \mathbf{y}, \boldsymbol{\beta}) p(\boldsymbol{\beta}|X, \mathbf{y}) d\boldsymbol{\beta} \\
&= \int \exp\{-\big((\mathbf{f}_* - \boldsymbol{\mu}_*)^T \Sigma_*^{-1}(\star) + (\boldsymbol{\beta} - \bar{\boldsymbol{\beta}})^T \hat{\beta}^{-1}(\star)\big)\} d\boldsymbol{\beta} \\
&= \int \exp\Big\{ - \Big(\big(\mathbf{f}_* - \Phi_*^T \boldsymbol{\beta} - K_*^T K_y^{-1}(\mathbf{y} - \Phi^T \boldsymbol{\beta})\big)^T \Sigma_*^{-1}\big(\star\big) + (\boldsymbol{\beta} - \bar{\boldsymbol{\beta}})^T \hat{\beta}^{-1}(\star)\Big)\Big\} d\boldsymbol{\beta} \\
&= \int \exp\{-\Big(\big((\Phi_*^T - K_*^T K_y^{-1} \Phi^T)\boldsymbol{\beta} - (K_*^T K_y^{-1} \mathbf{y} - \mathbf{f}_*)\big)^T \Sigma_*^{-1}\big(\star\big) + (\boldsymbol{\beta} - \bar{\boldsymbol{\beta}})^T \hat{\beta}^{-1}(\star)\Big)\} d\boldsymbol{\beta}
\end{aligned}
$$

$$
\begin{aligned}
&p(\mathbf{f}_*|X_*, X, \mathbf{y}) = \\
&\Big[\int e^{\Big\{-\Big(\boldsymbol{\beta}^T \big((\Phi_* - \Phi K_y^{-1} K_*)\Sigma_*^{-1}(\star)^T + \hat{\beta}^{-1}\big)\boldsymbol{\beta} - 2\boldsymbol{\beta}^T \big((\Phi_* - \Phi K_y^{-1} K_*)\Sigma_*^{-1}(K_*^T K_y^{-1}\mathbf{y} - f_*\big) + \hat{\beta}^{-1}\bar{\boldsymbol{\beta}})\Big)\Big\}} d\boldsymbol{\beta}\Big] \times \\
&\exp\Big\{ - \big((K_*^T K_y^{-1}\mathbf{y} - \mathbf{f}_*)^T \Sigma_*^{-1}(\star) + \bar{\boldsymbol{\beta}}^T \hat{\beta}^{-1} \bar{\boldsymbol{\beta}}\big)\Big\}
\end{aligned}
$$

Let's define

$$
R = \Phi_* - \Phi K_y^{-1} K_* \tag{6.22}
$$

Therefore,

$$
\begin{aligned}
&p(\mathbf{f}_*|X_*, X, \mathbf{y}) = \\
&\Big[\int \exp\{-\Big(\boldsymbol{\beta}^T (R\Sigma_*^{-1} R^T + \hat{\beta}^{-1})\boldsymbol{\beta} - 2\boldsymbol{\beta}^T \big(R\Sigma_*^{-1}(K_*^T K_y^{-1}\mathbf{y} - \mathbf{f}_*) + \hat{\beta}^{-1}\bar{\boldsymbol{\beta}}\big)\Big)\} d\boldsymbol{\beta}\Big] \times \\
&\qquad \exp\{-\big((K_*^T K_y^{-1}\mathbf{y} - \mathbf{f}_*)^T \Sigma_*^{-1}(\star) + \bar{\boldsymbol{\beta}}^T \hat{\beta}^{-1} \bar{\boldsymbol{\beta}}\big)\} \\
=&\Big[\int e^{\{-\big(\boldsymbol{\beta} - (R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}(R\Sigma_*^{-1}(K_*^T K_y^{-1}\mathbf{y} - \mathbf{f}_*) + \hat{\beta}^{-1}\bar{\boldsymbol{\beta}})\big)^T (R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})\big(\star\big)\}} d\boldsymbol{\beta}\Big] \ldots \\
&e^{\{\big(R\Sigma_*^{-1}(K_*^T K_y^{-1}\mathbf{y} - \mathbf{f}_*) + \hat{\beta}^{-1}\bar{\boldsymbol{\beta}})\big)^T (R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}\big(\star\big) - \big((\mathbf{f}_* - K_*^T K_y^{-1}\mathbf{y})^T \Sigma_*^{-1}(\star) + \bar{\boldsymbol{\beta}}^T \hat{\beta}^{-1}\bar{\boldsymbol{\beta}}\big)\}}.
\end{aligned}
$$

By taking logarithm of the above equation, we then have

$$\log\{p(\mathbf{f}_*|X_*, X, \mathbf{y})\} \propto$$
$$-\Big(\mathbf{f}_*^T\Sigma_*^{-1}\mathbf{f}_* - 2\mathbf{f}_*^T(\Sigma_*^{-1}K_*^TK_y^{-1}\mathbf{y}) - \big(R\Sigma_*^{-1}\mathbf{f}_* - (R\Sigma_*^{-1}K_*^TK_y^{-1}\mathbf{y} - \hat{\beta}^{-1}\bar{\boldsymbol{\beta}})\big)^T \times$$
$$(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}\big(\star\big)\Big)$$

$$\propto -\Big(\mathbf{f}_*^T\big(\Sigma_*^{-1} - \Sigma_*^{-1}R^T(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}R\Sigma_*^{-1}\big)\mathbf{f}_*$$
$$- 2\mathbf{f}_*^T\big(\Sigma_*^{-1}K_*^TK_y^{-1}\mathbf{y} - \Sigma_*^{-1}R^T(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}(R\Sigma_*^{-1}K_*^TK_y^{-1}\mathbf{y} - \hat{\beta}^{-1}\bar{\boldsymbol{\beta}})\big)\Big)$$

$$\log\{p(\mathbf{f}_*|X_*, X, \mathbf{y})\} \propto -\Big(\mathbf{f}_*^T\big(\Sigma_*^{-1} - \Sigma_*^{-1}R^T(\hat{\beta}^{-1} + R\Sigma_*^{-1}R^T)^{-1}R\Sigma_*^{-1}\big)\mathbf{f}_*$$
$$- 2f_*^T\big(\Sigma_*^{-1}K_*^TK_y^{-1}\mathbf{y} - \Sigma_*^{-1}R^T(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}(R\Sigma_*^{-1}K_*^TK_y^{-1}\mathbf{y} - \hat{\beta}^{-1}\bar{\boldsymbol{\beta}})\big)\Big).$$

Therefore, the covariance over the posterior distribution on $\mathbf{f}_*$ becomes

$$\Sigma_{f_*} = \Sigma_*^{-1} - \Sigma_*^{-1}R^T(\hat{\beta}^{-1} + R\Sigma_*^{-1}R^T)^{-1}R\Sigma_*^{-1}$$

$$\Sigma_{f_*} = \Sigma_* + R^T\hat{\beta}R, \tag{6.23}$$

where
$$\begin{cases} \hat{\beta} & = (\Phi K_y^{-1}\Phi^T + B^{-1})^{-1} \\ R & = \Phi_* - \Phi K_y^{-1}K_* \\ \Sigma_* & = K_{**} - K_*^TK_y^{-1}K_* \end{cases}$$

or equivalently,

$$\Sigma_{f_*} = K_{**} - K_*^TK_y^{-1}K_* + R^T(\Phi K_y^{-1}\Phi^T + B^{-1})^{-1}R.$$

Below, we cmputing and simplify the expected value of $\mathbf{f}_*$.

$$\boldsymbol{\mu}_{\mathbf{f}_*} = \Sigma_{f_*}\big(\Sigma_*^{-1}K_*^T K_y^{-1}\mathbf{y} - \Sigma_*^{-1}R^T(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}(R\Sigma_*^{-1}K_*^T K_y^{-1}\mathbf{y} - \hat{\beta}^{-1}\bar{\boldsymbol{\beta}})\big),$$

where

$$\Sigma_{f_*} = (\Sigma_* + R^T\hat{\beta}R).$$

Therefore,

$$\boldsymbol{\mu}_{\mathbf{f}_*} = (\Sigma_* + R^T\hat{\beta}R)\big(\Sigma_*^{-1}K_*^T K_y^{-1}\mathbf{y} - \Sigma_*^{-1}R^T(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}(R\Sigma_*^{-1}K_*^T K_y^{-1}\mathbf{y} - \hat{\beta}^{-1}\bar{\boldsymbol{\beta}})\big)$$

or equivalently

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{f}_*} = (\Sigma_* + R^T\hat{\beta}R)\big((\Sigma_*^{-1} - \Sigma_*^{-1}R^T(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}\times \\ R\Sigma_*^{-1}K_*^T K_y^{-1}\mathbf{y} + \Sigma_*^{-1}R^T(R\Sigma_*^{-1}R^T\hat{\beta}^{-1})\hat{\beta}^{-1}\bar{\boldsymbol{\beta}}\big)\end{aligned} \tag{6.24}$$

Notice that

$$(\Sigma_* + R^T\hat{\beta}R)^{-1} = (\Sigma_*^{-1} - \Sigma_*^{-1}R^T(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}R\Sigma_*^{-1}.$$

Hence

$$\boldsymbol{\mu}_{\mathbf{f}_*} = (\Sigma_* + R^T\hat{\beta}R)\big((\Sigma_* + R^T\hat{\beta}R)^{-1}K_*^T K_y^{-1}\mathbf{y} + \Sigma_*^{-1}R^T(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1}\hat{\beta}^{-1}\bar{\boldsymbol{\beta}}\big)$$

Also,

$$(R\Sigma_*^{-1}R^T + \hat{\beta}^{-1})^{-1} = \hat{\beta} - \hat{\beta}R(\Sigma_* + R^T\hat{\beta}R)^{-1}R^T\hat{\beta}$$

Thus

$$\boldsymbol{\mu}_{\mathbf{f}_*} = K_*^T K_y^{-1}\mathbf{y} + (\Sigma_* + R^T\hat{\beta}R)\Sigma_*^{-1}R^T(I - \hat{\beta}R(\Sigma_* + R^T\hat{\beta}R)^{-1}R^T)\bar{\boldsymbol{\beta}}$$

$$\boldsymbol{\mu}_{\mathbf{f}_*} = K_*^T K_y^{-1}\mathbf{y} + (\Sigma_* + R^T\hat{\beta}R)\Sigma_*^{-1}R^T\big(I - \hat{\beta}R(I + \Sigma_*^{-1}R^T\hat{\beta}R)^{-1}\Sigma_*^{-1}R^T\big)\bar{\boldsymbol{\beta}}$$

By using matrix inversion lemma, we then have

$$I - \hat{\beta}R(I + \Sigma_*^{-1}R^T\hat{\beta}R)^{-1}\Sigma_*^{-1}R^T = I - (I + \hat{\beta}R\Sigma_*^{-1}R^T)^{-1}\hat{\beta}R\Sigma_*^{-1}R^T$$

Hence

$$\boldsymbol{\mu}_{\mathbf{f}_*} = K_*^T K_{y*}\mathbf{y} + (\Sigma_* + R^T\hat{\beta}R)\Sigma_*^{-1}R^T\big(I - (I + \hat{\beta}R\Sigma_*^{-1}R^T)^{-1}\hat{\beta}R\Sigma_*^{-1}R^T\big)\bar{\boldsymbol{\beta}}$$

which yileds

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{f}_*} &= K_*^T K_{y*}\mathbf{y} + R^T\bar{\boldsymbol{\beta}} + \\ &\quad \big(R^T\hat{\beta}R\Sigma_*^{-1}R^T - (\Sigma_* + R^T\hat{\beta}R)\Sigma_*^{-1}R^T(I + \hat{\beta}R\Sigma_*^{-1}R^T)^{-1}\hat{\beta}R\Sigma_*^{-1}R^T\big)\bar{\boldsymbol{\beta}} \\ &= K_*^T K_{y*}\mathbf{y} + \Big(R^T(I + \hat{\beta}R\Sigma_*^{-1}R^T) - (I + R^T\hat{\beta}R\Sigma_*^{-1})\big(R^T(I + \hat{\beta}R\Sigma_*^{-1}R^T)^{-1}\hat{\beta}R\Sigma_*^{-1}R^T\big)\Big)\bar{\beta}\end{aligned}$$

Therefore,

$$\boldsymbol{\mu}_{f_*} = K_*^T K_y^{-1}\mathbf{y} + M\bar{\boldsymbol{\beta}},$$

where

$$M := R^T(I + \hat{\beta}R\Sigma_*^{-1}R^T) - (I + R^T\hat{\beta}R\Sigma_*^{-1})\big(R^T(I + R^T\hat{\beta}R\Sigma_*^{-1})^{-1}\hat{\beta}R\Sigma_*^{-1}R^T\big).$$

Also,

$$R^T(I + \hat{\beta}R\Sigma_*^{-1}R^T) = (I + R^T\hat{\beta}R\Sigma_*^{-1})R^T,$$

resulting in

$$M = R^T(I + \hat{\beta}R\Sigma_*^{-1}R^T)\big(I - (I + \hat{\beta}R\Sigma_*^{-1}R^T)^{-1}\hat{\beta}R\Sigma_*^{-1}R^T\big) \tag{6.25}$$

By applying matrix inversion lemma to (6.25), we will have

$$M = R^T(I + \hat{\beta} R \Sigma_*^{-1} R^T)(I + \hat{\beta} R \Sigma_*^{-1} R^T)^{-1} = R^T.$$

Hence

$$\boldsymbol{\mu}_{f_*} = K_*^T K_y^{-1} \mathbf{y} + R^T \bar{\boldsymbol{\beta}},$$

where

$$R = \Phi_* - \Phi K_y^{-1} K_*$$

Or equivalently,

$$\mu_{\mathbf{f}_*} = \Phi_*^T \bar{\boldsymbol{\beta}} + K_*^T K_y^{-1}(\mathbf{y} - \Phi^T \bar{\boldsymbol{\beta}}). \tag{6.26}$$

In summary, we have

$$p(\mathbf{f}_* | X_*, X, \mathbf{y}) \sim \mathcal{N}(\boldsymbol{\mu}_{f_*}, \Sigma_{f_*}),$$

where

$$\begin{cases} \boldsymbol{\mu}_{f_*} = \Phi_*^T \bar{\boldsymbol{\beta}} + K_*^T K_y^{-1}(\mathbf{y} - \Phi^T \bar{\boldsymbol{\beta}}) \\ \Sigma_{f_*} = K_{**} - K_*^T K_y^{-1} K_* + R^T (B^{-1} + \Phi K_y^{-1} \Phi^T)^{-1} R \\ \bar{\boldsymbol{\beta}} = (\Phi K_y^{-1} \Phi^T + B^{-1})^{-1} (\Phi K_y^{-1} \mathbf{y} + B^{-1} \mathbf{b}) \\ R = \Phi_* - \Phi K_y^{-1} K_*. \end{cases}$$

CHAPTER 7

WORK IN PROGRESS AND FUTURE WORK

This chapter provides some research that are still in progress and also some future work according to the results and the direction of the research that have been accomplished in this dissertation.

7.1 Work In Progress

This section represents two in progress works that yet need some extra work to be accomplished.

7.1.1 SBL-VB(GiG)-OMP: The Modified Version of SBL-VB(GiG)

Here, the focus is on the Gaussians-inverse Gammas modeling and the corresponding SBL-VB(GiG) algorithm in Chapter 5. More specifically, in Section 5.4.2, it was shown that the reconstructed signal via SBL-VB(GiG) tends to be nonsparse. Then, in Section 5.5.2 an attempt was made to improve the performance of this algorithm via a post-filtering step in an *ad hoc* way. The question that is examined here is whether we can do something other than finding the threshold in an experimental fashion, discussed in Section 5.5.2, and achieve better performance. For this purpose, here the SBL-VB(GiG) algorithm is combined with the greedy algorithm of orthogonal matching pursuit (OMP). OMP is a well-known sub-optimal algorithm that solves for the inverse problem of compressive sensing [76, 157]. At each iteration, OMP seeks the column of the sensing matrix A which has the maximum usefulness, where the usefulness u_j is related to the normalized version of the correlation between the residual at the current iteration and the jth column of the matrix A. Once the maximum usefulness is found, the corresponding index is added to the support set which denotes the nonzero locations in the solution vector $\hat{\boldsymbol{x}}_s$. The residual is then updated via finding the solution vector $\hat{\boldsymbol{x}}_s$ where just the corresponding supports are considered. The stopping condition of OMP can be made based on one of the three following criteria. The commonly used criterion is based on the knowledge on the sparsity level of the underlying sparse signal. However, in almost all practical applications, the sparsity level is unknown. Another stopping rule can be made based on having a fair knowledge on the measurement noise variance σ^2 in (5.1), which can be defined as [158]

$$\|\boldsymbol{r}\|_2^{[k]} < \sigma\sqrt{M + 2\sqrt{M \log M}}, \tag{7.1}$$

where $\|\boldsymbol{r}\|_2^{[k]} = A\hat{\boldsymbol{x}}_s^{[k]}$ is the residual at the kth iteration, and M is the number of measurements. Again, the noise characteristics are usually unknown. Also, the noise in the CS problem is defined based on the combination of the measurement noise and the small noise in the compressible signal to be treated as a sparse signal. Therefore, the estimated noise variance obtained from any good CS algorithm is generally off from the true variance of the measurement noise. The other stopping rule for the OMP algorithm can be made as follows.

$$\frac{\|\boldsymbol{r}^{[k]} - \boldsymbol{r}^{[k-1]}\|_2}{\|\boldsymbol{r}^{[k]}\|_2} \leq \text{Threshold.} \tag{7.2}$$

The OMP with the stopping rule (7.2) is sensitive to noise and as the noise increases, the performance of OMP decreases drastically. Also, the solution may tend to become nonsparse caused by the effort to fit the estimation to the model either via (7.2) or the criterion on the energy of the current residual itself.

The real question is whether we can somehow estimate the sparsity level in order to feed it to the OMP algorithm. If the active locations were known, then it would not cost us a lot to solve the problem for the very sparse signals. Suppose that there are K nonzero locations and the length of signal is N, where $K \ll N$. If the locations were known, then K measurements would suffice to solve the problem. Since the nonzero locations are unknown, we expect to have more than K measurements. Therefore, it is not possible to do perfect sparse recovery with the number of measurements lower than K. There exist some work regarding the issues with the OMP such as [157, 159–164]. In [157], it was shown that $M_{\min} = 4K \log N$ is the minimum number of measurements for a K sparse signal for the noiseless case. Fletcher and Rangan provided a better lower bound on the number of measurements in noisy CS as $M_{\min} = 2K \log\{N-K\}$ [160, 162]. They further showed that if K is unknown, then having knowledge on the bound of sparsity yields $M_{\min} = 2K_{\max} \log\{N - K_{\min}\}$, where $K_{\min} \leq K \leq K_{\max}$. Using the restricted isometry constant (RIC), Wen et al., proposed conditions for exact support recovery of K-sparse signal [163,164]. However, since we assume that the sparsity level is unknown, we cannot directly use the above ideas for our goal.

With the focus on the performance of SBL-VB(GiG), here an attempt is made to propose a new algorithm which is based on the combination of SBL-VB(GiG) and a modified version of OMP. As discussed above, somehow we need to know the sparsity level. For this purpose, we estimate the lower bound on the sparsity level of the underlying sparse signal using the reconstructed signal from the SBL-VB(GiG) algorithm. Suppose we have access to the true underlying signal $\boldsymbol{x}$ with $\boldsymbol{x} \in \mathbb{R}^N$. Then the numerical sparsity for this signal can be defined as [165]

$$s(x) := \frac{\|\boldsymbol{x}\|_1^2}{\|\boldsymbol{x}\|_2^2}, \tag{7.3}$$

which satisfies $1 \leq s(x) \leq N$ for any nonzero signal $\boldsymbol{x}$. Since the estimation of $\boldsymbol{x}$ using SBL-VB(GiG) may become poor for very low sampling ratios, it may overestimate the sparsity defined in (7.3). We define the estimated numerical sparsity as

$$\hat{K}_{\min} = \min\left(10\%N, \hat{K}_{\min,SBL-VB(GiG)}\right). \tag{7.4}$$

The upper bound on the sparsity is taken to be

$$\hat{K}_{max} = N/2, \tag{7.5}$$

otherwise the signal would not be sparse. This bound is incorporated into the algorithm. Below, the steps of the proposed algorithm are represented.

1) SBL-VB(GiG) only keeps the $\hat{K}_{max} = N/2$ highest amplitudes of the estimated signal and sets the other nonzero components to zero. The modified SBL-VB(GiG) for this purpose is

denoted as SBL-GVB(GiG).

SBL-GVB(GiG) Algorithm:

```
$\hat{\boldsymbol{x}}$ = **SBL-VB-GiG**$(Y, A)$
Iter= 1
**While** $\frac{|L^{[\text{Iter}]}-L^{[\text{Iter}-1]}|}{|L^{[\text{Iter}-1]}|} \geq 10^{-6}$
    Compute $\Sigma_{\tilde{x}}$ and $\tilde{\boldsymbol{x}}$ from (5.17)      % (Solution-value matrix component)
    Filtering the $N/2$ components of $\hat{\boldsymbol{x}}$ with lowest amplitudes
    Compute $T$ from (5.15)      % (Precisions on the solution)
    Compute $\varepsilon$ from (5.16)      % (Precision on the noise)
    Compute $L^{[\text{Iter}]}$ and then Iter=Iter+1
**End While**
```

2) The lower bound on the sparsity level of the signal is estimated from the approximated signal obtained from step 1 using (7.3). The obtained numerical sparsity will then computed by (7.4) and it will act as the minumum number of supports that we expect. Also, the upper loose bound will be computed from (7.5).

3) Then the modified version of the OMP algorithm is applied. The modified OMP that is proposed here is fed with the estimated noise variance obtained from SBL-VBG(GiG) i.e., (σ^2), $\hat{K}_{\min}$, $\hat{K}_{\max}$, $\boldsymbol{y}$, A and the index of $\hat{K}_{min}$ estimated supports from SBL-VBG(GiG). These supports correspond to the location of the $\hat{K}_{\min}$ largest amplitudes in $\hat{\boldsymbol{x}}$ of SBL-VBG(GiG), where we denote this set as $\hat{\mathcal{S}}_{\min}$. Unlike the regular OMP, our modified version does not start with an empty set for the supports, but rather contains the elements in the set $\hat{\mathcal{S}}_{\min}$. Also, $\hat{\mathcal{S}}_{max}$ contains the index of the $N/2$ highest amplitudes in $\hat{\boldsymbol{x}}$. Then, the supports are added from the set $\hat{\mathcal{S}}_{\max} \setminus \hat{\mathcal{S}}_{\min}$ to the support set $\hat{\mathcal{S}}$ the same the regular OMP. We make the stopping condition either when the cardinality of the support set reaches $\hat{K}_{max}$ or when the current residual satisfies (7.1) with $\hat{\sigma}^2$ instead of the true noise variance. The

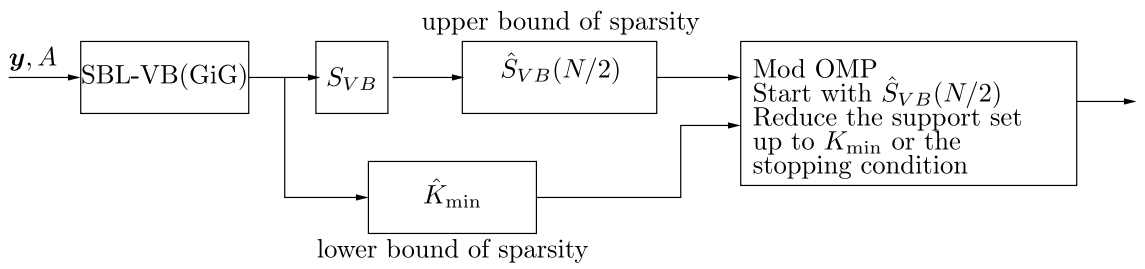
pseudo code of the modified OMP is described below.

Mod-OMP Algorithm:

```
[$\hat{\boldsymbol{x}}, \hat{\boldsymbol{s}}$] = Mod-OMP($Y, A, \hat{K}_{min}, \hat{K}_{max}, \hat{\mathcal{S}}_{min}, \hat{\mathcal{S}}_{max}, \hat{\sigma^2}$)

Initialization:
$\boldsymbol{x}^{(0)} = \boldsymbol{0}$, $\boldsymbol{r}^{(0)} = \boldsymbol{y}$, $\Omega = \hat{\mathcal{S}}_{max}$, and $S^{(0)} = \hat{\mathcal{S}}_{min}$
Increment $k$ by one and perform the followings:

  • Excluding the elements in $\mathcal{S}^{(k-1)}$ from the set $\Omega$
    $\bar{\Omega} = \Omega \setminus \mathcal{S}^{(k-1)}$

  • Computing the subset of the sensing matrix $A$ that contains the columns of $A$ corresponding to
    the elements in the set $\Omega$
    $A_{\bar{\Omega}} = A(:, \bar{\Omega})$

  • Compute usefulness:
    $u_j = (\boldsymbol{a}_j^T \boldsymbol{r}^{(k-1)} / \|\boldsymbol{a}_j\|)^2, \forall j \in \bar{\Omega}$

  • Update the support set:
    Select the index in $\bar{\Omega}$ with maximum usefulness and update the support set by $S^{(k)} = S^{(k-1)} \cup j_0$

  • Update solution: Compute $\boldsymbol{x}^{(k)}$ which is the minimizer of $\|A\boldsymbol{x}^{(k)} - \boldsymbol{y}\|_2^2$ subject to support$\{\boldsymbol{x}^{(k)}\} = S^{(k)}$

  • Update residual: Compute $\boldsymbol{r}^{(k)} = \boldsymbol{y} - A\boldsymbol{x}^{(k)}$

Repeat the process until $|S^{(k)}| = \hat{K}_{max}$ or (7.1) for the current residual
```

The block-diagram representing the modified SBL-VB(GiG) is illustrated in the Fig. 7.2.

As an initial simulation results, here the performance of the SBL-GVB(GiG) is compared with some of other state-of-the-art algorithms. More specifically, we consider the OMP algorithm when it is fed with the true sparsity level, Bayesian compressive sensing (BCS) [50], multi-task compressive sensing [66], and sparse Bayesian learning (MSBL) [19] algorithms. For this purpose, 200 random trials were generated in the same way explained in



Section 5.3 and Section 5.4. We study the performance of the algorithms for three levels of SNR with SNR:= 10, 20, and 30 dB. Figures Fig. 7.1 and Fig. 7.2 illustrate the results based on the normalized mean-squared error between the true and the reconstructed solutions, and also the difference between the detection and false alarm rate in finding the active locations of the true signal, respectively. According to Fig. 7.1 and Fig. 7.2, the NMSE comparison of our algorithm is close to the performance of MTCS in both metric comparisons. The OMP generally outperforms all the algorithms in terms of support recovery. This behavior of the OMP was expected, because it was fed with the true sparsity level for the SNR = 10 dB. The BCS algorithm provided a litte better results in terms of the error rate but with a lower performance in our support recovery evaluation. This means that the estimated signal via BCS tends to be nonsparse with many active locations, where most of the components have very small amplitudes. For the SNR = 20 dB, we observe that the error rates of our algorithm is close to BCS and MTCS. However, the proposed algorithm outperformed the other algorithms in support recovery for a wide range of sampling ratios. Finally, for SNR = 30 dB, we still see the same comparison performance of the algorithms as for the case with SNR = 20 dB.

It seems that the performance of the proposed algorithm could still be improved if another competing algorithm other than OMP is incorporated. The reason is because the OMP algorithm is sensitive to the noise. Another approach is to fuse the SBL-VB(GiG) algorithm with a greedy algorithm in a parallel sense. This means that we can feed both OMP (or another algorithm) and OSBL-VB(BGiG/GiG) with the measurements and let each of the algorithms estimate the support sets. Then, one can make decision on the

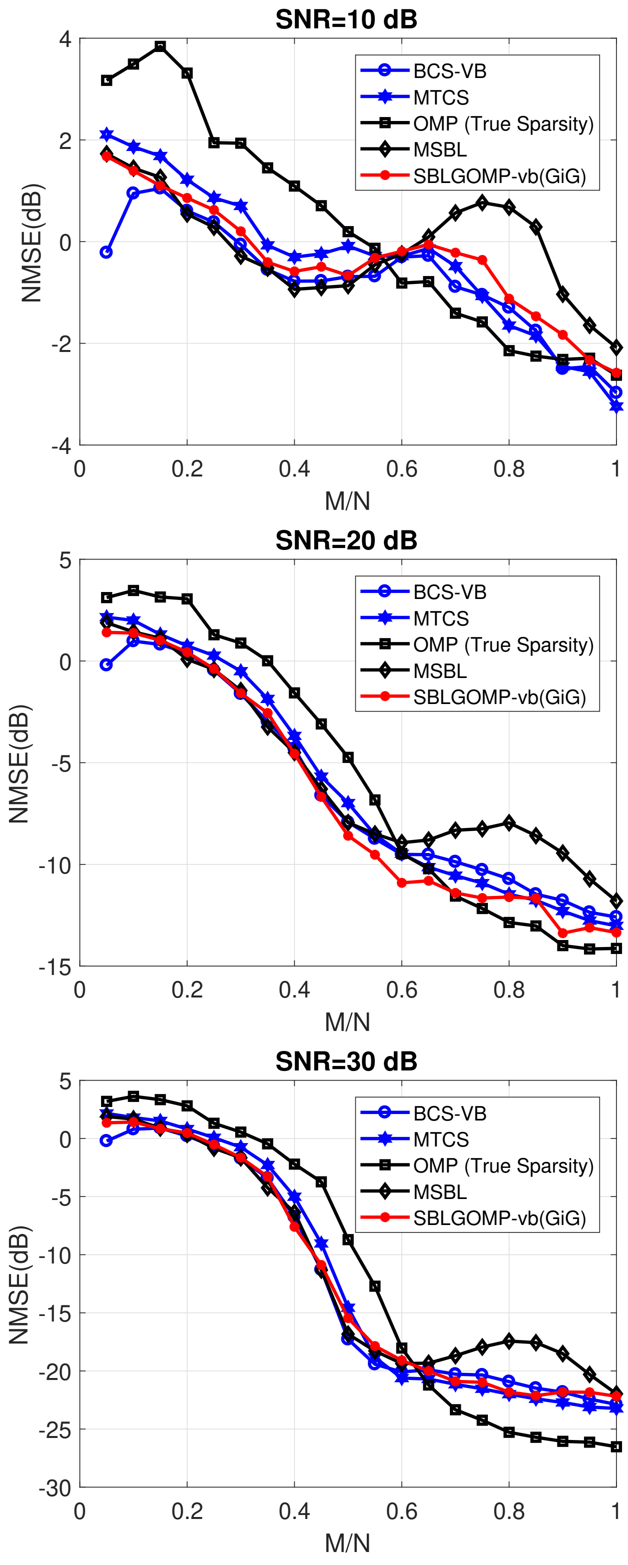


Fig. 7.1: Performance comparison in terms of the averaged NMSE.

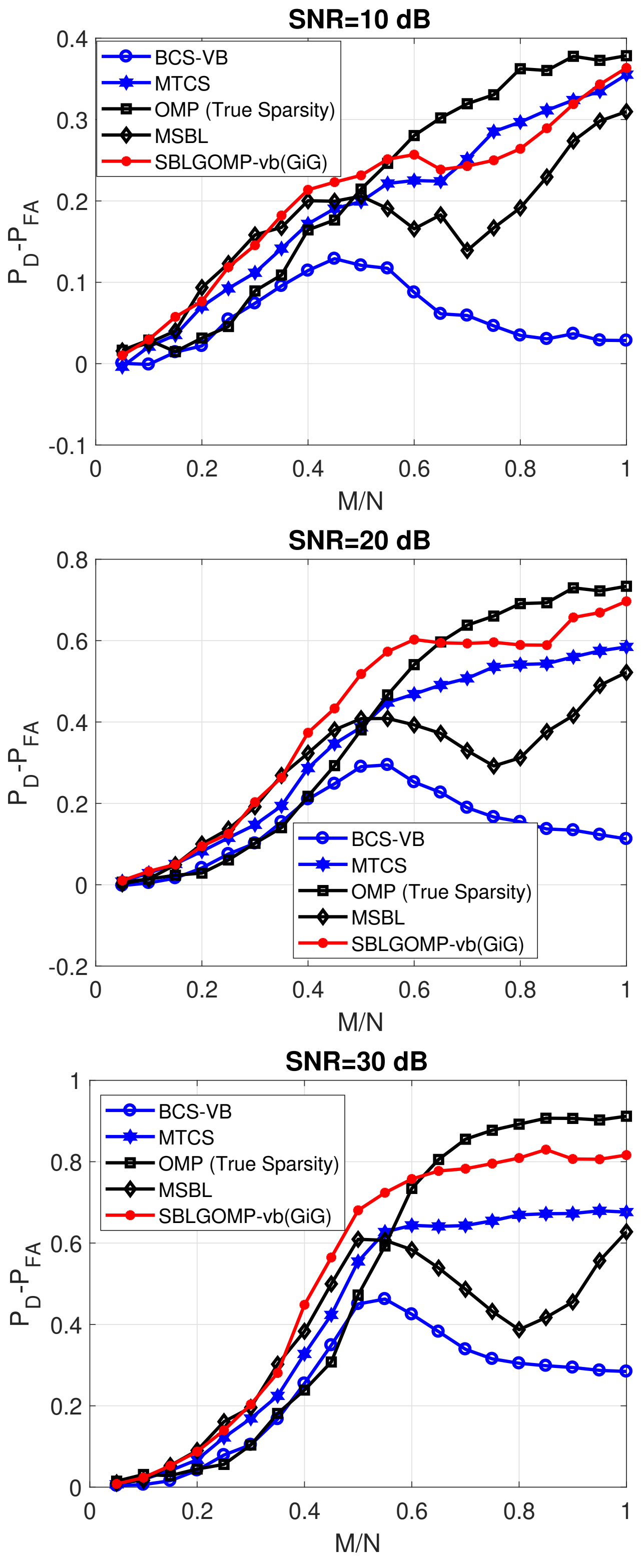


Fig. 7.2: Performance comparison in terms of averaged $P_D - P_{FA}$.

supports based on the intersection and the union of the supports found in each of the algorithms. The idea behind fusing the algorithms here is derived from [166]. However, we usually do not know the sparsity level to feed it to the OMP or any other greedy-based algorithm. This problem may still be resolved by estimating the lower bound on the sparsity level proposed in [165]. This task will add a bit more computational complexity to the algorithm at the cost of providing more reliable estimation on the sparsity level compared to the estimation based on the SBL-VB(GiG) mainly for low sampling ratios.

7.1.2 C-SBL(VB) Modeling and Algorithm

In this section, another new algorithm is proposed for solving the inverse problem of compressive sensing for sparse signals with unknown clustering pattern. This algorithm has a close relationship with the C-SBL(MCMC) algorithm proposed in Chapter 2. The difference of this algorithm with C-SBL(MCMC) is in the prior modeling of the support vector $\mathbf{s}$ and also in the inference technique. Specifically, here a variational Bayes rather than MCMC inference is used, which makes the algorithm much faster than C-SBL(MCMC) in terms of execution time. Althought the modeling and inference has been accomplished here, still some extra work is required to deal with the issues in estimating the noise precision. Therefore, the simulation results and comparisons against other state-of-the-art algorithms are yet left to be done. Below, the prior modeling on the hidden variables and parameters of the model are described and justified.

As was stated in Chapters 2-5, the CS problem can be modeled as $\mathbf{y} = A(\mathbf{s} \circ \mathbf{x})$, where $\mathbf{x}_s = \mathbf{s} \circ \mathbf{x}$ is unkown and $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{y} \in \mathbb{R}^M$ is the measurement vector, and $\mathbf{e}$ is the noise. The prior modeling is defined as follows.

- Defining the priors of the C-SBL(VB) algorithm

$$\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \tau^{-1} I_N),\ \tau \sim \text{Gamma}(a_0, b_0) \tag{7.6}$$

$$\boldsymbol{e} \sim \mathcal{N}(\mathbf{0}, \varepsilon^{-1} I_M),\ \varepsilon \sim \text{Gamma}(\theta_0, \theta_1) \tag{7.7}$$

$$s_n \sim \text{Bernoulli}(\gamma_n),\ \gamma_n \sim \text{Beta}(\alpha_{0,n}, \beta_{0,n}) \tag{7.8}$$

- Justification of the model

As a prior model, we assume that each element of the support vector $\boldsymbol{s}$ comes from a Bernoulli random variable governed by a probability γ_n defined in (7.8). Since γ_n is now defined as a probability, it can take vaules in the range $\gamma_n \in [0, 1]$. As a hyper-prior, we consider

$$\gamma_n \sim \text{Beta}(\alpha_{0,n}, \beta_{0,n}).$$

The good news for such a hyper-prior is that it already has the support range $[0, 1]$. Also, Bernoulli and Beta distributions are conjugates, thus it simplifies the approach for finfding the posterior distribution on s_n. The next question is how to promote clustered pattern solutions. For this purpose, we use the measure of clumpiness defined by $(\Sigma\Delta)$ which is a measure of total variation on the support vector, and is defined as

$$(\Sigma\Delta)(\boldsymbol{s}) := \sum_{n=1}^{N} |s_n - s_{n-1}|. \tag{7.9}$$

For the purpose of increasing or decreasing the probability of a specific element s_n, the following procedure is performed. According to (7.9) and the desire for promoting the clustered pattern supports, intuitively one may wish to have $(\Sigma\Delta)$ to be small. Therefore, in case of having $s_{n-1} = s_{n+1} = 1$, we would like to give s_n a higher chance to become active, otherwise the measure of $(\Sigma\Delta)$ increases. In case of having $s_{n-1} = 1, s_{n+1} = 0$ or $s_{n-1} = 0, s_{n+1} = 1$, we may wish to have equal chances for s_n to be active or inactive. For the case of having $s_{n-1} = s_{n+1} = 0$, we would prefer to give s_n less chance to become active, otherwise the measure of $(\Sigma\Delta)$ increases, yielding to a nonsparse solution.

The question then becomes how to increase or decrease the probability of a specific element 's_n'. We measure $(\Sigma\Delta)(\boldsymbol{s})$ for both cases of $s_n = 0$ and $s_n = 1$. In other words, we compute

$$(\Sigma\Delta)_{0,n} = \sum_{i=2}^{N} |s_i - s_{i-1}|,\ \forall s_n = 0$$

and

$$(\Sigma\Delta)_{0,n} = \sum_{i=2}^{N} |s_i - s_{i-1}|,\ \forall s_n = 1$$

Then, the above measurements are used to promote the clustered pattern in the solution. Let us first take a look at the difference between $(\Sigma\Delta)_{0,n}$ and $(\Sigma\Delta)_{1,n}$.

$$(\Sigma\Delta)_{0,n} - (\Sigma\Delta)_{1,n} = 2(s_{n-1} + s_{n+1} - 1), \tag{7.10}$$

where in case of binary outcomes for s_n we have

$$(\Sigma\Delta)_{0,n} - (\Sigma\Delta)_{1,n} = \begin{cases} 2\ ,\ s_{n-1} = s_{n+1} = 1 \\ 0\ ,\ s_{n-1} = 0, s_{n+1} = 1 \text{ or } s_{n-1} = 1, s_{n+1} = 0 \\ -2\ ,\ s_{n-1} = s_{n+1} = 0. \end{cases}$$

Based on (7.8) and (7.10), we can then expect to have the following result on the average for γ_n

$$\gamma_n = \frac{1}{1 + e^{-\alpha(\bar{\Sigma\Delta})_n}},\ \forall n \tag{7.11}$$

where

$$(\bar{\Sigma\Delta})_n := (\Sigma\Delta)_{n,0} - (\Sigma\Delta)_{n,1},\ \text{for some } \alpha > 0. \tag{7.12}$$

In (7.11), the term α is an emphasizing factor on the amount of clumpiess on the support vector of the solution. Therefore, we use a Beta distribution on γ_n as

$$\gamma_n \sim \text{Beta}\big(e^{-\alpha(\Sigma\Delta)_{n,1}}, e^{-\alpha(\Sigma\Delta)_{n,0}}\big),\ \forall n, \tag{7.13}$$

which yields to

$$E[\gamma_n] = \frac{1}{1 + e^{-2\alpha(s_{n-1}+s_{n+1}-1)}}.$$

Finally, the prior on s_n will be defined as

$$s_n \sim \text{Bernoulli}(\gamma_n), \begin{cases} \gamma_n \sim \text{Beta}\big(1, e^{-2\alpha(s_{n-1}+s_{n+1}-1)}\big), \ \forall n = 2, 3, ..., N-1 \\ \gamma_n \sim \text{Beta}\big(1, e^{-2\alpha(2s_2-1)}\big), \ \forall n = 1 \\ \gamma_n \sim \text{Beta}\big(1, e^{-2\alpha(2s_{N-1}-1)}\big), \ \forall n = N. \end{cases} \tag{7.14}$$

Although the support modeling in (7.14) seems to be a reasonable choice, it turns out that learning the parameter α using variational Bayes yields computing the expectation of non-linear terms, which will make the learning procedure inefficient. Instead, below another way of defining the prior on the componet s_n is considered.

- Defining the prior on the elements of the support vector

$$s_n \sim \text{Bernoulli}(\gamma_n), \ \gamma_n \sim \text{Beta}(\alpha_{0,n}, \beta_{0,n}), \tag{7.15}$$

where we set $\alpha_{0,n} = 1$ and

$$\beta_{0,n} = \exp\{-(s_{n-1} + s_{n+1} - 1.2)\} \tag{7.16}$$

According to (7.15) and (7.16), we expect to have the following values on the average for γ_n

$$E[\gamma_n] = \begin{cases} 0.2315 \ , \ s_{n-1} = s_{n+1} = 0 \\ 0.4502 \ , \ s_{n-1} = 0, s_{n+1} = 1 \text{ or } s_{n-1} = 1, s_{n+1} = 0 \\ 0.6900 \ , \ s_{n-1} = s_{n+1} = 1. \end{cases} \tag{7.17}$$

We then approximate the exponential term (7.16) via least squares, which yields to

$$\beta_{0,n} \simeq -1.3052(s_{n-1} + s_{n+1}) + 2.6321,$$

resulting in

$$E[\gamma_n] = \begin{cases} 0.2753\ ,\ s_{n-1} = s_{n+1} = 0 \\ 0.4298\ ,\ s_{n-1} = 0, s_{n+1} = 1 \text{ or } s_{n-1} = 1, s_{n+1} = 0 \\ 0.9788\ ,\ s_{n-1} = s_{n+1} = 1. \end{cases} \tag{7.18}$$

Below, the update rules of the parameters and variables of the proposed model using variational Bayes is represented in detail.

- The update rules of the variables and parameters of the model using variational Bayes inference
  - Update rule for γ_n using VB

$$\begin{aligned} q_{\gamma_n}(\gamma_n) &\propto p(\gamma_n; s_{n-1}, s_{n+1}) \exp\{\langle \log p(\boldsymbol{s}|\boldsymbol{\gamma})\rangle_{q_s}\} \\ &\propto \gamma_n^{\alpha_{0,n}-1}(1-\gamma_n)^{-1.3052(s_{n-1}+s_{n+1})+2.6321-1} \exp\{\langle \log p(s_n|\gamma_n)_{q_{s_n}}\rangle\} \\ &\propto (1-\gamma_n)^{-1.3052(s_{n-1}+s_{n+1})+2.6321-1} \exp\{\langle \log\{\gamma_n^{s_n}(1-\gamma_n)^{1-s_n}\}\rangle_{q_{s_n}}\} \\ &\propto (1-\gamma_n)^{-1.3052(s_{n-1}+s_{n+1})+2.6321-1}\gamma_n^{\tilde{s}_n}(1-\gamma_n)^{1-\tilde{s}_n} \\ &\propto \gamma_n^{(1+\tilde{s}_n)-1}(1-\gamma_n)^{\left(3.6321-\tilde{s}_n-1.3052(s_{n_1}+s_{n+1})\right)-1}, \end{aligned}$$

which yields to

$$q_{\gamma_n}(\gamma_n) \sim \text{Beta}\big(\alpha_{1,n}, \beta_{1,n}\big), \tag{7.19}$$

where $\alpha_{1,n} := 1 + \tilde{s}_n$ and $\beta_{1,n} := 3.6321 - \tilde{s}_n - 1.3052(s_{n+1} + s_{n+1})$. Therefore, the update rule for γ_n can be written as

$$\begin{aligned} \tilde{\gamma}_n &= \frac{\alpha_{1,n}}{\alpha_{1,n} + \beta_{1,n}} \\ &= \frac{1 + \tilde{s}_n}{4.6321 - 1.3052(s_{n-1} + s_{n+1})}. \end{aligned} \tag{7.20}$$

  - Update rule for the support vector components

$$\begin{aligned} q_{s_n}(s_n) &\sim \exp\{\langle \log\{p(\boldsymbol{x},\boldsymbol{s},\boldsymbol{y}|\theta,\mathcal{M})\}\rangle_{q_\theta q_x}\} \\ &\propto e^{\{\left\langle \log\{p(\boldsymbol{x}\boldsymbol{s}|\theta)\}\right\rangle_{q_\theta q_x}\}} e^{\{\left\langle \log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\theta,\mathcal{M})\}\right\rangle_{q_\theta q_x}\}} \\ &\propto e^{\left\langle \log p(s_n;\gamma_n)\right\rangle_{q_{\gamma_n}}} e^{\{\left\langle \log\{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\}\right\rangle_{q_{s^{-n}} q_x q_\varepsilon}\}} \end{aligned}$$

Therefore,

$$q_{s_n}(s_n) \propto e^{\left\langle \log\{\gamma_n^{s_n}(1-\gamma_n)^{1-s_n}\}\right\rangle_{q_{\gamma_n}}} e^{\left\langle \log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\right\rangle_{q_{s_{-n}} q_x q_\varepsilon}}. \tag{7.21}$$

After some simplification, the update rule for the components of the support vector will become

$$q_{s_n}(s_n) \sim \text{Bernoulli}\Big(\frac{1}{1+e^{\{\psi(\beta_{1,n}-\psi(\alpha_{1,n})+\frac{1}{2}\tilde{\varepsilon}\left(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma_{\tilde{x}_n}^2)-2\tilde{x}_n\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n}\right))\}}}\Big),$$

where $y_m^{-n} := y_m - \sum_{l\neq n}^N a_{ml}s_l x_l$ and $\boldsymbol{y}^{-n} := [y_1^{-n}, \cdots, y_M^{-n}]^T$. Finally, we have

$$\tilde{s}_n = \frac{1}{1+e^{\{\psi(\beta_{1,n}-\psi(\alpha_{1,n})+\frac{1}{2}\tilde{\varepsilon}\left(\|\boldsymbol{a}_n\|_2^2(\tilde{x}_n^2+\sigma_{\tilde{x}_n}^2)-2\tilde{x}_n\boldsymbol{a}_n^T\tilde{\boldsymbol{y}}^{-n}\right))\}}}, \ \forall n. \tag{7.22}$$

– Update rule for the solution value vector

$$\begin{aligned} q_x(x) &\sim e^{\{\left\langle \log p(\boldsymbol{x},\boldsymbol{s},\boldsymbol{y}|\theta,\mathcal{M})\right\rangle_{q_\theta q_s}\}} \\ &\propto e^{\left\langle \log p(\boldsymbol{x}|\tau)\right\rangle_{q_\theta}} e^{\left\langle \log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\right\rangle_{q_\varepsilon q_s}} \end{aligned}$$

After some work, we will end up with $q_x(\boldsymbol{x}) \sim \mathcal{N}(\tilde{\boldsymbol{x}}, \Sigma_{\tilde{x}})$, where

$$\Sigma_{\tilde{x}} = \Big(\tilde{\tau} I_N + \tilde{\varepsilon}\big((A^TA)\circ(\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \text{diag}\{\tilde{\boldsymbol{s}}\circ(1-\tilde{s})\})\big)\Big)^{-1}$$

and

$$\tilde{\boldsymbol{x}} = \tilde{\varepsilon}\Sigma_{\tilde{x}}\tilde{S}A^T\boldsymbol{y},$$

where $\tilde{S} := \text{diag}\{\tilde{\boldsymbol{s}}\}$.

– Update rule for the precision 'τ' of the solution vector $\boldsymbol{x}$

$$\begin{aligned} q_\tau(\tau) &\sim p(\tau; a_0, b_0) e^{\left\langle \log p(\boldsymbol{x}|\tau I_N)\right\rangle_{q_x}} \\ &\propto \tau^{a_0-1} e^{-b_0\tau} e^{\left\langle \log \{\prod_{n=1}^N p(x_n|\tau^{-1})\}\right\rangle_{q_x}} \\ &\propto \tau^{a_0-1} e^{-b_0\tau} e^{\left\langle \log \{\tau^{\frac{N}{2}} e^{-\frac{\tau}{2}\|\boldsymbol{x}\|_2^2}\}\right\rangle_{q_x}}. \end{aligned}$$

After some work, we will then have

$$q_\tau(\tau) \sim \text{Gamma}\big(a_0 + \frac{N}{2}, b_0 + \frac{1}{2}(\|\tilde{\boldsymbol{x}}\|_2^2 + \sum_{n=1}^N \sigma_{\tilde{x}_n}^2)\big).$$

Therefore, the update rule for τ can be written as

$$\tilde{\tau} = \frac{a_0 + \frac{N}{2}}{b_0 + \frac{1}{2}(\|\tilde{\boldsymbol{x}}\|_2^2 + \sum_{n=1}^N \sigma_{\tilde{x}_n}^2)}. \tag{7.23}$$

– Update rule for the noise precision 'ε'

$$\begin{aligned} q_\varepsilon(\varepsilon) &\sim p(\varepsilon; \theta_0, \theta_1) e^{\left\langle \log \{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{s},\varepsilon)\}\right\rangle_{q_x q_s}} \\ &\propto \varepsilon^{\theta_0 - 1} e^{-\theta_1\varepsilon} e^{\left\langle \log \{\varepsilon^{\frac{M}{2}}\}\right\rangle_{q_x q_s}}, \end{aligned}$$

which yileds to

$$\begin{aligned} q_\varepsilon(\varepsilon) \sim \text{Gamma}\Big(\theta_0 + \frac{M}{2}, \theta_1 + \frac{1}{2}\Big(&\boldsymbol{y}^T\boldsymbol{y} - 2(\tilde{\boldsymbol{s}} \circ \tilde{\boldsymbol{x}})^T A^T \boldsymbol{y} + \\ &\text{Tr}\left((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}})((A^T A) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \text{diag}\{\tilde{\boldsymbol{s}} \circ (1 - \tilde{\boldsymbol{s}})\}))\right)\Big)\Big). \end{aligned}$$

Therefore, the update rule for the noise precision can be written as

$$\tilde{\varepsilon} = \frac{\theta_0 + \frac{M}{2}}{\theta_1 + \frac{1}{2}\Big(\boldsymbol{y}^T\boldsymbol{y} - 2(\tilde{\boldsymbol{s}} \circ \tilde{\boldsymbol{x}})^T A^T \boldsymbol{y} + \text{Tr}\left((\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T + \Sigma_{\tilde{x}})((A^T A) \circ (\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}^T + \text{diag}\{\tilde{\boldsymbol{s}} \circ (1 - \tilde{\boldsymbol{s}})\}))\right)\Big)}. \tag{7.24}$$

7.1.3 Adaptive Constellation Design via the Actual Trajectories and Orbit Assignment Based on ΔV-Budget

Here, we assume that there is a dictionary of possible trajectories generated via Keplerian orbital elements. The task now becomes selecting a trajectory from the dictionary of

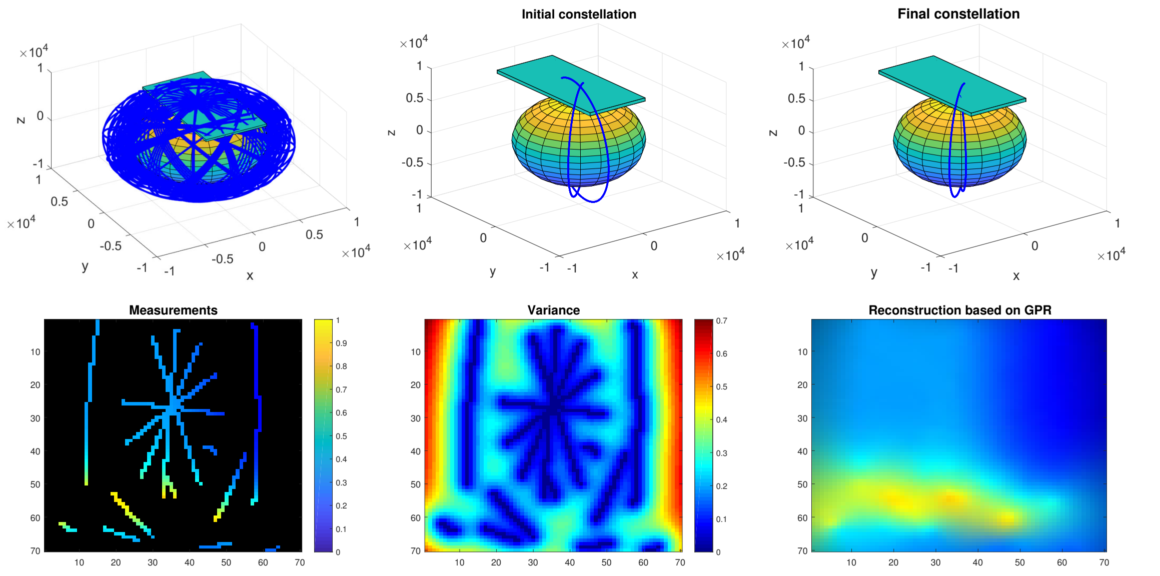


Fig. 7.3: First row illustrates the orbit dictionary, the initial and final constellation. Second row shows the measurements, measure of uncertainty, and the signal/image reconstruction of phenomenon based on the initial twelve successive trajectories when only taking into account the exploration desire.

feasible orbits based on the proposed framework in Chapter 6 in order to track the interesting phenomenon. Once a trajectory is selected, the required ΔV for the orbit transfer is computed for all the satellites in the constellation, and the one which requires minimum ΔV to transfer to the target orbit is chosen. The required ΔV is computed based on Hohmann transfer. An example of the orbit dictionary, initial and final constellation, and the reconstruction for the problem explained in Cahpter 6, is illustrated in Fig. 7.3. The decision making on the orbits, shown in Fig. 7.3, is only based on the exploration desire or equivalently the measure of variance computed in the first stage of the proposed framework in Chapter 6.

The results shown in Fig. 7.3 is the initial simulation results and further study is still required. Also, the results shown in this figure is only for the exploration purposes without taking into the account the desire for data refinement at the adjacency of already known (measured) locations. Therefore, the decision-maker should turn into a solver of the optimization problem explained in (6.8), where λ will be related to the affordable ΔV budget. The problem with a surrogate dictionary of trajectories rather than the feasible

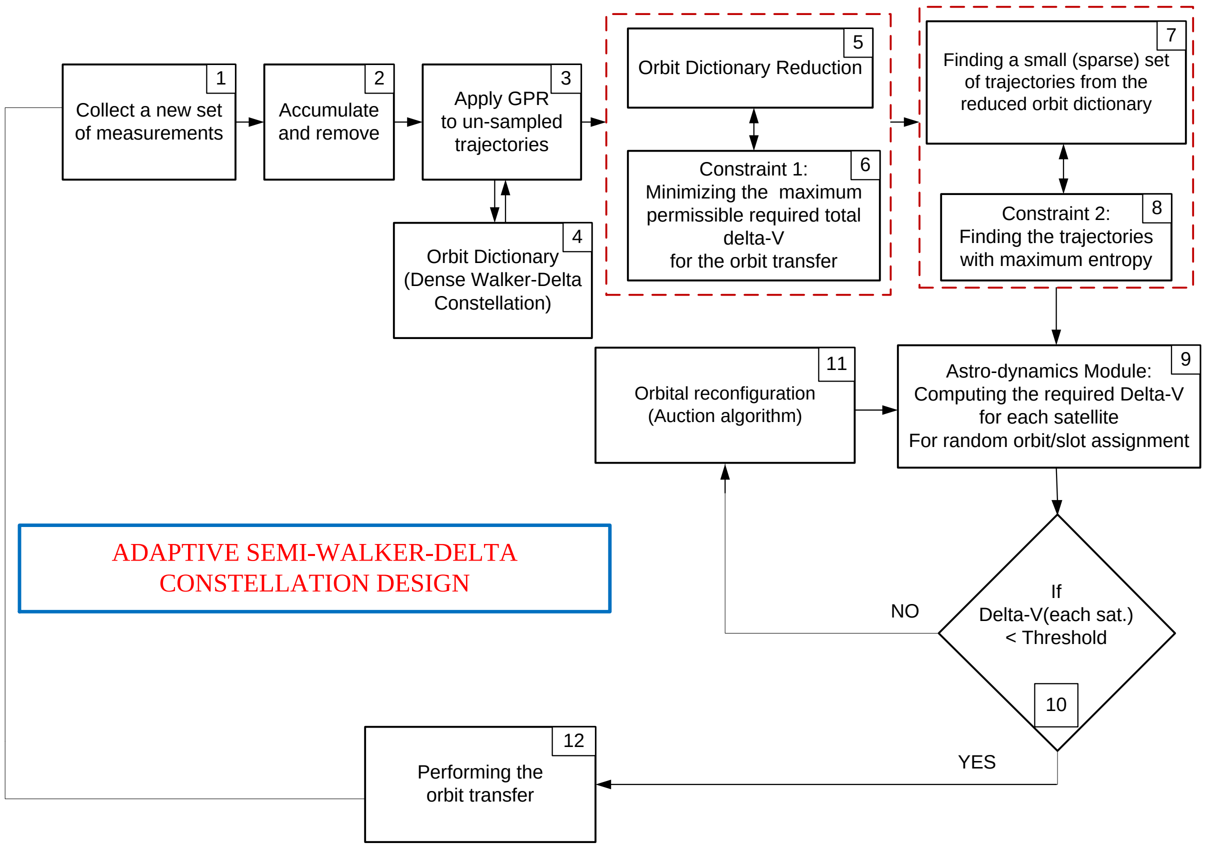

Collect a new set of measurements
1
Accumulate and remove
2
Apply GPR to un-sampled trajectories
3
Orbit Dictionary (Dense Walker-Delta Constellation)
4
Orbit Dictionary Reduction
5
Constraint 1: Minimizing the maximum permissible required total delta-V for the orbit transfer
6
Finding a small (sparse) set of trajectories from the reduced orbit dictionary
7
Constraint 2: Finding the trajectories with maximum entropy
8
Astro-dynamics Module: Computing the required Delta-V for each satellite For random orbit/slot assignment
9
If Delta-V(each sat.) < Threshold
10
Orbital reconfiguration (Auction algorithm)
11
Performing the orbit transfer
12
NO
YES
ADAPTIVE SEMI-WALKER-DELTA CONSTELLATION DESIGN

Fig. 7.4: Block diagram representing the proposed adaptive-semi-walker-delta constellation design.

trajectories has been partially published in [167].

In Fig. 7.4, we illustrate the proposed diagram as an initial strategy of the adaptive constellation desing for the exploration purposes. In this strategy, the dictionary of possible orbital planes contains a set of dense Walker-delta constellation for different inclination angles and altitues. However, one can modify the dictionary based on the objectives of a specific mission.

We refer to the constellation design illustrated in Fig. 7.4 as a semi-Walker-delta constellation. The reason is that only some orbits will be selected from a dictionary of dense Walker-delta constellation. Therefore, the resulting constellation will not be exactly a Walker-delta constellation. This is due to the fact that the selected orbits may not be necessarily equally spaces due to the selection made based on the constraint on the ΔV

budget. In Fig. 7.4, the Auction algorithm can be applied to decide which satellite from which orbit has a lower ΔV cost to perform orbital change and move to the selected orbit.

7.2 Future Work

In this section, some direction for the future work on the constellation desing of sensor-bearing satellites is provided. To reach this goal, three sub-problems are described below. However, once these problems are investigated and resolved, a more concrete and realistic framework, compared to the initial framework proposed in Chapter 6, for the constellation design of sensor-bearing satellites for the exploration problem can be proposed.

7.2.1 Reducing the Computational Complexity of GPR

In Chapter 6, a new framework was proposed for making decision on the trajectory of the mobile sensors. As discussed in Chapter 6, the first stage of the framework is the GPR-stage, which has the duty of predicting the PoI at the unseen locations. Although using GPR model seems to be a powerful tool for the prediction problem, it becomes inefficinet as the size of training data, measurements for our case, increases. In other words, after visiting the region of interest for a couple of times, the size of measurements becomes large. This results in a big covariance matrix in the GPR model in (6.1), which is then required to be inverted. In order to restrain the increase in the number of training data fed to the GPR, we want to retain only the last M set of measurements corresponding to the set of measurements taken at last M th time frames, as explained in Section 6.3. Once a new trajectory is determined and the corresponding measurements are collected, we then seek adding the new information to the training set and discard the oldest set of measurements. This is due to the assumption that the oldest measurements may have very low correlation with the new measurements and the PoI may have been changed. This avoids inverting a big covariance matrix as we continue collecting new measurements. Some good references for dealing with this problem are [154, 155, 168–171].

7.2.2 Adaptive Constellation Design via the Actual Trajectories with Constraint on the Available ΔV Budget

This problem is also related to the problem raised in Chapter 6, as it adds a practical constraint to the problem raised in Section 7.2.1. Notice that in Section 7.2.2, the problem is set up such that the decision-maker decides on selecting an orbit with minimum ΔV cost. However, the problem raised here is that the ΔV budget is limited due to some practical constaints and yet we desire to fullfill both desires of exploration and data refinement in an unknown region. In this case, our formularization of deciding in the next trajectory of mobile sensors will become

$$k^{\star} = \underset{s^{(k)}}{\operatorname{argmax}} \left[q(U_{s^{(k)}}) - bm(U_{s^{(k)}})\right] + \lambda \frac{1}{(\Delta V)^{s^{(k)}}}, \tag{7.25}$$

where $k^{\star}$ denotes the resulting index obtained from (7.25) corresponding to the set of Keplerian orbital elements from the available dictionary of feasible orbits. Also, the term λ balances between the informational value of the decisions and the required energy to perform the orbit transfer.

7.2.3 Connection of CS Algorithms to the Constellation Configuration of Sensor-Bearing Satellites Problem

In Chapters 2-5, we proposed some new algorithms using the compressive sensing technique to recover a sparse or compressible signal from a small set of linear measurements. It worth mentioning that such algorithms are able to learn the pattern on the sparse signal without any sort of concerete prior knowledge on the structure of the signal. In real-world applications, there are also many phenomena which exhibit sparsity and probably with some smoothness behavior, either in time, space or some other domains.

Also, according to the work presented in Chapter 6, the ongoing work described in Section 7.1.3, and the remained study discussed in Section 7.2.3, the goal has been devoted to the constellation adaptation design for the exploration problem. In the settings considered for the proposed framework in Chapter 6, it was assumed that the mobile sensors are capable of taking direct measurements from the phenomenon along their trajectories. In this case, the training data to be fed to the GPR-stage of the poposed framework were collected based on the information with direct measurements taken along the trajectories of sensor-bearing satellites. In fact, if the phenomenon is already known to be sparse in some domain, then there is a high chance to take lower number of measurements and yet being able to reconstruct larger sub-regions of the whole region of under study using compressive sensing technique. Suppose that the sensors are able to take a small set of indirect measurements using the CS technique along their trajectories. This means that instead of obtaining a measurement $f(u_1^{(s)}, T_m)$ at time frame T_m, we get a collection of measurements modeled via $A\boldsymbol{f}(\mathbf{u}_1^{(s)}, T_m)$, which are now indirect measurements of the PoI. Matrix A represents the sensing device modeling and is designed such that it meets the requirements of CS to compressively sense the important information of the sparse phenomenon within the sensing range of sensors. In this setting, $\boldsymbol{u}_1^{(s)}$ will contain the central spatiotemporal information of all the cells in the corresponding sub-region. The reconstruction of the PoI at each sub-region along the trajectory of the sensors will then be accomplished via one of the proposed SBL algorithms in Chapters 2-5 for sparse signals with unknown clustering pattern. In summary, the measurements of the sub-regions will fed into a CS reconstruction algorithm in order to find a super sparse representation of the PoI in such sub-region via very few measurements. Then, the GPR stage of the proposed framework will be fed with the estimation of the PoI and it smoothes out the behavior of the PoI over the region under study. Decision making on the next trajectories of the sensors will be the same as the one explained in Chapter 6 or based on the ongoing or future work explained in this chapter.

CHAPTER 8
CONCLUSION

In this dissertation solving the inverse problem of compressive sensing to reconstruct the underlying sparse signal was investigated for the case where the signal exhibits unknown clustering pattern. In this case, some new Bayesian modeling and algorithms for solving the inverse problem of CS were proposed. These algorithms were implemented using techniques such as Markov chain Monte Carlo (MCMC), message passing, and variational Bayesian (VB) inference. The results shown to outperform other state-of-the-art algorithms for most cases. Some of the proposed algorithms were appeared in this chapter. Also, some other algorithms were proposed in this research, which are not included in this dissertation. However, they can be found in the curriculum vitae section appeared at the end of this dissertation. The list of contributions of this dissertation for the compressive sensing problem are itemized as follows.

- Contributions of Chapter 2
  - A new hierarchical Bayesian modeling and algorithm was proposed for the recovery of jointly-sparse signals with unknown clustering patterns for the multiple measurement vector (MMV) problem
  - The algorithm can be used to solve for either the single measurement vector (SMV) or MMVs
  - The modeling incorporates an emphasizing parameter on the clustering pattern over the supports of the solution, which is learned in a Bayesian fashion in the proposed algorithm. This parameter does not exist in the other existing algorithms and is learned via the proposed hierarchical algorithm
- Contributions of Chapter 3

  - A new sparse Bayesian learning (SBL) algorithm for solving the SMV problem for sparse signals with unknown clustering pattern was proposed
  - The modeling and the corresponding algorithm is based on Bayesian message passing and factor graphs
  - The algorithm was futher simplified via the approximate message passing (AMP) framework
  - The proposed algorithm has lower computational complexity compared to the MCMC based algorithm proposed in Chapter 2
  - However, the overall reconstruction performance of the MCMC-based algorithm described in Chapter 2 turns out to be higher than the algorithm proposed in Chapter 3

- Contributions of Chapter 4
  - Chapter 4 considers the recovery of sparse signals with unknown clustering pattern in the case of having partial erroneous prior knowledge on the supports of the signal
  - A new sparse Bayesian learning model was proposed to incorporate the erroneous prior knowledge on the supports of the solution and simultaneously learn the unknown clustering pattern
  - One more layer was added to the support-aided sparse Bayesian learning algorithm (SA-SBL). This layer adds a prior on the shape parameters of Gamma distributions, those modeled to account for the precision of the solution elements. The shape parameters are made depend on the total variations on the estimated supports of the solution

- Contributions of Chapter 5
  - The performance of sparse signal recovery from a set of compressively sensed noisy measurements using variational Bayesian (VB) inference was investigated

- Two Bayesian modellings on the signal were considered, the issues of each model were studied, and the reconstruction performances were compared
- The issues of each modeling using variational Bayes inference for only promoting the sparsity had not been investigated before

In Chapter 6, the exploration problem using multiple mobile sensors to explore an unknown region was studied. The exploration probelm here was defined such that it trades off between two desiderata of continuing to take data in a region known to be interesting with the intent of refining the measurements *vs.* to take data in unobserved areas to attempt to discover new interesting regions. For this problem, a new framework based on Gaussian regression models was proposed. The framework also employs a decision making stage to select the trajectories of sensors. The decision-maker stage is based on an epistemic utility controller. As an application, a surrogate example for the exploration problem using a constellation of satellites with the goal of tuning or changing the orbital planes of the constellation was considered. The proposed framework is not only limited to sensor-bearing satellites but also is applicable to sensors carried by mobile robots and so on.

REFERENCES

[1] M. Mishali and Y. C. Eldar, "Blind multiband signal reconstruction : Compressed sensing for analog signals," *IEEE Trans. Sig. Proc.*, vol. 57, no. 3, pp. 993–1009, 2009.

[2] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Sig. Proc. Mag.*, vol. 25, no. 2, pp. 83–91, 2008.

[3] C. Li, *An Efficient Algorithm for Total Variation Reguralization with Applications to the Single Pixel-Camera and Compressive Sensing.* PhD dissertation, Rice University, 2010.

[4] J. Yang, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Imag. Proc.*, vol. 19, no. 11, pp. 2861–2873, 2010.

[5] Z. Pan, H. Huang, S. Hu, A. Zhang, H. Ma, and W. Sun, "Super-resolution based on compressive sensing and structural self-similarity for remote sensing images," *IEEE Trans. Geoscience and Remote Sensing*, vol. 51, no. 9, pp. 4864–4876, 2013.

[6] T. Wan and Z. Qin, "An application of compressive sensing for image fusion," *Int. J. Comp. Math.*, vol. 88, no. 18, pp. 3915–3930, 2011.

[7] S. Li, H. Yin, and L. Fang, "Remote sensing image fusion via sparse representations over learned dictionaries," *IEEE Trans. Geoscience and Remote Sensing*, vol. 51, no. 9, pp. 4779–4789, 2013.

[8] M. Lustig, D. Donoho, and J. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magnetic Resonance Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[9] C. R. Berger, Z. Wang, J. Huang, and S. Zhou, "Applications of compressive sesing to sparse channel estimation," *IEEE Com. Mag.*, vol. 48, no. 11, 2010.

[10] M. T. Alonso, P. Lopez-Dekker, and J. J. Mallorqui, "A novel strategy for radar imaging based on compressive sensing," *IEEE Trans. Geoscience and Remote Sensing*, vol. 48, no. 12, pp. 4285–4295, 2010.

[11] J. H. Ender, "A compressive sensing applied to radar," *Signal Processing*, vol. 90, no. 5, pp. 1402–1414, 2010.

[12] J. Yang, X. Yuan, P. Llull, D. J. Brady, G. Sapiro, and L. Carin, "A compressive sensing applied to radar," *IEEE Trans. Image Proc.*, vol. 23, no. 11, pp. 4863–4878, 2014.

[13] P. Nagesh and B. Li, "A compressive sensing approach for expression-invariant face recognition," in *IEEE Conf. on CVRP*, 2009, pp. 1518–1525.

[14] X. Ding, L. He, and L. Carin, "Bayesian robust principal component analysis," *IEEE Trans. Image Proc.*, vol. 20, no. 12, pp. 3419–3430, 2011.

[15] J. Fang, Y. Shen, H. Li, and P. Wang, "Pattern-coupled sparse Bayesian learning for recovery of block-sparse signals," *IEEE Trans. Sig. Proc.*, vol. 63, no. 2, pp. 360–372, 2015.

[16] D. Cohen, K. V. Mishra, and Y. C. Eldar, "Spectrum sharing radar: Coexistence via xampling," *arXiv preprint arXiv:1611.06443*, 2016.

[17] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Info. Th.*, vol. 52, no. 2, pp. 489–509, 2006.

[18] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Trans. Sig. Proc.*, vol. 58, no. 1, pp. 269–280, 2010.

[19] D. P. Wipf and B. D. Rao, "An empirical Bayesian strategy for solving the simultaneous sparse approximation problem," *IEEE Trans. Sig. Proc.*, vol. 55, no. 7, pp. 3704–3716, 2007.

[20] Z. Zhang and B. D. Rao, "Extension of SBL algorithms for the recovery of block sparse signals with intra-block correlation," *IEEE Trans. Sig. Proc.*, vol. 61, no. 8, pp. 2009–2015, Apr. 2013.

[21] J. Q. Shi and T. Choi, *Gaussian Process Regression Analysis for Functional Data.* CRC Press, 2011.

[22] C. Rasmussen and C. Williams, *Gaussian Processes in Machine Learning.* MIT Press, 2006.

[23] P. Goovaerts, *Geostatistics for Natural Resources Evaluation.* Oxford University Press, 1997.

[24] E. H. Isaak and R. M. Srivastava, *An Introduction to Applied Statistics.* Oxford University Press, 1989.

[25] J. P. Chiles and P. Delfiner, *Geostatistics: Modeling Spatial Uncertainty.* John Wiley and Sons, 1999.

[26] W. F. Caselton and J. V. Zidek, "Optimal monitoring network designs," *Statistics and Probability Letters*, vol. 2, no. 4, pp. 223–227, 1984.

[27] N. A. C. Cressie, *Statistics for Spatial Data.* Wiley, 1991.

[28] N. Ramakrishnan, C. Bailey-Kellogg, S. Tadepalli, and V. N. Pandey, "Gaussian processes for active data mining of spatial aggregates," in *SIAM Data Mining*, 2005.

[29] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies," *J. Machine Learning Research*, vol. 9, pp. 235–284, 2008.

[30] R. Garnett, M. A. Osborne, and S. J. Roberts, "Bayesian optimization for sensor set selection," in *ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010, pp. 209–219.

[31] Y. Xu and J. Choi, "Adaptive sampling for learning Gaussian processes using mobile sensor networks," *Sensors*, vol. 11, no. 3, pp. 3051–3066, 2011.

[32] T. K. Moon, S. E. Budge, W. C. Stirling, and J. B. Thompson, "Epistemic decision theory applied to multiple-target tracking," *IEEE Trans. on Syst., Man, and Cyber.*, vol. 24, no. 2, pp. 234–245, Feb. 1994.

[33] W. C. Stirling and R. L. Frost, "Making value-laden decisions under conflict," in *IEEE Int. Conf. on Syst., Man, and Cyber.*, Oct. 1994, pp. 1559–1564.

[34] W. C. Stirling and D. R. Morrell, "Convex Bayes decision theory," *IEEE Trans. on Syst., Man, and Cyber.*, vol. 21, no. 1, pp. 173–183, 1991.

[35] W. C. Stirling, "Multi agent coordinated decision-making using epistemic utility theory," *Artificial Social Systems. Springer Berlin Heidelberg*, pp. 164–183, 1994.

[36] M. Al-Shoukairi and B. Rao, "Sparse Bayesian learning using approxiamte message passing," in *48th Asilomar Conf. on Sig., Syst. and Compt.*, Nov. 2014, pp. 1957–1961.

[37] M. Shekaramiz, T. Moon, and J. Gunther, "Hierarchical Bayesian approach for jointly-sparse solution of multiple-measurement vectors," in *48th Asilomar Conf. on Sig., Syst. and Compt.*, Nov. 2014, pp. 1962–1966.

[38] ——, "On the block-sparsity of multiple-measurement vectors," *Presented at the IEEE Sig. Proc. and SP Edu. Workshop, Utah, Aug.*, 2015.

[39] D. L. Donoho, "Compressed sensing," *IEEE Trans. Info. Th.*, vol. 52, no. 4, pp. 1289–1306, 2006.

[40] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Sig. Proc. Mag.*, vol. 25, no. 2, pp. 21–30, 2008.

[41] M. Elad, *Sparse and Redundant Representation: From Theory to Applications in Signal and Image Processing.* Springer, 2010.

[42] R. G. Baraniuk, "Compressive sensing," *IEEE Sig. Proc. Mag.*, vol. 24, no. 4, pp. 118–124, 2007.

[43] M. Mishali and Y. C. Eldar, "Reduce and boost: Recovering arbitrary sets of jointly sparse vectors," *IEEE Trans. Sig. Proc.*, vol. 56, no. 10, pp. 4692–4702, Oct. 2008.

[44] S. Chen and D. Donoho, "Basis pursuit," in *Asilomar Conf. of Signals, Systems, and Computers*, Oct. 1994, pp. 41–44.

[45] P. R. Gill, A. Wang, and A. Molnar, "The in-crowd algorithm for fast basis pursuit denoising," *IEEE Trans. Sig. Proc.*, vol. 59, no. 10, pp. 4595–4605, Oct. 2011.

[46] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Science*, vol. 2, no. 1, pp. 183–202, 2009.

[47] S. Becker, J. Bobin, and E. J. Candes, "NESTA: A fast and accurate first order method for sparse recovery," *SIAM J. Imaging Science*, vol. 4, no. 1, pp. 1–39, 2009.

[48] J. Zhang and B. Ghanem, "ISTA-NET: Iterative shrinkage-thresholding algorithm inspired deep network for image compressive sensing," *arXiv: 1706.07929*, 2017.

[49] D. Wipf and B. Rao, "Sparse Bayesian learning for basis pursuit selection," *IEEE Trans. Sig. proc.*, vol. 52, no. 8, pp. 2153–2164, Aug. 2004.

[50] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. Sig. Proc.*, vol. 56, no. 6, pp. 2346–2356, 2008.

[51] Z. Zhang and B. Rao, "Recovery of block sparse signals using the framework of block sparse Bayesian learning," in *IEEE Int. Conf. on Acoust., Speech and Sig. Proc. (ICASSP)*, 2012, pp. 3345–3348.

[52] J. Fang, Y. Shen, F. Li, H. Li, and Z. Chen, "Support knowledge-aided sparse Bayesian learning for compressed sensing," in *IEEE Int. Conf. on Acoust., Speech and Sig. Proc. (ICASSP)*, Apr. 2015, pp. 3786–3790.

[53] S. F. Cotter, B. D. Rao, K. Engan, and K. K. Delgado, "Sparse solutions to linear inverse problem with multiple measurement vectors," *IEEE Trans. Sig. Proc.*, vol. 53, no. 7, pp. 2477–2488, 2005.

[54] J. Chen and X. Huo, "Theoretical results on sparse representaions of multiple-measurement vectors," *IEEE Trans. Sig. Proc.*, vol. 54, no. 12, pp. 4634–4643, 2006.

[55] J. Ding, L. Chen, and Y. Gu, "Robustness of orthogonal matching pursuit for multiple measurement vectors in noisy scenario," in *IEEE Int. Conf. on Acoust., Speech and Sig. Proc. (ICASSP)*, 2012, pp. 3813–3816.

[56] M. M. Hyder and K. Mahata, "A robust algorithm for joint-sparse recovery," *IEEE Sig. Proc. Let.*, vol. 16, no. 12, pp. 1091–1094, 2009.

[57] M. Mishali and Y. C. Eldar, "Xampling: Signal acquisition and processing in union of subspaces," *IEEE Trans. Sig. Proc.*, vol. 59, no. 10, pp. 4719–4734, 2011.

[58] H. E. Kwon and B. Rao, "On the benefits of the block-sparsity structure in sparse signal recovery," in *IEEE Int. Conf. on Acoust., Speech and Sig. Proc. (ICASSP)*, 2012, pp. 3685–3688.

[59] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *J. of the Royal Statistical Society Series B*, vol. 67, no. 1, pp. 91–108, 2005.

[60] D. Hernandez-Lobato, J. M. Hernandez-Lobato, and P. Dupont, "Generalized spike-and-slab priors for Bayesian group feature selection using expectation propagation," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 1891–1945, 2013.

[61] I. Bilik, "Spatial compressive sensing for direction-of-arrival estimation of multiple sources using dynamic sensor arrays," *IEEE Trans. Aerospace and Elect. Syst.*, vol. 47, no. 3, pp. 1754–1769, 2011.

[62] Y. C. Eldar, P. Kuppinger, and H. Bolcskei, "Block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Trans. Sig. Proc.*, vol. 58, no. 6, pp. 3042–3054, 2010.

[63] J. Huang, T. Zhang, and D. Metaxas, "Learning with structured sparsity," *J. Mach. Learn. Res.*, vol. 12, pp. 3371–3412, 2011.

[64] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Mach. Learn. Soc. Ser. B*, vol. 68, no. 1, pp. 49–67, 2006.

[65] Y. Qi, D. Liu, L. Carin, and D. Dunson, "Multi-task compressive sensing with Dirichlet process priors," in *Int. Conf. on Mach. Learn. (ICML)*, 2008, pp. 768–775.

[66] S. Ji, D. Dunson, and L. Carin, "Multitask compressive sensing," *IEEE Trans. Sig. Proc.*, vol. 57, no. 1, pp. 92–106, 2009.

[67] M. Shekaramiz, T. K. Moon, and J. H. Gunther, "On the block-sparse solution of single measurement vectors," in *49th Asilomar Conf. of Signals, Systems, and Computers*, Nov. 2015, pp. 508–512.

[68] L. Yu, H. Sun, J. P. Barbot, and G. Zheng, "Bayesian compressive sensing for cluster structured sparse signals," *Signal Process.*, vol. 92, no. 1, pp. 259–269, 2012.

[69] M. R. Anderson, O. Winther, and L. K. Hansen, "Bayesian inference for structured spike and slab priors," in *Adv. Neural Inf. Process. Syst.*, 2014, pp. 1745–1753.

[70] X. Meng, S. Wu, L. Kuang, D. Huang, and J. Lu, "AMP-NNSPL," arXiv:1601.00543[cs.IT]/, 2016.

[71] J. Ziniel and P. Schniter, "Efficient high-dimensional inference in the multiple measurement vector problem," *IEEE Sig. Proc. Mag.*, vol. 61, no. 2, pp. 340–354, 2013.

[72] J. Fang, L. Zhang, and H. Li, "Two-dimensional pattern-coupled sparse Bayesian learning via generalized approximate message passing," *IEEE Trans. Image Proc.*, vol. 25, no. 6, pp. 2920–2930, 2016.

[73] J. P. Vila and P. Schniter, "Expectation-maximization Gaussian-mixture approximate message passing," *IEEE Trans. Sig. Proc.*, vol. 61, no. 19, pp. 4658–4672, 2013.

[74] C. M. Bishop, *Pattern Recognition And Machine Learning.* Springer, 2009.

[75] C. S. Grant, T. K. Moon, J. H. Gunther, M. R. Stites, and G. P. Williams, "Detection of amorphously shaped objects using spatial information detection enhancement (SIDE)," *IEEE J. Sel. Topics in Appl. Earth Observ. Remote Sens.*, vol. 5, no. 2, pp. 478–487, 2012.

[76] Y. C. Pati, R. Rezaifar, and P. S. Krishnaprasad, “Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition,” in *27th Asilomar Conf. of Signals, Systems, and Computers*, Nov. 1993, pp. 40–44.

[77] E. V. D. Berg and M. P. Friedlander, “SPGL1,” https://www.math.ucdavis.edu/~mpf/spgl1/, 2015.

[78] S. Ji, “MT-CS,” http://people.ee.duke.edu/~lcarin/BCS.html, 2007.

[79] J. Fang and Y. Shen, “PC-SBL,” http://junfang-uestc.net/swf/publication.html, 2015.

[80] Z. Zhang, “Materials: Matlab-Codes,” https://sites.google.com/site/researchbyzhang/software/, 2011.

[81] L. Yu, “Materials: Matlab-Codes,” https://sites.google.com/site/link2yulei/publications/materials/, 2010.

[82] Z. Zhang and B. D. Rao, “Sparse signal recovery with temporally correlated source vectors using sparse Bayesian learning,” *IEEE J. of Selected Topics in Sig. Proc.*, vol. 5, no. 5, pp. 912–926, 2011.

[83] ——, “Clarify some issues on the sparse Bayesian learning for sparse signal recovery,” Univ. of Cal., San Diego, Technical Report, Sep., 2011.

[84] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to documnet recognition,” *Proc. of IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[85] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, “Compressed sensing using generative models,” *arXiv perprint arXiv: 1703.03208*, 2017.

[86] E. W. Tramel, A. Dremeau, and F. Krzakala, “Approximate message passing with restricted Boltzmann machine priors,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2016, no. 7, pp. 1–15, 2016.

[87] H. S. Mousavi, V. Monga, and T. D. Tran, “Iterative convex refinement for sparse recovery,” *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1903–1907, 2015.

[88] M. Shekaramiz, T. K. Moon, and J. H. Gunther, “AMP-B-SBL: An algorthm for clustered sparse signals using approximate message passing,” in *IEEE Uniquitous Computing, Electronics and Mobile Communication (UEMCON)*, Oct. 2016, pp. 1–5.

[89] H. Palangi, R. Ward, and L. Deng, “Distributed compressive sensing: A deep learning approach,” *IEEE Trans. Signal Processing*, vol. 64, no. 17, pp. 4504–4518, 2016.

[90] J. H. Chang, C. L. Li, B. Poczos, B. V. V. Kumar, and A. C. Sankaranarayanan, “One network to solve them all- solving linear inverse problems using deep projection models,” *arXive preprint*, 2017.

[91] T. H. Vu, H. S. Mousavi, and V. Monga, “Adaptive matching pursuit for sparse signal recovery,” in *IEEE Int. Conf. on Acoust., Speech and Sig. Proc. (ICASSP)*, Mar. 2017, pp. 4331–4335.

[92] A. Gelman and D. B. Rubin, "Inference from iterative simulation using multiple sequences," *Statistical Sci.*, vol. 7, no. 4, pp. 457–511, 1992.

[93] S. P. Brooks and A. Gelman, "General methods for monitoring convegence of iterative simulations," *J. Comput. Graphical Stat.*, vol. 7, no. 4, pp. 434–455, 1998.

[94] R. E. Kass, B. P. Carlin, A. Gelman, and R. M. Neal, "Markov chain Monte Carlo in practice: A roundtable discussion," *The American Statistical Association*, vol. 52, no. 2, pp. 93–100, 1998.

[95] M. Elad and I. Yavneh, "A plurality of sparse representations is better than the sparsest one alone," *IEEE Trans. Info. Theory*, vol. 55, no. 10, pp. 4701–4714, Oct. 2009.

[96] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. motivations and construction," in *Proc. Inform. Theory Workshop, (Cairo, Egypt)*, Jan. 2010, pp. 1–5.

[97] J. Ziniel and P. Schniter, "Dynamic compressive sensing of time-varying signals via approximate message passing," *IEEE Trans. Sig. Proc.*, vol. 61, no. 21, pp. 5270–5284, Nov. 2013.

[98] P. Schniter, "Turbo reconstruction of structured sparse signals," in *Conf. Inf. Sci. Syst., Princton, NJ*, Mar. 2010, pp. 1–5.

[99] Z. Zhang, "SBL algorithms matlab code," https://sites.google.com/site/researchbyzhang/software, accessed Sep. 21, 2015.

[100] J. A. Luo, X. P. Zhang, and Z. Wang, "Direction-of-arrival estimation using sparse variable projection optimization," in *Proc. Int. Symposium on Circs. and Syst. (ISCAS)*, May 2012, pp. 3106–3109.

[101] J. M. Kim, O. K. Lee, and J. C. Ye, "Dynamic sparse support tracking with multiple measurement vectors using compressive music," in *IEEE Int. Conf. on Acoust., Speech and Sig. Proc. (ICASSP)*, 2012, pp. 2717–2720.

[102] Z. Zhang and B. D. Rao, "Exploiting correlation in sparse signal recovery problems: Multiple measurement vectors, block sparsity, and time-varying sparsity," in *ICMLâĂŹ11 Workshop Structured Sparsity: Learning and Inference, Washington, DC*, 2011.

[103] N. Vaswani and W. Lu, "Modified-CS: Modifying compressive sensing for problems with partially known support," *IEEE Trans. Sig. Proc.*, vol. 58, no. 9, pp. 4595–4607, 2010.

[104] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressed channel sensing: A new approach to estimating sparse multipath channels," *Proc. IEEE*, vol. 98, no. 6, pp. 1058–1076, 2010.

[105] M. A. Herman and T. Strohmer, "High-resolution radar via compressed sensing," *IEEE Trans. on Sig. Proc.*, vol. 57, no. 6, pp. 2275–2284, 2009.

[106] H. Huang, S. Misra, W. Tang, H. Barani, and H. Al-Azzawi, "Applications of compressed sensing in communication networks," *arXiv:1305.3002v3*, 2014.

[107] G. Kutynoik, "Theory and applications of compressed sensing," *GAMM-Mitt*, vol. 36, pp. 79–101, 2013.

[108] S. K. Sharma, S. Chatzinotas, and B. Ottersten, "Compressive sparsity order estimation for wideband cognitive radio receiver," in *IEEE Int. Conf. on Communications (ICC)*, 2014, pp. 1361–1366.

[109] K. Chang, P. Ding, and B. Li, "Compressive sensing reconstruction of correlated images using joint regularization," *IEEE Signal Processing Letters*, vol. 23, no. 4, pp. 449–453, 2016.

[110] U. L. Wijewardhana, M. Codreanu, and M. Latva-aho, "Bayesian method for image recovery from block compressive sensing," in *50th Asilomar Conf. of Sig., Syst., and Compt.*, 2016, pp. 379–383.

[111] S. Qaisar, R. M. Bilal, W. Iqbal, M. Naureen, and S. Lee, "Compressive sensing: From theory to applications, a survey," *Commun. Netw. J.*, vol. 15, no. 5, pp. 443–456, 2013.

[112] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. on Sig. Proc.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[113] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressive sensnig," *Appl. Computat. Harmon. Anal.*, vol. 27, no. 3, pp. 265–274, 2009.

[114] L. Yu, C. Wei, J. Jia, and H. Sun, "Compressive sensing for cluster structured sparse signals: Variational Bayes approach," *IET Signal Process.*, vol. 10, no. 7, pp. 770–779, 2016.

[115] L. Stankoviċ, M. Dakoviċ, and S. Vujoviċ, "Adaptive variable step algorithm for missing samples recovery in sparse signals," *IET Signal Process.*, 2014.

[116] S. Babacan, R. Molina, and A. Katsaggelos, "Bayesian compressive sensing using Laplace priors," *IEEE Trans. image Proc.*, vol. 19, no. 1, pp. 53–63, 2010.

[117] L. He and L. Carin, "Exploiting structure in wavelet-based Bayesian compressive sensing," *IEEE Trans. Sig. Proc.*, vol. 57, no. 9, pp. 3488–3497, 2009.

[118] J. Vila and P. Schniter, "Expectation-maximization Bernoulli-Gaussian approximate message passing," in *45th Asilomar Conf. of Sig., Syst., and Compt.*, 2011, pp. 799–803.

[119] M. Shekaramiz, T. K. Moon, and J. H. Gunther, "AMP-B-SBL: An algorithm for clustered sparse signals using approximate message passing," in *7th Annual Ubiq. Compt., Elect. and Mob. Comm. Conf. (UEMCON)*, 2016, pp. 1–5.

[120] ——, "Sparse Bayesian learning using variational Bayes inference based on a greedy criterion," in *51th Asilomar Conf. on Sig., Syst. and Comp.*, 2017.

[121] M. Al.-Shoukairi, P. Schniter, and B. D. Rao, "A GAMP-based low complexity sparse Bayesian learning algorithm," *IEEE Trans. on Sig. Proc.*, vol. 66, no. 2, pp. 294–308, 2018.

[122] L. He, H. Chen, and L. Carin, "Tree-structured compressive sensing with variational Bayesian analysis," *IEEE Sig. Proc. Let.*, vol. 17, no. 3, pp. 233–236, 2010.

[123] S. D. Babacan, S. Nakajima, and M. N. Do, "Bayesian group-sparse modeling and variational inference," *IEEE Trans. on Sig. Proc.*, vol. 62, no. 11, pp. 2906–2921, 2014.

[124] M. Beal, *Variational Algorithms for Approximate Bayesian Inference.* PhD dissertation, University College London, 2003.

[125] D. G. Tzikas, A. C. Likas, and N. P. Galatsanos, "The variational approximation for Bayesian inference," *IEEE Sig. Proc. Mag.*, 2008.

[126] C. Fox and S. Roberts, "A tutorial on variational Bayesian inference," *Artificial Intelligence Review*, vol. 38, no. 2, pp. 85–95, 2011.

[127] I. A. Antonov and V. M. Saleev, "An economic mehtod of computing LP_ τ-sequences," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, vol. 19, no. 1, pp. 243–245, 1979.

[128] S. Joe and F. Y. Kuo, "Notes on generating Sobol sequences," *ACM Trans. Math. Softw.*, vol. 29, no. 1, pp. 57–57, 2008.

[129] L. Han, C. Li, and Y. Zeng, "Study on satellite orbit design for boundary environment monitoring," in *International Conference on Remote Sensing, Environment and Transportation Engineering (RSETE), Nanjing*, 2011, pp. 783–786.

[130] Y. Ulybyshev, "Satellite constellation design for complex coverage," *J. Spacecraft Rockets*, vol. 45, no. 4, pp. 843–849, 2009.

[131] R. Píriz, B. M. Peiró, and M. R. Merino, "The Galileo constellation design: A systematic approach," in *18th International Technical Meeting of the Satellite Division*, Sep. 2005, pp. 1296–1306.

[132] T. W. Beech, S. Cornara, M. B. Mora, and G. D. Lecohier, "A study of three satellite constellation design algorithms," in *14th International Symposium on Space Flight Dynamics*, Feb. 1999.

[133] T. J. Lang, "Symmetric circular-orbit satellite constellations for continuous global coverage," in *AAS/AIAA Astrodynamics Specialist*, 1987, pp. 1111–1132.

[134] W. S. Adams and L. Rider, "Circular polar constellations providing continuous single or multiple coverage above a specified latitude," *Journal of Astronautical Sciences*, vol. 35, pp. 155–192, May 1987.

[135] D. C. Beste, "Design of satellite constellations for optimal continuous coverage," *IEEE Trans. on Aerospace and Elec. Syst.*, vol. 14, no. 3, pp. 466–473, May 1978.

[136] J. G. Walker, "Continuous whole-Earth coverage by circular-orbit satellites pattern," in *Royal Aircraft Establishment, Technical Report 77044*, 1977.

[137] O. L. de Weck, U. Scialom, and A. Siddiqi, "Optimal reconfiguration of satellite constellations with the auction algorithm," *ACATA Astronautica*, vol. 62, pp. 112–130, 2008.

[138] O. L. de Weck, R. de Neufville, and M. Chaize, "Staged deployment of communications satellite constellations in low Earth orbit," *Journal of Aerospace Computing, Information and Communication (JACIC)*, vol. 1, no. 3, pp. 119–136, Mar. 2004.

[139] S. W. Paek, *Reconfigurable Satellite Constellations for Geo-Spatially Adaptive Earth Observation Missions.* M.Sc dissertation, Massachusetts Institute of Technology, 2012.

[140] D. Pearson, S. U. Pillay, and Y. Lee, "An algorithm for near-optimal placement of sensor elements," *IEEE Trans. Information Theory*, vol. 36, no. 6, pp. 1280–1284, 1990.

[141] H. Zhang, "Optimal sensor placement," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1992, pp. 825–830.

[142] E. Marchand and F. Chaumette, "Active sensor placement for complete scene reconstruction and exploration," in *Proc. of 1997 IEEE Int. Conf. on Robotics and Automation*, 1997, pp. 743–750.

[143] N. Bartolini, T. Calamoneri, T. L. Porta, and S. Silvestri, "Mobile sensor deployment in unknown fields," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–5.

[144] B. Sharif and F. Kamalabadi, "Optimal sensor array configuration in remote image formation," *IEEE Trans. Image Proc.*, vol. 17, no. 2, pp. 155–166, Feb. 2008.

[145] T. Imbiriba, J. C. M. Bermudez, J.-Y. Tourneret, and C. Richard, "Detection of nonlinear mixtures using Gaussian processes: Application to hyperspectral imaging," in *IEEE Int. Conf. on Acoust., Speech and Sig. Proc. (ICASSP)*, May 2014, pp. 7949–7953.

[146] T. Wu and Y. Li, "Spatial interpolation of temperature in the United States using residual kriging," *Applied Geography*, vol. 44, pp. 112–120, 2013.

[147] A. Ranganathan, M. H. Yang, and J. Ho, "Online sparse Gaussian process regression and its applications," *IEEE Trans. on Image Proc.*, vol. 20, no. 2, pp. 391–404, Feb. 2011.

[148] Z. Xing, M. Zhou, A. Castrodad, G. Sapiro, and L. Carin, "Dictionary learning for noisy and incomplete hyperspectral images," *SIAM J. Imag. Sci.*, vol. 5, no. 1, pp. 33–56, 2012.

[149] P. Monestiez, D. Courault, D. Allard, and F. Ruget, "Spatial interpolation of air temperature using evironmental context: Application to a crop model," *Env. and Ecol. Stat.*, vol. 8, pp. 297–309, 2001.

[150] M. R. Holdaway, "Spatial modeling and interpolation of monthly temperature using kriging," *Climate Research*, vol. 6, pp. 215–225, 1996.

[151] M. A. Goodrich, W. C. Stirling, and R. L. Frost, "A theory of satisficing decisions and control," *IEEE Trans. on Syst., Man, and Cyber. Part A: Systems and Humans*, vol. 28, no. 6, pp. 763–779, 1998.

[152] K. P. Murphy, *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[153] A. J. Storkey, "Truncated covariance matrices and Toeplitz methods in Gaussian processes," in *Artificial Neural Networks (ICANN)*, 1999.

[154] A. J. Smola and P. L. Bartlett, "Sparse greedy Gaussian process regression," *Advances in Neural Information Processing Systems, Cambridge, Massachussetts, MIT Press*, vol. 13, pp. 619–625, 2001.

[155] J. Quinonero-Candela and C. E. Rasmussen, "Analysis of some methods for reduced rank Gaussian process regression," in *Switching and Learning in Feedback Systems, Springer, Berlin, Heidelberg*, 2005, pp. 98–127.

[156] I. Levi, *The Enterprise of Knowledge; An Essay on Knowledge, Credal Probability, and Chance.* MIT Press, 1980.

[157] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on Inf. Theory*, vol. 53, no. 2, pp. 4655–4666, 2007.

[158] M. M. Abdel-Sayed, A. Khattab, and M. F. Abu-Elyazeed, "RMP: Reduced-set matching pursuit approach for efficient compressed sensing signal reconstruction," *J. Advanced Research*, vol. 7, no. 6, pp. 851–861, 2016.

[159] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach," *IEEE Sig. Proc. Letters*, vol. 9, no. 4, pp. 137–140, 2002.

[160] S. Rangan and A. K. Fletcher, "Orthogonal matching pursuit from noisy random measurements: A new analysis," in *Advances in Neural Info. Proc. Syst.*, 2009, pp. 540–548.

[161] T. T. Cai and L. Wang, "Orthogonal matching pursuit for sparse signal recovery with noise," *IEEE Trans. on Inf. Theory*, vol. 57, no. 7, pp. 4680–4688, 2011.

[162] A. K. Fletcher and S. Rangan, "Orthogonal matching pursuit: A Brownian motion analysis," *IEEE Trans. on Sig. Proc.*, vol. 63, no. 3, pp. 1010–1021, 2012.

[163] J. Wen, Z. Zhou, J. Wang, X. Tang, and Q. Mo, "A sharp condition for exact support recovery with orthogonal matching pursuit," *IEEE Trans. on Sig. Proc.*, vol. 56, no. 6, pp. 1370–1382, 2017.

[164] S. Kallummil and S. Kalyani, "Tuning free orthogonal matching pursuit," *arXiv:1703.05080*, 2017.

[165] M. Lopes, "Estimating unknown sparsity in compressed sensing," in *Int. Conf. on Machine Learning*, 2013, pp. 217–225.

[166] S. K. Ambat, S. Chatterjee, and K. V. S. Hari, "Fusion of greedy pursuits for compressed sensing signal reconstruction," in *20th IEEE Euro. Sig. Proc. Conf. (EUSIPCO)*, 2012, pp. 1434–1438.

[167] M. Shekaramiz, T. Moon, and J. Gunther, "Exploration and data refinement via multiple mobile sensors," in *51th Asilomar Conf. on Sig., Syst. and Compt.*, Oct. 2017, pp. 885–889.

[168] A. Banerjee, D. B. Dunson, and S. T. Tokdar, "Efficient Gaussian process regression for large datasets," *Biometrica*, vol. 100, no. 1, pp. 75–89, 2012.

[169] J. Schreiter, D. Nguyen-Tuong, and M. Toussaint, "Efficient sparsification for Gaussian process regression," *Neurocomputing*, vol. 192, pp. 29–37, 2016.

[170] E. L. Snelson, *Flexible and Efficinet Gaussian Process Models for Machine Learning.* PhD dissertation, University College London (UCL), 2007.

[171] J. Hensman, N. Fusi, and N. Lawrence, "Gaussian processes for big data," in *29th Conf. Uncert. Artif. Intel.*, 2013, pp. 282–290.

CURRICULUM VITAE

Mohammad Shekaramiz

Personal Professional Profiles

- Google Scholar: hhttps://scholar.google.com/citations?user=mQfYi4QAAAAJ&hl=en
- Researchgate: https://www.researchgate.net/profile/Mohammad_Shekaramiz
- Linkedin: https://www.linkedin.com/in/mohammad-shekaramiz/
- Meritpage Profile: https://meritpages.com/MohammadShekaramiz

Awards

- Awarded Graduate Student Dissertation Fellowship generously provided by Research and Graduate Studies (RGS) of Utah State University in the amount of $5000, June 2018.
- Outstanding graduate poster presentation (Engineering Discipline), Students Research Symposium (SRS), Utah State University, Apr. 13, 2017.
  Link: https://rgsawards.usu.edu/student-awards/srs-winners/
- Awarded travel grant, generously provided by Research and Graduate Studies (RGS) of Utah State University in the amount of $300 to attend and present conference papers (Nov. 2015, Oct. 2016, and Nov. 2017).
- Awarded PhD student travel grant, generously provided by the College of Engineering of Utah State University in the amount of $250 to attend and present conference papers (Nov. 2014 and Oct., 2016).

- Awarded a scholarship in the amount of $300 generously provided by Golden Key International Honor Society-USU chapter, May 2016.

Membership in Professional Societies

- Member, Golden Key International Honor Society, since Sept. 2013
- Student Member, Institute of Electrical and Electronics Engineers (IEEE), since Mar. 2015
- Student Member, IEEE Young Professionals, since Nov. 2015
- Member, Honor Society of Phi Kappa Phi, since Oct. 2015
- Member, Institute of Research Engineers and Doctors, since Jan. 2018
- Member, Honor Society of Tau Beta Pi, since Mar. 2018

Technical Paper Review Experience

- Reviewer for Journal of Intelligent and Fuzzy Systems (JIFS), since Jan. 2013
- Reviewer for Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2016-2019)
- Reviewer on the technical program committee for the 2016 IEEE Workshop on Statistical Signal Processing (SSP 2016 and SSP 2018)
- Invited reviewer for IEEE Transactions on Aerospace and Electronic Systems (TAES) in 2017 and 2018
- Invited reviewer for Elsevier Signal Processing 2018

Journal Editorial Board Experience

- Journal of Autonomous Intelligence, since Jun. 2018
- Progress in Human Computer Interaction, since May 2018

Teaching Experience

- Instructor: Detection and Estimation Theory (course # ECE 7030), ECE Dept., Utah State University (USU), Logan, Utah (Fall 2016)
- Lecturer: Electrical Engineering Dept., Islamic Azad Univ. of Shahr Ray, Tehran, Iran (Sept. 2009- Aug. 2012) Instructed Courses: Electronic Circuits, Electronics, Digital Logic Circuits, Pulse Techniques, Operational Amplifiers, and Special Topics in Electrical Eng.
- Instructed Laboratories: Electronics (I) Lab., Digital Electronics Lab., and Pulse Technique Lab. Project Supervisor: Supervised 18 Senior Projects
- Teaching Assistant: Continuous-Time Systems and Signals (course # ECE 3620), Under the Supervision of Dr. Zeljko Pantic, Utah State University (USU), Logan, Utah (Spring 2018)
- Teaching Assistant: Control Systems (course # ECE 5310), Under the Supervision of Dr. Cripps, USU (Fall 2018)
- Teaching Assistant: Mathematical methods and algorithms for signal processing (course # ECE 6030), Under the Supervision of Prof. Todd Moon, Utah State University (USU), Logan, Utah (Spring 2015)
- Teaching Assistant: Mechatronics (course # ECE 5320), Under the Supervision of Dr. Cripps, USU (Spring 2013)
- Teaching Assistant: Control Systems (course # ECE 5310), Under the Supervision of Dr. Cripps, USU (Fall 2012)
- Teaching Assistant: Digital Control Course, Isfahan Univ. of Tech., Isfahan, Iran (Fall 2005, Fall 2006)

Publications and Presentations

- Accepted Papers
  - M. Shekaramiz, T. K. Moon, and J. H. Gunther, *Sparse Recovery via Variational Bayesian Inference: Comparing Bernoullis-Gaussians-Inverse Gamma and Gaussians-Inverse Gammas Modeling*, in Proceedings of 52th Asilomar Conf. on Sig., Syst., and Compt. (ACSSC), IEEE, Cal., USA, to be presented on Oct., 2018.
  - M. Shekaramiz, T. K. Moon, and J. H. Gunther, *Sparse Bayesian Learning Using Variational Bayes Inference Based on a Greedy Criterion*, in Proceedings of 51th Asilomar Conf. on Sig., Syst., and Compt. (ACSSC), IEEE, Cal., USA, pp. 858-862, 29 Oct.-1 Nov., 2017.
  - M. Shekaramiz, T. K. Moon, and J. H. Gunther, *Exploration and Data Refinement via Multiple Mobile Sensors Based on Gaussian Processes*, in Proceedings of 51th Asilomar Conf. on Sig., Syst., and Compt. (ACSSC), IEEE, pp. 885-889, Cal., USA, 29 Oct.-1 Nov., 2017.
  - M. Shekaramiz, T. K. Moon, and J. H. Gunther, *Clustered Pattern Sparse Signal Recovery Using Hierarchical Bayesian Learning*, in Proceedings of 42th IEEE Int. Conf. on Acoust, Speech and Sig. Proc. (ICASSP), PhD Forum, New Orleans, LA, USA, pp. 1-2, 5-9 Mar.,2017,
    Link:
    https://pdfs.semanticscholar.org/1f9c/8ec1d5399e67750e4948163005903ec011ac.pdf
  - M. Shekaramiz, T. K. Moon, and J. H. Gunther, *Sparse Bayesian Learning Boosted by Partial Erroneous Support Knowledge*, in Proceedings of 50th Asilomar Conf. on Sig., Syst., and Compt. (ACSSC), IEEE, Cal., USA, pp. 389-393, 6-9 Nov., 2016.

- M. Shekaramiz, T. K. Moon, and J. H. Gunther, *AMP-B-SBL: An Algorithm for Clustered Sparse Signals Using Approximate Message Passing*, in Proceedings of IEEE 7th Annual Ubiquitous Computing, Electronics and Mobile Communication (UEMCON), NY, USA, pp. 1-5, 20-22 Oct., 2016.
- M. Shekaramiz and F. Sheikholeslam, *Stability Analysis for Continuous T-S Fuzzy Models Having Uncertainties: A Systematic Approach*, Bulletin de la Societe Royale des Sci. de Liege, Vol. 85 (1), pp. 96-105, Jun., 2016.
- M. Shekaramiz, T. K. Moon, and J. H. Gunther, *On the Block-Sparse Solution of Single Measurement Vectors*, in Proceedings of 49th Asilomar Conf. on Sig., Syst., and Compt. (ACSSC), IEEE, Cal., USA, pp. 508-512, 8-11 Nov., 2015.
- M. Shekaramiz, T. K. Moon, and J. H. Gunther, *the Block-Sparsity of Multiple-Measurement Vectors*, in Proceedings of IEEE Sig. Proc. and Sig. Proc. Edu. Workshop (SP/SPE), Snow bird, UT., USA, pp. 220-225, 9-12 Aug., 2015.
- M. Shekaramiz, T. K. Moon, and J. H. Gunther, *Bayesian Approach for Jointly-Sparse Solution of Multiple-Measurement Vectors*, in Proceedings of 48th Asilomar Conf. on Sig., Syst., and Compt. (ACSSC), IEEE, Cal., USA, pp. 1962-1966, 2-5 Nov., 2014.
- M. Shekaramiz and F. Sheikholeslam, *On the Stability of Continuous-Time T-S Model*, in Proceedings of 8th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), IEEE, Shanghai, China, pp. 226-230, 26-28 July, 2011.
- F. Sheikholeslam and M. Shekaramiz, *Criterion for Takagi-Sugeno Models*, in Proceedings of 8th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), IEEE, Shanghai, China, pp. 262-267, 26-28 July, 2011.

- Symposium Poster Presentations
  - M. Shekaramiz, T. K. Moon, and J. H. Gunther, *New Adaptive Decision Maker Framework for the Exploration in an Unknown Region Using Multiple Mobile Sensors*, Student Research Symposium (SRS), Utah State University, Apr., 2018.
    Link:
    https://digitalcommons.usu.edu/researchweek/ResearchWeek2018/All2018/21/
  - M. Shekaramiz, T. K. Moon, and J. H. Gunther, *Bayesian Compressive Sensing Algorithms for Sparse Signal Recovery*, Student Research Symposium (SRS), Utah State University, Apr., 2017.
    Link:
    https://digitalcommons.usu.edu/researchweek/ResearchWeek2017/Session4Poster/5/
  - M. Shekaramiz, T. K. Moon, and J. H. Gunther, *CAMP: An Algorithm to Recover Sparse Signals with Unknown Clustering Pattern Using Approximate Message Passing*, Student Research Symposium (SRS), Utah State University, Apr., 2016.
    Link:
    https://digitalcommons.usu.edu/researchweek/ResearchWeek2016/AllResearchWeek2016/25/
  - M. Shekaramiz, T. K. Moon, and J. H. Gunther, *Sparse Recovery with Unknown Sparsity Pattern via Multiple Measurement Vectors*, Student Research Symposium (SRS), Utah State University, Apr., 2016.
    Link:
    https://digitalcommons.usu.edu/researchweek/ResearchWeek2016/AllResearchWeek2016/24/