Bayesian Hypothesis Testing for Sparse Representation

H. Zayyani[1], M. Babaie-Zadeh[1] *Member* and C. Jutten[2] *Fellow*

***Abstract*— In this paper, we propose a Bayesian Hypothesis Testing Algorithm (BHTA) for sparse representation. It uses the Bayesian framework to determine active atoms in sparse representation of a signal.**

The Bayesian hypothesis testing based on three assumptions, determines the active atoms from the correlations and leads to the activity measure as proposed in Iterative Detection Estimation (IDE) algorithm. In fact, IDE uses an arbitrary decreasing sequence of thresholds while the proposed algorithm is based on a sequence which derived from hypothesis testing. So, Bayesian hypothesis testing framework leads to an improved version of the IDE algorithm.

The simulations show that Hard-version of our suggested algorithm achieves one of the best results in terms of estimation accuracy among the algorithms which have been implemented in our simulations, while it has the greatest complexity in terms of simulation time.

***Index Terms*-Sparse representation, Compressed sensing, Sparse component analysis, Blind source separation, Bayesian approaches, Pursuit algorithms.**

I. Introduction

Finding (sufficiently) sparse solutions of underdetermined systems of linear equations (possibly in the noisy case) has been used extensively in signal processing community. This problem has found applications in a wide range of diverse fields. Some applications are Blind Source Separation (BSS) and Sparse Component Analysis (SCA) [1], [2], decoding [3], image de-noising [4], sampling and signal acquisition (compressed sensing) [5], [6] and regression [7].

The problem can be stated in various contexts such as sparse representation, SCA or Compressed Sensing (CS). Here, we use the notation of sparse representation of signals. Let the model be:

$$\mathbf{x} = \mathbf{\Phi}\mathbf{y} + \mathbf{e}. \tag{1}$$

where $\mathbf{x}$ is an $n \times 1$ signal vector, $\mathbf{y}$ is an $m \times 1$ sparse coefficient vector, $\mathbf{\Phi}$ is an $n \times m$ matrix called dictionary and $\mathbf{e}$ is a $n \times 1$ error vector. It is assumed that $n < m$ which means that the signal length is smaller than the number of columns of the dictionary (which are called atoms [8]). So, the number of columns of the dictionary is more than the number of rows of the dictionary, that is, the dictionary is overcomplete. The main assumption is that the signal has a sparse representation in this overcomplete dictionary. The main goal is to find the sparse coefficient vector $\mathbf{y}$ based on the signal $\mathbf{x}$ and knowing the dictionary $\mathbf{\Phi}$. This problem is nominated as sparse representation of the signal and the methods are called sparse representation or sparse recovery algorithms.

According to applications, the vector interpretations are different, but in all of them the model follows (1). For example, in the context of CS, $\mathbf{\Phi}$ is the measurement matrix, $\mathbf{x}$ is a vector whose the few components are measurements of the signal and $\mathbf{y}$ is the sparse representation of the true signal. In the context of SCA, $\mathbf{\Phi}$ is the mixing matrix, $\mathbf{x}$ is the mixture vector and $\mathbf{y}$ is the source vector.

Because of $n < m$, there are usually infinitely many solutions of this underdetermined system of linear equations. In the exact sparse representation case, if we restrict ourselves to sufficiently sparse coefficient vectors, it is proved that under some conditions the sparsest solution is unique [9], [10], [11]. In the noisy case, there are theoretical guarantees (in terms of sparse coefficients and dictionary) for accurately and efficiently solving the problem [12], [13].

Finding the sparsest solution, that is, the solution with the minimum number of nonzero elements, is an NP-hard combinatorial problem. Different methods have been proposed to solve the problem in a tractable way. Most of them can be divided in two main categories: 1) Optimization approaches and 2) Greedy approaches (or pursuit algorithms). The first category solves the problem by optimizing a cost function according to different methods. The second set of methods tries to find active coefficients (with nonzero elements) directly through an algorithm.

The optimization approaches are basically split into convex and non-convex optimization methods. The most successful approach which is Basis Pursuit (BP) [14], suggests a convexification of the problem by replacing the ℓ^0-norm[1] with the ℓ^1-norm. It can then be implemented by Linear Programming (LP) methods. Recently, a Gradient-Projection algorithm for Sparse Reconstruction (GPSR) is used for bound-constrained quadratic programming formulation of these problems [15]. A method for large scale ℓ^1-Regularized Least Square (ℓ^1-RLS) is also devised in [16]. In addition, an Iterative Bayesian Algorithm (IBA) is used for solving the problem with a convex cost function which its steps resemble E-step and M-step of an EM algorithm [17], [18].

[1]Electrical engineering department and Advanced Communication Research Institute (ACRI), Sharif university of technology, Tehran, Iran.

[2]GIPSA-lab, Grenoble, and Institut Universitaire de France, France.

* This work has been partially funded by Iran NSF (INSF) under contract number 86/994, by Iran Telecom Research Center (ITRC), and also by center for International Research and Collaboration (ISMO) and French embassy in Tehran in the framework of a GundiShapour collaboration program.

[1]ℓ^0 norm of a vector is defined as the number of its non-zero components. Although it is not a mathematical norm, we use this name because it is frequently used in the literature.

Among the nonconvex cost function methods, the FOCUSS algorithm uses ℓ^p-norm with $p \leq 1$ instead of ℓ^0-norm in the noise-free case [9], [19]. Regularized-FOCUSS (R-FOCUSS) method extends FOCUSS for the noisy case with a Bayesian framework [20]. There are also some Bayesian methods such as Relevance Vector Machine (RVM) [21], Sparse Bayesian Learning (SBL) [22] and recently a Bayesian Compressive Sensing approach (BCS) [23] which mainly solve a nonconvex problem. Recently, a smoothed version of the ℓ^0-norm was used for solving the problem by a gradient-ascent method which is called Smoothed-ℓ^0(SL0) [24]. Moreover, a Sparse Reconstruction by Separable Approximation (SpaRSA) algorithm is suggested for group separable regularizers which is usually nonsmooth and possibly also nonconvex [25]. There is also an Iterative Reweighted Algorithm for nonconvex CS (IRA) [26].

The other category is the greedy algorithms which choose successively the active coefficients without having any explicit cost function. Generally, they use the correlation between the signal (or residual signal) and the atoms of the dictionary as an informative measure for deciding which coefficients are actually active (or nonzero). These algorithms are Matching Pursuit (MP) [8], Orthogonal Matching Pursuit (OMP) [27], Stage-wise OMP (StOMP) [28], Weighted MP (WMP) [29], Tree-Based Pursuit (TBP) [30], Regularized OMP (ROMP) [31], Gradient Pursuit (GP) [32], Stagewise weak Gradient Pursuit (StGP) [33] and Compressive Sampling MP (CoSaMP) [34].

Besides these two main approaches, one can mention Iterative THresholding algorithms (ITH) [35], an Iterative Detection Estimation (IDE) method [36] and three Minimum Mean Square Estimation (MMSE) algorithms [7], [37] which can be considered as Bayesian approaches which use discrete search techniques for finding the dominant posteriors. In [7], an algorithm is proposed for Approximating the MMSE estimate (A-MMSE) for the sparse vector in the application of linear regression. [37] also presents a Fast Bayesian Matching Pursuit (FBMP) method for recursive MMSE estimation in linear regression models.

Table I shows the overall sparse representation algorithms that we have mentioned. In this table, Bayesian methods are highlighted with bold characters. One can consider the Bayesian methods as a distinct category, but we did not do that because they also need some kind of cost function and optimization techniques or algorithms to solve their problem.

An important task in the sparse representation is to determine which coefficients are nonzero or in other words which atoms are active in the sparse representation of the signal. This is mainly done with Correlation Maximization (CM) in the pursuit algorithms with some differences. So, the core idea of the pursuit algorithms is to use the correlation of the residual signal with the atoms to determine the active atoms. For example, MP uses the CM to select at each iteration one active atom. StOMP uses a thresholding to select several active atoms at a same time. Other methods like IDE, use a measure of activity to determine the corresponding nonzero coefficients. The IBA algorithm [17] uses a steepest-ascent to determine a vector which is defined as the activity vector.

TABLE I
SPARSE REPRESENTATION ALGORITHMS (BOLD NAMES ARE BAYESIAN APPROACHES).

Optimization algorithms		Greedy	Other
Convex	Nonconvex	Algorithms	Algorithms
BP [14]	FOCUSS [9], [19]	MP [8]	ITH [35]
GPSR [15]	**R-FOCUSS** [20]	OMP [27]	IDE [36]
ℓ^1-RLS [16]	**RVM** [21]	StOMP [28]	**A-MMSE** [7]
IBA [17]	**SBL** [22]	**WMP** [29]	**FBMP** [37]
	BCS [23]	TBP [30]	
	SL0 [24]	ROMP [31]	
	SpaRSA [25]	GP [32]	
	IRA [26]	StGP [33]	
		CoSaMP [34]	

The simplicity of the greedy algorithms or pursuit algorithms arises in determining one active atom (e.g., in MP) or several active atoms (e.g., in StOMP) at an instant. So, they determine the activity vector in a simple way rather than to solve a hard optimization problem in a multi-dimensional space. The basic idea of this paper is to use the correlation between the signal and atoms like pursuit algorithms. Then, a Bayesian hypothesis test is used to estimate the activity measure for each coefficient separately. So, the aim of this paper is to estimate simple activity measures using a Bayesian framework. This is done by three simple assumptions which are needed when we devise our algorithm. These assumptions are just approximations and the algorithm is devised under these simplifying assumptions. The results of this work have been partially presented in [38].

The activity measure we obtain in this method is similar to what has already been obtained by IDE algorithm [36]. The main difference, however, is that the threshold is obtained mathematically and is calculated throughout the algorithm by some simple parameter estimation techniques.

In this paper, we first introduce our system model and some notations in Section II. Then, in Section III, we propose our Bayesian Hypothesis Testing Algorithm (BHTA). Section IV investigates the stability analysis of the algorithm. Finally, in Section V, we investigate the experimental performance of the BHTA in comparison with other main algorithms.

II. SYSTEM MODEL

The noise vector **e** in (1) is assumed to be zero-mean Gaussian with covariance matrix $\sigma_e^2\mathbf{I}$. In the model, the coefficients are inactive with probability p, and are active with probability $1-p$ (sparsity of **y** implies that p should be near 1). In the inactive case, the values of the coefficients are zero and in the active case the values are obtained from a Gaussian distribution. We call this model the 'spiky model' which is a special case of the Bernoulli-Gaussian model with the variance of the inactive samples being zero. This model has been also used in [39] and [7]. It is suitable for sparse representation of a signal where we would like to decompose a signal as a combination of only a few atoms of the dictionary and the coefficients of the other atoms are zero. So, the probability density of the coefficients in our problem is:

$$p(y_i) = p\delta(y_i) + (1-p)N(0,\sigma_r^2). \quad (2)$$

where $\delta(.)$ denotes the Dirac impulse function. In this model, each coefficient can be written as $y_i = q_i r_i$ where q_i is a binary variable (with a binomial distribution) and r_i is the amplitude of the i'th coefficient with a Gaussian distribution. Each element q_i is the activity of the corresponding coefficient (or corresponding atom):

$$q_i = \begin{cases} 1 & \text{if } y_i \text{ is active (with probability } 1-p) \\ 0 & \text{if } y_i \text{ is inactive (with probability } p) \end{cases}. \quad (3)$$

Consequently, the probability $p(\mathbf{q})$ of the activity vector $\mathbf{q} \triangleq (q_1, q_2, ..., q_m)^T$ is equal to:

$$p(\mathbf{q}) = (1-p)^{n_a}(p)^{m-n_a}. \quad (4)$$

where n_a is the number of active coefficients, i.e., the number of 1's in $\mathbf{q}$. So, the coefficient vector can be written as:

$$\mathbf{y} = \mathbf{Q}\mathbf{r}. \quad (5)$$

where $\mathbf{Q} = \text{diag}(\mathbf{q})$ and $\mathbf{r} \triangleq (r_1, r_2, ..., r_m)^T$ is the 'amplitude vector'. Note that, in this paper, we use the same notation $p(\cdot)$ for both probability and Probability Density Function (PDF).

III. Bayesian Hypothesis Testing Algorithm (BHTA)

The main task in sparse representation algorithms is to determine which atoms are active in the sparse representation of the signal. This can be viewed as a detection task like in the IDE algorithm [36] which an activity function is compared with a decreasing threshold. In some pursuit algorithms (e.g., MP), it is determined by Correlation Maximization (CM). In some other pursuit algorithms (e.g., StOMP), it is done by comparing the correlations with a threshold. In the MAP sense, it is done with posterior maximization over all possible activity vectors [40]. In IBA algorithm [17], the maximization is done by a steepest-ascent algorithm in the M-step within a MAP sense framework. Here we want to determine the activity by a Bayesian hypothesis testing from the correlations. The possible strategies for determining the active atoms for the various algorithms are schematically depicted in Fig. 1(a)-(e).

To develop a hypothesis testing approach, we write (1) as:

$$\mathbf{x} = \sum_{i=1}^{m} \boldsymbol{\varphi}_i y_i + \mathbf{e}. \quad (6)$$

where $\boldsymbol{\varphi}_i$ is the i'th column (i.e., the i'th atom) of the dictionary. So, the correlations between the original signal and the atoms are:

$$z_j \triangleq < \mathbf{x}, \boldsymbol{\varphi}_j > = y_j + \sum_{\substack{i=1 \\ i \neq j}}^{m} y_i b_{ij} + v_j. \quad (7)$$

where $b_{ij} \triangleq < \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j >$ and $v_j \triangleq < \mathbf{e}, \boldsymbol{\varphi}_j >$, and the atoms are assumed to have unit Euclidean norm.

To do a Bayesian hypothesis test based on correlations for determining the activity of the j'th atom, we must compute the posteriors $p(H_1|\mathbf{z})$ and $p(H_2|\mathbf{z})$, where H_1 is the hypothesis that the j'th atom is active and H_2 is the hypothesis that the j'th atom is inactive. To obtain a simple algorithm like pursuit algorithms, assuming the previous estimations of all other coefficients (except the j'th coefficient), we want to detect the activity of only the j'th atom and then update only the j'th coefficient.

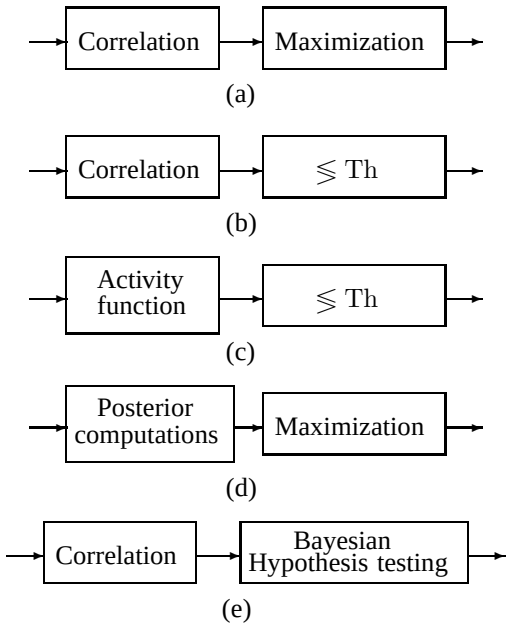

Fig. 1. Idea of detection in various algorithms. (a) MP or OMP (b) StOMP (c) IDE (d) MAP sense (e.g., IBA) (e) Bayesian Hypothesis Testing Algorithm (BHTA).

Since we assume that we know previous estimations of other coefficients, (7) can be written as:

$$z_j - \sum_{i=1, i\neq j}^{m} \hat{y}_i b_{ij} = y_j + \sum_{i=1, i\neq j}^{m} (y_i - \hat{y}_i) b_{ij} + v_j. \quad (8)$$

where $\hat{y}_i$ is the estimation of the i'th coefficient at the current iteration. Let define:

$$c_j \triangleq \sum_{i=1, i\neq j}^{m} \hat{y}_i b_{ij}. \quad (9)$$

$$\gamma_j \triangleq \sum_{i=1, i\neq j}^{m} (y_i - \hat{y}_i) b_{ij} + v_j. \quad (10)$$

The two hypotheses H_1 and H_2 are then:

$$\text{Hypotheses}: \begin{cases} H_1 : z_j - c_j = r_j + \gamma_j \\ H_2 : z_j - c_j = \gamma_j \end{cases}. \quad (11)$$

where c_j is known and γ_j is a noise or error term. In fact, Eq. (11) is a classical detection problem.

A. Hard-BHTA

In this section, we suggest a classical detection solution for solving the problem (11). As it was said before, the hypothesis test involves the computation of the overall posteriors $p(H_1|\mathbf{z})$ and $p(H_2|\mathbf{z})$. But, with the previous formulations, we reach a relatively simple detection problem as in (11). For the simplicity of the algorithm like the pursuit algorithms, we rely only on the correspondent correlation (e.g., z_j) and

hence the simpler posteriors as $p(H_1|z_j)$ and $p(H_2|z_j)$. So, the hypothesis H_1 is chosen when $p(H_1|z_j) > p(H_2|z_j)$, otherwise H_2 is chosen.

Based on Bayes' rule, the above posteriors are proportional to $p(H_1)p(z_j|H_1)$ and $p(H_2)p(z_j|H_2)$ respectively. The prior probabilities for the hypotheses are $p(H_1) = 1-p$ and $p(H_2) = p$ where p is defined in Section II.

Now, for developing our algorithm, we assume the following three main assumptions:

Assumption 1: $(y_i - \hat{y}_i)$ and $(y_j - \hat{y}_j)$ are assumed to be uncorrelated for $i \neq j$.

Assumption 2: The noise term v_j is uncorrelated of the error $(y_i - \hat{y}_i)$ for $i \neq j$.

Assumption 3: The term γ_j in (10) has a Gaussian distribution.

Strictly speaking, Assumption 1 is not mathematically true, because the estimated value of one coefficient clearly influences the estimation of the other coefficients. However, in the following, this assumption provides a first order approximation of a sequence of thresholds for the algorithm, instead of using a heuristically predetermined sequence of thresholds as done in IDE [36]. More precisely, in the following, this assumption is used only for deriving (18). On the other hand, heuristically, we expect that, as the algorithm converges to the true solution, the outputs are closer to the true estimated values and the estimation of each coefficient has less influence to estimating the other ones. We will study this heuristic in our simulations (see Fig. 5). In fact, (18) requires Assumption 2, too which is just used here. Consequently, the experiment of Fig. 5 will experimentally study both Assumptions 1 and 2.

Moreover, Assumption 3 is not strictly true, too. However, since $\gamma_j = \sum_{i=1,i\neq j}^{m}(y_i - \hat{y}_i)b_{ij} + v_j$ is a sum of many (especially for large m's) random variables, one expects from Central Limit Theorem (CLT) that this assumption be a good approximation. We will also study the validity of this assumption experimentally (see Fig. 6). This assumption will be used in deriving the activity measure.

Now, let $\sigma^2_{\gamma_j}$ denote the variance of γ_j which is assumed to be a Gaussian random variable by Assumption 3. Therefore, the activity condition writes:

$$\frac{(1-p)}{\sqrt{2\pi(\sigma^2_{\gamma_j}+\sigma^2_r)}}\exp(\frac{-(z_j-c_j)^2}{2(\sigma^2_{\gamma_j}+\sigma^2_r)}) > \frac{p}{\sqrt{2\pi\sigma^2_{\gamma_j}}}\exp(\frac{-(z_j-c_j)^2}{2\sigma^2_{\gamma_j}}). \tag{12}$$

Simplifying (12) with the assumption that the (unknown) parameters p, σ_r and σ_{γ_j} are known, leads to the following decision rule for the hypothesis testing:

$$\text{Activity}(y_j) \triangleq |z_j - c_j| > \text{Th}_j. \tag{13}$$

where Th_j is the threshold defined as:

$$\text{Th}_j \triangleq \frac{\sigma_{\gamma_j}}{\sigma_r}\sqrt{2(\sigma^2_r+\sigma^2_{\gamma_j})\ln(\frac{p}{1-p}\frac{\sqrt{\sigma^2_r+\sigma^2_{\gamma_j}}}{\sigma_{\gamma_j}})}. \tag{14}$$

The decision rule and the activity function in (13) are the same as in IDE algorithm [36], where one uses a predefined decreasing sequence of thresholds. Improvements with respect to IDE method is that the value of threshold is obtained mathematically with respect to the parameters of the statistical model, i.e., following a Bayesian hypothesis test. Another important difference is that, IDE only uses the same threshold for all coefficients, while BHTA could use a different threshold for each coefficient. However, as we will state in Section V, we use the same threshold for all the coefficients to simplify the algorithm.

Although (14) determines the optimal threshold, it depends on unknown parameters (p, σ_r and σ_γ) which should be estimated from the original signal (**x**). Since estimating the parameters needs also the activity vector **q** which is derived itself by the value of threshold, we use an iterative algorithm. To estimate the parameters p, σ_r and σ_e, we can use sample estimate formulas, which are:

$$\hat{p} = \frac{||\mathbf{q}||_0}{m}, \tag{15}$$

$$\hat{\sigma_e} = \frac{||\mathbf{x} - \mathbf{\Phi}\hat{\mathbf{y}}||_2}{\sqrt{n}}, \tag{16}$$

$$\hat{\sigma_r} = \frac{||\mathbf{r}||_2}{\sqrt{m}}. \tag{17}$$

where **q** is obtained from the previous iteration of the decision rule (13). In [17], it has been proved that these estimates are the MAP estimation of these parameters knowing all other parameters. The initialization of these parameters is also detailed in Section V-A.

The problem here is to estimate the parameter σ_{γ_j} which is the standard deviation of γ_j in (10). By taking the variance from (10), since v_j is a Gaussian random variable with the same variance as e_j which is equal to σ^2_e, then by Assumption 1 and Assumption 2, we will have:

$$\sigma^2_{\gamma_j} = \sigma^2_e + \sum_{i=1,i\neq j}^{m} b^2_{ij}\sigma^2_{i,e_y}. \tag{18}$$

where σ^2_{i,e_y} is the variance of the error term $(y_i - \hat{y}_i)$. The accuracy of the above formula depends on the validity of Assumptions 1 and 2, and will be experimentally studied in Section V.

If the algorithm converges, we expect that $\sigma^2_{\gamma_j}$ decreases. So, we enforce the error variance to decrease geometrically:

$$\sigma^{(n+1)}_{i,e_y} = \alpha\sigma^{(n)}_{i,e_y}. \tag{19}$$

where the parameter α, less than but close to 1, determines the rate of convergence.

In Appendix I, it is shown that if we choose the minimum ℓ^2-norm solution for the first iteration, then the initial estimate of the variance $\sigma^{2^{(0)}}_{j,e_y}$ is:

$$\sigma^{2^{(0)}}_{j,e_y} = \sigma^2_r(\sum_{i\in\text{supp}(\mathbf{y})}\psi^2_{ji}) + \sigma^2_e\sum_i l^2_{ji}. \tag{20}$$

where $\mathbf{\Psi} = [\psi_{ij}] \triangleq -\mathbf{I} + \mathbf{\Phi}^\dagger\mathbf{\Phi}$ and $\mathbf{L} = [l_{ij}] \triangleq 2\mathbf{\Phi}^T - \mathbf{\Phi}^\dagger$. The notation $\text{supp}(\mathbf{y})$ denotes the indices where the coefficients are nonzero. But, we do not know in advance where the nonzero elements are. So, we replace the first right side

term of (20) by its mathematical expectation. The expectation $E(\sum_{i\in \mathrm{supp}(\mathbf{y})} \psi_{ji}^2)$ is equal to $E(\sum_i q_i \psi_{ji}^2) = \sum_i E(q_i)\psi_{ji}^2$ where q_i is the activity of the i'th element which is a Bernoulli variable, and hence $E(q_i) = (1-p)$. So, the following formula can be used to estimate the initial parameter estimation:

$$\sigma_{j,e_y}^{2^{(0)}} \approx \sigma_r^2(1-p)||\boldsymbol{\psi}_j||_2^2 + \sigma_e^2 \sum_i l_{ji}^2. \tag{21}$$

where $\boldsymbol{\psi}_j$ is the j'th row of the matrix $\boldsymbol{\Psi}$.

From (21), (18) and the assumption that the error variances σ_{i,e_y}^2 tends to zero at final iterations, we can find that the value of $\sigma_{\gamma_j}^2$ varies from a large initial value $\sigma_{\gamma_j}^{2^{(0)}} = \sigma_e^2 + \sum_{i=1,i\neq j}^m b_{ij}^2 \sigma_{i,e_y}^{2^{(0)}}$ to a small value $\sigma_{\gamma_j}^{2^{(\infty)}} = \sigma_e^2$. So, from (14), the threshold is changed from an initial large value $\mathrm{Th}_j^{(0)} \triangleq \mathrm{Th}|_{\sigma_{\gamma_j}^{(0)}}$ to a small final value $\mathrm{Th}_j^{(\infty)} \triangleq \mathrm{Th}|_{\sigma_{\gamma_j}^{(\infty)}}$. The initial value and the final value (after infinite iterations) of the threshold are:

$$\mathrm{Th}_j^{(0)} = \mathrm{Th}|_{\sigma_{\gamma_j}^{(0)}}, \tag{22}$$

$$\mathrm{Th}_j^{(\infty)} = \mathrm{Th}^{(\infty)} = \mathrm{Th}|_{\sigma_{\gamma_j}=\sigma_e} \approx K\sigma_e. \tag{23}$$

where $K = \sqrt{2\ln(\frac{p}{1-p}\frac{\sigma_r}{\sigma_e})}$. In (23), it has been assumed that the algorithm converges to the true solution and hence $\sigma_{\gamma_j} \to \sigma_e$. As we can see from (22), the initial thresholds are different for each coefficient. But, all the thresholds are converging to the same value (23).

As we explain in Section V, a common threshold is used for all coefficients for simplicity of the algorithm. As the value of threshold changes from a large value to a small value, the algorithm can detect more and more atoms. During the first iterations, the optimal thresholding strategy in (14) changes the thresholds very fast and then after a few iterations, the thresholds converge to the final small value.

In the thresholding strategy (14), although there can be a simple stopping rule for iterations based on the value of thresholds (which will be explained in our experiments), the number of required iterations for convergence are not known in advance. So, we can use another threshold to predict the number of the iterations in advance. The simplest way for updating the threshold is to decrease the threshold geometrically from the initial value $\mathrm{Th}_j^{(0)}$ in (22) to final value $\mathrm{Th}_j^{(\infty)}$ in (23):

$$\mathrm{Th}_j^{*(n+1)} = \alpha \mathrm{Th}_j^{*(n)}. \tag{24}$$

where superscript $*$ is for simple thresholding strategy. Since the final threshold is the same for all coefficients, we should also use the same threshold for initialization in the simple thresholding strategy. As we see in Section V, we also use the same thresholds for optimal thresholding. So, we can use the same initial value for threshold in simple thresholding just like in optimal thresholding (i.e., $\mathrm{Th}_j^{(0)} = \mathrm{Th}^{(0)}$). So, in simple thresholding, the required number of iterations is:

$$t = \frac{\ln(\frac{\mathrm{Th}^{(\infty)}}{\mathrm{Th}^{(0)}})}{\ln(\alpha)}. \tag{25}$$

where $\mathrm{Th}^{(\infty)}$ is as defined in (23). In other words, using t iterations of (24), $\mathrm{Th}_j^{*(n)}$ changes from $\mathrm{Th}^{(0)}$ to $\mathrm{Th}^{(\infty)}$ which were defined in (22) and (23). So, with this strategy of selecting the thresholds, we can predict the number of iterations in advance. The practical choice will be explained in the experimental results section. We will refer to this method as simple thresholding, while the straightforward method is referred to as optimal thresholding. The simple thresholding strategy is similar to IDE with the difference that here we know the first and last values of thresholds while IDE has no ideas for initial and last values of thresholds.

After updating the activity vector based on decision rule in (13), the estimation of amplitude vector $\mathbf{r}$ which was defined in Section II, based on this estimated activity vector can be done by a Linear Least Square (LLS) estimation [43],[40]:

$$\hat{\mathbf{r}} = \sigma_{\mathbf{r}}^{\mathbf{2}}\hat{\mathbf{Q}}\boldsymbol{\Phi}^{\mathbf{T}}(\sigma_{\mathbf{r}}^{\mathbf{2}}\boldsymbol{\Phi}\hat{\mathbf{Q}}\boldsymbol{\Phi}^{\mathbf{T}} + \sigma_{\mathbf{e}}^{\mathbf{2}}\mathbf{I})^{-\mathbf{1}}\mathbf{x}. \tag{26}$$

where $\hat{\mathbf{q}}$ is the estimated activity vector and $\hat{\mathbf{Q}} = \mathrm{diag}(\hat{\mathbf{q}})$. It is worth mentioning that (26) has the same type of update as used in iterative re-weighted least squares algorithms such as FOCUSS algorithm [9]. In fact, (26) is nothing but this standard approach, but with a novel way for calculating the weights.

B. Soft-BHTA

In the previous subsection, we presented the Hard-version of BHTA. As we saw, determining the threshold is relatively complex. Therefore, in this section we suggest a Soft-version of BHTA to avoid the threshold computation. The main idea of soft version of BHTA is to use soft posterior probabilities as the soft hypothesis testing results instead of binary values '0' or '1' for activity measure q_i. If the value of the posterior probability $p(q_i = 1|z_i)$ is high, then it means that it is more probable that the i'th coefficients are nonzero or the i'th atom is active. So, we simply replace q_i by $p(q_i = 1|z_i)$ as we will see at (28). At the final iteration, we use a hard thresholding for providing a binary value for each q_i. So, with this trick, all atoms are participated in the sparse representation in the initial iterations. Then, we replace the activity measures by the posteriors, which determine active (or inactive) atoms of the sparse representation. In this case, the z_i's are the correlations which are used in pursuit algorithms and $p(q_i = 1|z_i)$ is the posterior of the i'th coefficient conditionally to the correlations.

To compute the posteriors, we use the Bayes rule as:

$$p(q_i = 1|z_i) = p(H_1|z_i) = \frac{p(H_1)p(z_i|H_1)}{p(H_1)p(z_i|H_1) + p(H_2)p(z_i|H_2)}. \tag{27}$$

Using (11), in each iteration of soft version of BHTA, we must do the following update for the soft-activity measure:

$$q_i \leftarrow \frac{\frac{(1-p)}{\sqrt{\sigma_r^2+\sigma_\gamma^2}}\exp(\frac{-(z_i-m_i)^2}{2(\sigma_r^2+\sigma_\gamma^2)})}{\frac{(1-p)}{\sqrt{\sigma_r^2+\sigma_\gamma^2}}\exp(\frac{-(z_i-m_i)^2}{2(\sigma_r^2+\sigma_\gamma^2)}) + \frac{p}{\sigma_\gamma}\exp(\frac{-(z_i-m_i)^2}{2\sigma_\gamma^2})}. \tag{28}$$

where parameters p, σ_r and σ_γ are obtained and updated as in the hard version of BHTA. After the convergence, we can use a simple hard thresholding as $p(q_i = 1|z_i) \geq 0.5$ to obtain the active atoms.

After updating the activity, we can use a formula similar to (26) for updating the amplitude vector.

- Initialization:
  1) Initial parameter estimation:
  $\hat{p}^{(0)} = 0.8$,
  $\hat{\sigma}_r^{(0)} = \frac{||\mathbf{x}||_2}{\sqrt{m(1-\hat{p}^{(0)})}}$,
  $\hat{\sigma}_e^{(0)} = \frac{\hat{\sigma}_r^{(0)}}{5}$,
  $\sigma_{i,e_y}^{2(0)} = \sigma_r^2(1-p)||\boldsymbol{\psi}_i||_2^2 + \sigma_e^2 \sum_k l_{ik}^2$.
  2) Let the initial solution from minimum ℓ^2-norm:
  $\hat{\mathbf{y}}^{(0)} = \boldsymbol{\Phi}^\dagger \mathbf{x}$
  3) Let the correlations $z_j \triangleq < \mathbf{x}, \boldsymbol{\varphi}_j >$
- Until Convergence do:
  1) $c_j = \sum_{i=1, i\neq j}^m \hat{y}_i b_{ij}$
  2) $\sigma_{i,e_y} \leftarrow \alpha \sigma_{i,e_y}$
  3) $\sigma_{\gamma_j}^2 = \sigma_e^2 + \sum_{i=1,i\neq j}^m b_{ij}^2 \sigma_{i,e_y}^2$
  4) $\mathrm{Th}_j = \frac{\sigma_{\gamma_j}}{\sigma_r} \sqrt{2(\sigma_r^2 + \sigma_{\gamma_j}^2) \ln(\frac{p}{1-p} \frac{\sqrt{\sigma_r^2+\sigma_{\gamma_j}^2}}{\sigma_{\gamma_j}})}$
  5) Activity Detection:
  $\mathrm{Activity}(y_j) \triangleq \hat{q}_j = (|z_j - c_j| > \mathrm{Th}_j)$
  6) Amplitude Estimation:
  $\hat{\mathbf{y}} = \sigma_{\mathbf{r}}^{\mathbf{2}} \hat{\mathbf{Q}} \boldsymbol{\Phi}^{\mathbf{T}} (\sigma_{\mathbf{r}}^{\mathbf{2}} \boldsymbol{\Phi} \hat{\mathbf{Q}} \boldsymbol{\Phi}^{\mathbf{T}} + \sigma_{\mathbf{e}}^{\mathbf{2}} \mathbf{I})^{-\mathbf{1}} \mathbf{x}$.
  7) Parameter Estimation Step: using (15), (16) and (17).
- Final answer is $\hat{\mathbf{y}}$.

Fig. 2. The Hard-BHTA algorithm.

C. Summary

Finally, to summarize the presentation of BHTA, we represented the detailed Hard-BHTA algorithm in Fig. 2. Initialization is done by minimum ℓ^2-norm solution (see (34) in Appendix I). Updating the activity vector in Hard-BHTA is done by the decision rule in (13). Updating the coefficients or amplitudes is done by (26). Similarly, updating the parameters are done by equations (18), (19), (21) and with parameters p, σ_e and σ_r computed according to (15), (16) and (17). Threshold determination in Hard-BHTA is done by equations (21), (19) and then (18) for the variance of coefficient errors. Then, as we explained in Section III-A, we suggested two different thresholding strategy which are optimal thresholding (14) and simple thresholding (24). We investigate these different strategies in the simulation results.

IV. Stability Analysis

Because of the thresholding strategy, a complete convergence analysis to the algorithm is very tricky, and is not addressed in this paper. Hence, in this section, we only study the stability of BHTA, i.e., if the algorithm does not diverge and is stable.

The stability of the BHTA is equivalent to the convergence of the sequence of thresholds in (14). As we know, this is a positive sequence and it is bounded below by zero. So, if this sequence is a decreasing sequence, then it converges of course to the value in (23). In Appendix II, we show that with the assumption that $\sigma_{\gamma_j} \ll \sigma_r$, the sufficient condition for the convergence of the sequence of j'th threshold is:

$$\frac{\sigma_r^2}{\sigma_e^2} > \frac{\sum_{i=1,i\neq j}^m b_{ij}^2 \sum_r l_{jr}^2}{(\frac{p}{(1-p)e})^2 - \sum_{i=1,i\neq j}^m b_{ij}^2 \sum_{r\in \mathrm{supp}(\mathbf{y})} \psi_{jr}^2}. \quad (29)$$

where e is the neper number, $\mathbf{B} = \boldsymbol{\Phi}^T \boldsymbol{\Phi}$, $\boldsymbol{\Psi} = -\mathbf{I} + \boldsymbol{\Phi}^\dagger \boldsymbol{\Phi}$ and $\mathbf{L} = [l_{ij}] \triangleq 2\boldsymbol{\Phi}^T - \boldsymbol{\Phi}^\dagger$. If we define the input SNR as in (33), then (29) is equivalent to have an input SNR greater than a minimum input SNR, i.e., $\mathrm{SNR}_i > \mathrm{SNR}_{min}$ which is:

$$\mathrm{SNR}_{min}(\mathbf{y}, j) \triangleq 10 \log \frac{\sum_{i=1,i\neq j}^m b_{ij}^2 \sum_r l_{jr}^2}{(\frac{p}{(1-p)e})^2 - \sum_{i=1,i\neq j}^m b_{ij}^2 \sum_{r\in \mathrm{supp}(\mathbf{y})} \psi_{jr}^2}. \quad (30)$$

where this minimum SNR depends on the unknown $\mathbf{y}$. For canceling the dependence on $\mathbf{y}$, we replace the denominator by its expectation with respect to $\mathbf{y}$ (like for deriving (21)) and the minimum SNR becomes:

$$\mathrm{SNR}_{min}(j) \triangleq 10 \log \frac{||\boldsymbol{\ell}_j^T||_2^2 (||\mathbf{b}_j||_2^2 - 1)}{K + (1-p)||\boldsymbol{\psi}_j||_2^2 (||\mathbf{b}_j||_2^2 - 1)}. \quad (31)$$

where $K = (\frac{p}{(1-p)e})^2$, $\mathbf{b}_j$ is the j'th column of $\mathbf{B}$ and $\boldsymbol{\ell}_j^T$ is the j'th row of $\mathbf{L}$. Although the formula for minimum input SNR in (31) seems complicated, it is not very restrictive, i.e., this minimum value is not very high. To evaluate the values for the minimum input SNR, we compute them for the practical case of CS where the real signals are sparse in DCT domain. In this case, the matrix $\boldsymbol{\Phi} = \boldsymbol{\Gamma}\mathbf{D}$ where $\boldsymbol{\Gamma}$ is the random CS measurement matrix and $\mathbf{D}$ is the DCT matrix. The random measurement matrix elements are drawn from a zero mean normal random distribution with unit variance. The columns of the dictionary matrix $\boldsymbol{\Phi}$ are normalized to have unit norms. A typical simulation in this case shows that the minimum and maximum values of $\mathrm{SNR}_{min}(j)$ over different j's are -11.4127 dB and 0.3560 dB. So, the practical values of SNR_{min} are not very high, and the sufficient condition for stability (29) is a weak condition and easily satisfied.

For the stability analysis of soft-BHTA, we define the two terms in (28) as $l_1 \triangleq \frac{(1-p)}{\sqrt{\sigma_r^2+\sigma_\gamma^2}} \exp(\frac{-(z_i-m_i)^2}{2(\sigma_r^2+\sigma_\gamma^2)})$ and $l_2 \triangleq \frac{p}{\sigma_\gamma} \exp(\frac{-(z_i-m_i)^2}{2\sigma_\gamma^2})$. With these definitions, $l_1 \geq l_2$ is equivalent to $q_i \geq 0.5$ and $l_1 < l_2$ is equivalent to $q_i < 0.5$. It is simple to show that the condition $l_1 \geq l_2$ is equivalent to $|z_i - m_i| \geq \mathrm{Th}_i$ which is similar to the decision rule of Hard-BHTA in (13). Therefore, the stability conditions for hard-BHTA and soft-BHTA are the same.

V. Experiments

The BHTA algorithm is investigated in this section with three different categories of simulation. First, in subsection V-A, we only consider soft and hard versions of the BHTA algorithm. It includes the two different thresholding strategies and some detailed implementation issues of the algorithm. Secondly, in Section V-C, comparison will be done with the other main algorithms for sparse representation both from complexity and estimation accuracy viewpoints. The performance of the algorithms is compared using the Signal to Noise Ratio between the true coefficients and the recovered coefficients, which is defined as:

$$\mathrm{SNR}_o \triangleq 10 \log(\frac{||\mathbf{y}||_2}{||\mathbf{y} - \hat{\mathbf{y}}||_2}). \quad (32)$$

where the index o denotes output SNR. In fact, this SNR in the coefficient domain determines the capability of the sparse representation algorithm to recover the true sparse coefficients

in average. We define another measure which determines the noise level. We refer to it as input SNR:

$$\text{SNR}_i \triangleq 20\log(\frac{\sigma_r}{\sigma_e}). \quad (33)$$

This input SNR is varied from 20dB to 50dB in the experiments.

We use the CPU time as a measure of complexity. Although, the CPU time is not an exact measure, it can give us a rough estimation of the complexity for comparing our algorithms. Our simulations were performed in MATLAB7.0 environment using an AMD Athlon Dual core 4600 with 896 MB of RAM and under Windows XP operating system.

A. Implementation issues of BHTA

In this part of our experiments, the implementation aspects of the BHTA algorithm is experimentally discussed and evaluated. We mainly have two hard and soft versions of the BHTA algorithm, since we use two distinct methods for updating the threshold.

We used a random dictionary matrix with normalized columns whose entries are previously drawn according to a uniform distribution in $[-1, 1]$. The number of atoms is set to $m = 512$ and the signal length to $n = 256$. For the sparse coefficients, we used the model (2) with the probability $p = 0.9$ and unit variance for the active coefficients ($\sigma_r = 1$). So, on the average, about 51 atoms are active in the sparse representation of the signal. The noises or errors are Gaussian with zero-mean and different variances. The measure of performance, the output SNR (32), is averaged over 100 different random realizations of the dictionary, sparse coefficients and noise vector.

For simplifying the algorithm, we use the same variance and threshold for all coefficients. This simplification reduces some of our calculations by a factor of $\frac{1}{m}$. Since the value of σ_e^2 is not known in advance and the term $\sigma_e^2 \sum_i l_{ji}^2$ is small in comparison to other term, we select $\sigma_{j,e_y}^{2^{(0)}} \approx \sigma_r^2(1-p)||\boldsymbol{\psi}_j||_2^2$. To remove the dependency on the index j, we select $\sigma_{j,e_y}^{2^{(0)}} \approx \sigma_r^2(1-p)||\boldsymbol{\Psi}||_F^2$ where the approximation $||\boldsymbol{\psi}_j||_2^2 \approx \frac{||\boldsymbol{\Psi}||_F^2}{m}$ is assumed for large random matrix $\boldsymbol{\Psi}$. With this initialization (which is independent of the coefficient index) and (19), all the error variances σ_{i,e_y} are independent of the index i and assumed to be σ_{e_y}. So, (18) will reduce to $\sigma_{\gamma_j}^2 = \sigma_e^2 + \sigma_{e_y}^2(||\mathbf{b}_j||_2^2 - 1)$. To omit the dependency on j, we use $\sigma_\gamma^2 = \sigma_e^2 + \beta\sigma_{e_y}^2$ where $\beta \approx \frac{||\mathbf{B}||_F^2}{m} - 1$ (since $||\mathbf{b}_j||_2^2 \approx \frac{||\mathbf{B}||_F^2}{m}$ is a first order approximation of $E\{||\mathbf{b}_j||_2^2\} = E\{\frac{||\mathbf{B}||_F^2}{m}\}$ which holds for random dictionaries. Finally, the values of thresholds are the same for all the coefficient indexes.

The initial values of the unknown statistical parameters (p, σ_r and σ_e) are $\hat{p}^{(0)} = 0.8$, $\hat{\sigma}_r^{(0)} = \frac{||\mathbf{x}||_2}{\sqrt{m(1-\hat{p}^{(0)})}}$ and $\hat{\sigma}_e^{(0)} = \frac{\hat{\sigma}_r^{(0)}}{5}$ which is similar to the initialization used in [17], [18]. We can propose some stopping rules for Hard-BHTA. In Hard-BHTA, we used $\text{Max}(|\mathbf{Th}^{(n+1)} - \mathbf{Th}^{(n)}|) < \frac{\hat{\sigma}_r}{1000})$ as an stopping rule. For Soft-BHTA, a similar stopping rule is $\text{Max}(|\mathbf{q}^{(n+1)} - \mathbf{q}^{(n)}|) < \frac{1}{100}$.

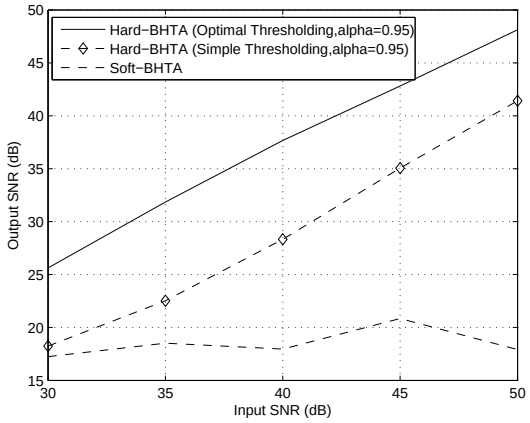

Fig. 3. The output SNR averaged on 100 runs versus the input SNR for Hard-BHTA with two different thresholding strategies and Soft-BHTA. The parameters are $m = 512$, $n = 256$, $p = 0.9$, $\sigma_r = 1$, $\alpha = 0.95$.

For the simple thresholding strategy (which is very similar to IDE), we start from the initial threshold $\text{Th}^{(0)}$ to the final value $\text{Th}^{(\infty)}$ by the geometric series (24). To compute $\text{Th}^{(\infty)} \approx K\sigma_e$, we need the value $\frac{\sigma_r}{\sigma_e}$. In the simulations of this section, we select $\frac{\sigma_r}{\sigma_e} = 100$ for any noise levels and the parameter $\alpha = 0.95$ for both simple thresholding and optimal thresholding. Figure 3 shows the results of the two versions of Hard-BHTA (with the two thresholding strategies) and Soft-BHTA. Clearly, performance of Hard-BHTA is much better than Soft-BHTA performance. Of course, the optimal thresholding strategy (14) yields better results than the simpler strategy (24).

Finally, to determine the best value of the parameter α in (19) and (24), we represent the results of our algorithm with respect to the value of α when $\sigma_e = 0.01$ in Fig. 4. As it can be seen, better results are obtained when the value is around $\alpha = 0.95$ for optimal thresholding and for simple thresholding. However, we use $\alpha = 0.95$ for the next experiments unless we state otherwise. As we can see, the Soft-BHTA and Hard-BHTA with simple thresholding are sensitive to the value of parameter α, while Hard-BHTA with optimal thresholding is less sensitive to this parameter.

B. Investigating the assumptions

In these experiments, the Assumptions 1 to 3 of Section III-A are investigated. Since Assumptions 1 and 2 have only been used in deriving (18), the influence of these assumptions is experimentally investigated by computing the absolute difference between both sides of (18) i.e., the error term $|\sigma_{\gamma_j}^2 - \sigma_e^2 - \sum_{i=1, i\neq j}^m b_{ij}^2 \sigma_{i,e_y}^2|$. Figure 5 shows the error term over all indices $1 \leq j \leq m$ versus the iteration number. It can be seen that the averaged error term is small in comparison to $\sigma_{\gamma_j}^2$, and vanishes after a few (5 to 6) iterations. In other words, as the algorithms converges to the solution, the Assumptions 1 and 2 become very accurate.

Assumption 3 (the Gaussianity of γ_j), is evaluated by computing the normalized Kurtosis defined as $\text{Kurt}(\gamma_j) \triangleq$

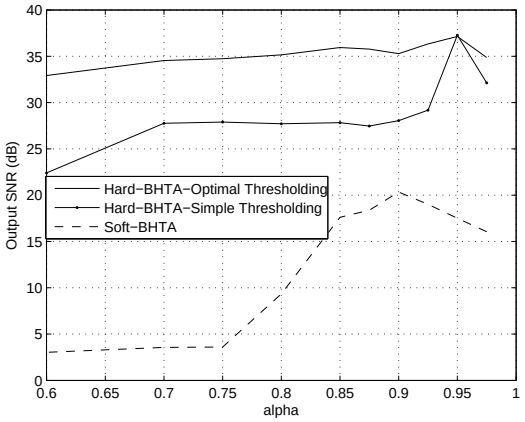

Fig. 4. The output SNR averaged on 100 runs versus the simulation parameter α. Other parameters are $m = 512$, $n = 256$, $p = 0.9$, $\sigma_r = 1$.

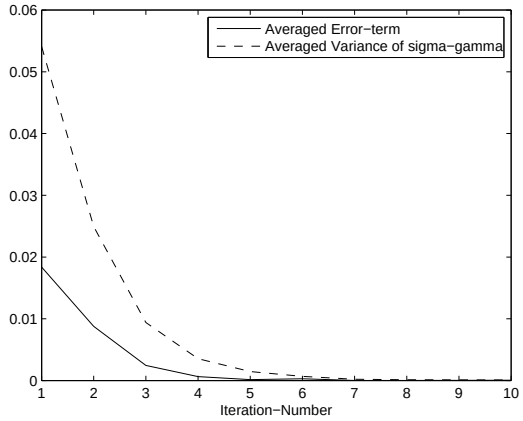

Fig. 5. The error term $|\sigma^2_{\gamma_j} - \sigma^2_e - \sum_{i=1,i\neq j}^{m} b^2_{ij}\sigma^2_{i,e_y}|$ and variance $\sigma^2_{\gamma_j}$ averaged over all indexes $1 \leq j \leq m$ versus the iteration number. It is computed over 100 runs of simulations. The parameters are $m = 512$, $n = 256$, $p = 0.9$, $\sigma_r = 1$ and $\sigma_n = 0.01$.

$\frac{\gamma_j^4}{E^2(\gamma_j^2)} - 3$ [42]. Recall that the kurtosis of a Gaussian random variable is zero. We averaged this measure over all coefficients indexes j and also over runs of simulation. Averaged kurtosis versus iteration number is showed in Fig. 6. It can be seen that the value of kurtosis is small after some iterations and hence the assumption of Gaussianity of γ_j would be a good approximation after a few iterations.

C. Comparison with other sparse representation algorithms

In this experiment, we only compare the optimal thresholding version of Hard-BHTA and Soft-BHTA with other main sparse representation algorithms such as BP, MP, OMP, StOMP, SL0, BCS, GP, GPSR and IBA. In this experiment, we use another model for generating the sparse coefficients. We choose the inactivity probability $p = 0.9$ and all active coefficients are set equal to 1 instead to be distributed as a Gaussian random variable with a unit variance. The locations of active coefficients are uniformly random. The input SNR

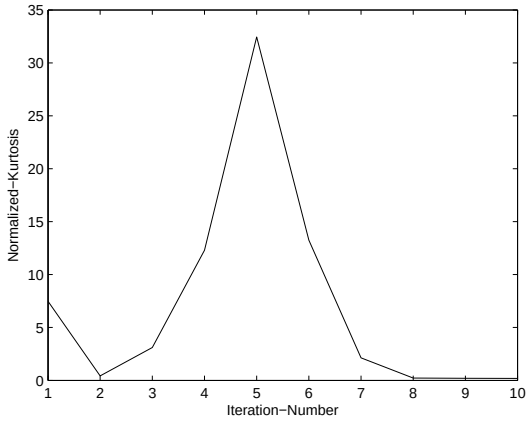

Fig. 6. The normalized kurtosis computed over all indexes $1 \leq j \leq m$ and 100 runs of simulations versus the iteration number. The parameters are $m = 512$, $n = 256$, $p = 0.9$, $\sigma_r = 1$ and $\sigma_n = 0.01$.

is defined as $\text{SNR}_i \triangleq 20\log(\frac{1}{\sigma_e})$. The comparisons are done in three cases. The first case is the comparison of the average estimation accuracy (Output SNR) versus the input noise level (Input SNR). The second comparison is the same measure of estimation accuracy (Output SNR) versus the sparsity level. Finally, we compare complexity of the different algorithms. In all experiments, the results are averaged over 100 different runs, with random dictionary and random sparse coefficents.

For BHTA, we use the simulation parameters used in the previous experiment. BP algorithm was tested using ℓ^1-magic package [45]. Since there are 51 active atoms in average, we run the MP, OMP and StOMP algorithms (implemented by SparseLab[2]) for twice the number of active atoms which is 102 (a similar strategy is used in [32] for yielding better performances). For StOMP, we used default parameters of the SparseLab code with the difference that we used similar number of iterations to MP and OMP (instead of 10 which is the default value). For SL0 algorithm[3], we used the minimum σ equal to σ_e and the decreasing factor, a parameter which determined a tradeoff between accuracy and speed, equal to 0.9. For the IBA algorithm, we used 4 iterations for both the M-step and the overall algorithm [17]. For GPSR algorithm[4] [15], we used $\tau = 0.1||\mathbf{\Phi}^T\mathbf{x}||_\infty$ as suggested by the authors. We also use a debiasing step in GPSR algorithm. The algorithm stops if the norm of the difference between two consecutive estimates, divided by the norm of one of them falls below 10^{-4}. The other parameters of GPSR are the default values. We also used the recommended and default parameters for BCS[5] [23]. We used Sparsify toolbox for GP algorithm[6] [32], with default parameters and we stop the algorithm if

[2]The codes SolveMP, SolveOMP, SolveStOMP.m are available at http://sparselab.stanford.edu

[3]The code sl0.m is used which is available at http://ee.sharif.edu/~SLzero

[4]The code GPSR_fun.m is used which is available at http://www.lx.it.pt/~mtf/GPSR/GPSR6.0

[5]The codes in bcs-vb.zip are used which are available at http://people.ee.duke.edu/~lihan/cs

[6]We used the latest version of code greed_gp.m, available at http://www.see.ed.ac.uk/~tblumens/sparsify/sparsify.html

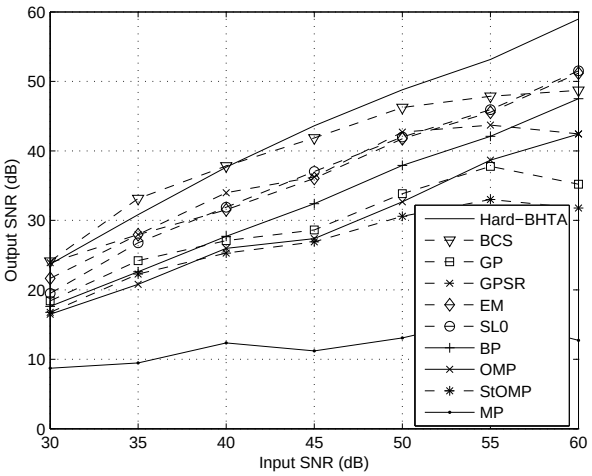

Fig. 7. The averaged output SNR on 100 runs of simulations versus input SNR for various algorithms. The parameters are $m = 512$, $n = 256$, $p = 0.9$, $\alpha = 0.95$.

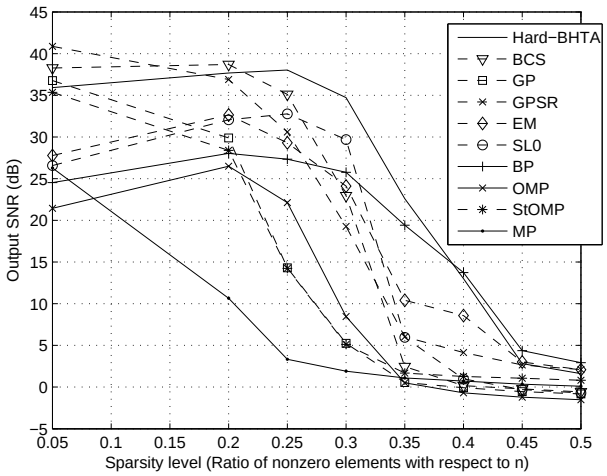

Fig. 8. The averaged Output SNR on 100 runs of simulations versus sparsity level. The parameters are $m = 512$, $n = 256$, $\sigma_e = 0.01$, $\alpha = 0.95$.

the mean squared error of residual is below σ_e^2. Figure 7 shows the performance of the various algorithms (output SNR in coefficient domain) versus the noise level (input SNR). It shows that our algorithm is one of the best algorithms in terms of estimation accuracy specially for low noises.

To investigate the performance of the algorithms for various sparsity levels, we plot (Fig. 8) the output SNR versus sparsity level which is determined in our statistical model (2) by probability $(1 - p)$. In this experiment, we used a fixed number of nonzero coefficients with amplitudes equal to 1. The sparsity ratio is defined as $\frac{||y||_0}{n}$. Again, it can be seen in Fig. 8 that the Hard-BHTA algorithm is one of the best algorithms.

Finally, we compare the algorithm in terms of speed. Figure 9 shows the average simulation time of various algorithms with respect to the dimension of our sparse representation problem (i.e., signal length). The dimension of our problem is determined with the number of atoms and the length of the signal. In this experiment, we used $m = 2n$ for different signal lengths from 64 to 512. It shows that our algorithm is the most complex method.

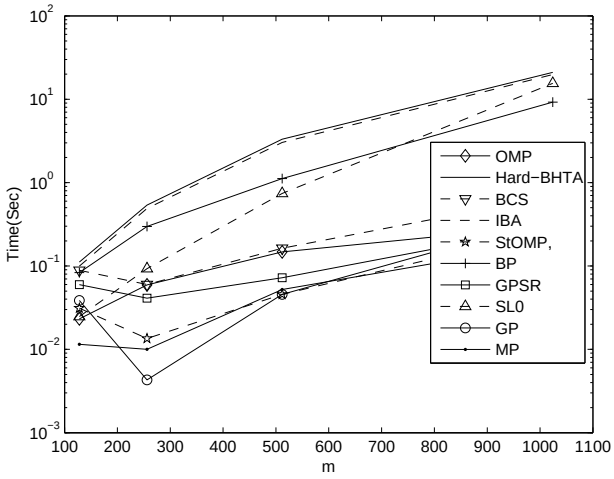

Fig. 9. The averaged simulation time on 100 runs of simulations versus m in the case of $m = 2n$. The parameters are $p = 0.9$, $\sigma_e = 0.01$, $\alpha = 0.95$.

D. Comparison of algorithms in real-field decoding application

In this section, we compare the algorithms in real-field coding. In real-field coding, we first encode a block vector of real-valued samples by a random generating matrix. Assume the input message vector is $\mathbf{s} = [s_1, s_2, ..., s_n]^T$. The encoded message is $\mathbf{x} = \mathbf{Gs}$ where $\mathbf{G}$ is an $n \times m$ matrix with $n < m$ (adding redundancy to input messages). Then, we assume that channel adds both impulse errors and a background noise. So, the channel output is equal to $\mathbf{y} = \mathbf{x} + \mathbf{e} + \mathbf{v}$ where $\mathbf{e}$ is channel errors and $\mathbf{v}$ is the background noise. We can define a parity check matrix $\mathbf{H}$ associated to the generating matrix $\mathbf{G}$ such that $\mathbf{HG} = 0$ [3]. Then, the errors can be reconstructed by solving the underdetermined linear system of equations $\tilde{\mathbf{y}} \triangleq \mathbf{Hy} = \mathbf{He} + \mathbf{w}$ where $\mathbf{w} \triangleq \mathbf{Hv}$ is the noise term. After estimating the error vector $\hat{\mathbf{e}}$ by means of sparse representation algorithms, it can be subtracted from the output channel to yield the corrected encoded message $\hat{\mathbf{x}}$. Finally, the original messages can be recovered using $\hat{\mathbf{s}} = \mathbf{G}^{\dagger}\hat{\mathbf{x}}$ where $\mathbf{G}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{G}$.

The standard Lena image is used as input message. The pixels of image are vectorized and then divided in blocks of length $n = 128$. Entries of the generating matrix g_{ij} are also randomly selected from uniform distribution in $[-1, 1]$. For channel impulse errors, we used the model (2). The background noise $\mathbf{v}$ is generated from zero mean Gaussian distribution with variance σ_v^2. The input SNR is defined as $\text{SNR}_i \triangleq 20\log(\frac{\sigma_r}{\sigma_v})$. The output SNR between the original message $\mathbf{s}$ and the estimated message $\hat{\mathbf{s}}$ is similarly defined as $\text{SNR}_o \triangleq 10\log(\frac{||\mathbf{s}||_2}{||\mathbf{s}-\hat{\mathbf{s}}||_2})$. We vary the input SNR from 30dB to 60dB. Figure 10 shows the averaged result (over 100 blocks

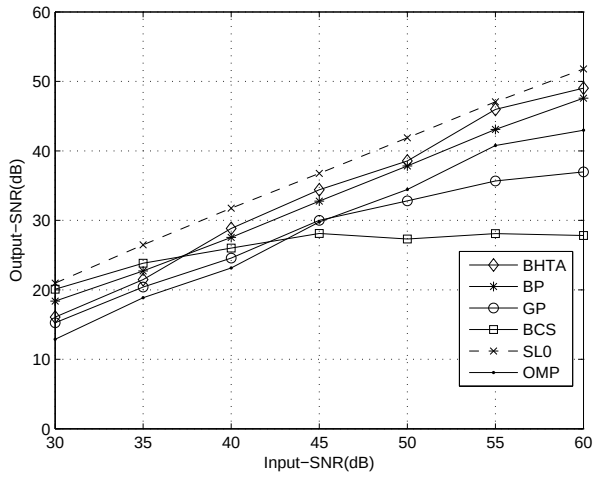

Fig. 10. The averaged output SNR on 100 runs of simulations versus input SNR for various algorithms in real-field decoding when impulse noise is BG with parameters $p = 0.9$ and $\sigma_r = 1$. The other parameters are $m = 256$, $n = 128$, $\alpha = 0.95$.

of the Lena image) of output SNR versus input SNR for BG model for errors. For more clarity, we only compare our BHTA algorithm with BP, GP, BCS, SL0 and OMP, and parameters are chosen as in the previous experiments. There are just one difference: we used stopTol $= 10^{-6}$ for GP algorithm for achieving better results. The results show that again the BHTA algorithm is one of the best algorithms for real-field decoding application.

VI. Conclusions

In this paper, we proposed a Bayesian hypothesis testing algorithm for sparse representation problem which can also be used in other contexts like CS or SCA. The main idea of this algorithm is to use rather simple Bayesian hypothesis test to estimate which atoms are active in the sparse expansion of the signal. The activities of atoms, which are detected through a Bayesian test, are based on a comparison of the activity measure with a threshold. The interest of the Hard-BHTA algorithm is its ability to determine the thresholds mathematically with simple parameter estimation techniques rather than heuristically. It can be computed practically with simple parameter estimation techniques. The comparison of Hard-BHTA algorithm with the state of the art algorithms shows that Hard-BHTA algorithm achieves one of the best performances, but at the price of the highest complexity.

Appendix I
Initial parameter estimation for minimum ℓ^2-norm solution

If the minimum ℓ^2-norm solution is selected as the solution for the first iteration, then we have:

$$\hat{\mathbf{y}}^{(0)} = \mathbf{\Phi}^\dagger \mathbf{x} = \mathbf{H}\mathbf{y} + \boldsymbol{\eta}. \tag{34}$$

where $\mathbf{H} \triangleq \mathbf{\Phi}^\dagger\mathbf{\Phi}$ and $\boldsymbol{\eta} \triangleq \mathbf{\Phi}^\dagger\mathbf{e}$. Then, each element of the initial solution can be written as:

$$\hat{y}_i^{(0)} = \sum_j h_{ij} y_j + \eta_i. \tag{35}$$

By definition:

$$\gamma_j^{(0)} = v_j + \sum_{i=1, i\neq j}^{m} (y_i - \hat{y}_i^{(0)}) b_{ij}. \tag{36}$$

Now, replacing (35) in (36) results in:

$$\gamma_j^{(0)} = v_j - \sum_{i=1,i\neq j}^{m} \sum_r h_{ir} y_r b_{ij} + \sum_{i=1,i\neq j}^{m} y_i b_{ij} + \sum_{i=1,i\neq j}^{m} \eta_i b_{ij}. \tag{37}$$

If we add and subtract the terms with $i = j$, then after some simplifications and calculations, we have:

$$\gamma_j^{(0)} = v_j - \sum_{i=1}^{m} b_{ij} \sum_r h_{jr} y_r + \sum_r h_{jr} y_r + \sum_{i=1}^{m} y_i b_{ij} - y_j + \sum_{i=1}^{m} \eta_i b_{ij} - \eta_j. \tag{38}$$

It leads to the following matrix form:

$$\boldsymbol{\gamma}^{(0)} = \mathbf{v} + (\mathbf{B} - \mathbf{I})\boldsymbol{\eta} - (\mathbf{I} - \mathbf{B} + \mathbf{B}^T\mathbf{H} - \mathbf{H})\mathbf{y}. \tag{39}$$

with $b_{ij} \triangleq < \boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j >$ and $v_j \triangleq < \mathbf{e}, \boldsymbol{\varphi}_j >$. Using $\mathbf{v} = \mathbf{\Phi}^T\mathbf{e}$ and $\boldsymbol{\eta} = \mathbf{\Phi}^\dagger\mathbf{e}$, then we have:

$$\boldsymbol{\gamma}^{(0)} = \mathbf{\Psi}\mathbf{y} + \mathbf{L}\mathbf{e}. \tag{40}$$

where $\mathbf{L} = \mathbf{\Phi}^T + (\mathbf{B} - \mathbf{I})\mathbf{\Phi}^\dagger$ and $\mathbf{\Psi} = -\mathbf{B}^T\mathbf{H} + \mathbf{B} + \mathbf{H} - \mathbf{I}$. Using $\mathbf{B} = \mathbf{\Phi}^T\mathbf{\Phi}$ and $\mathbf{H} \triangleq \mathbf{\Phi}^\dagger\mathbf{\Phi}$, we have $\mathbf{L} = 2\mathbf{\Phi}^T - \mathbf{\Phi}^\dagger$ and $\mathbf{\Psi} = -\mathbf{I} + \mathbf{H}$. Finally, (40) results in (20).

Appendix II
Sufficient condition for stability of Hard-BHTA

For small values of σ_{γ_j} in comparison to σ_r, it can be seen that the threshold is proportional to:

$$\mathrm{Th}_j \propto \sigma_{\gamma_j} \sqrt{\ln(\frac{p}{1-p}\frac{\sigma_r}{\sigma_{\gamma_j}})}. \tag{41}$$

Therefore, if we define $x = \sigma_{\gamma_j}$ and $c \triangleq \frac{p}{1-p}\sigma_r$, then we should investigate monotonicity of the function $f(x) = x\sqrt{\ln(\frac{c}{x})}$. Using the derivative of this function, it can be seen that the function is decreasing for $x < \frac{c}{e}$. This means $\sigma_{\gamma_j}^2 < k^2\sigma_r^2$ where $k = \frac{p}{(1-p)e}$. So, we should have:

$$\sigma_e^2 + \sum_{i=1,i\neq j}^{m} b_{ij}^2 \sigma^{2}{}_{i,e_y}^{(n)} < k^2\sigma_r^2. \tag{42}$$

Then, for the next iteration, it is obvious that the above condition is satisfied because if $\alpha < 1$ then:

$$\sigma_e^2 + \alpha^2 \sum_{i=1,i\neq j}^{m} b_{ij}^2 \sigma^{2}{}_{i,e_y}^{(n)} < k^2\sigma_r^2. \tag{43}$$

We use the condition in (42) at initialization as the sufficient condition for a decreasing threshold. Replacing the initial variance of (20) in (42) for $n = 0$, then after some simple manipulations, leads to the sufficient condition (29).

ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their fruitfull suggestions. Moreover, the first author would also like to thank METISS group and INRIA/Rennes (IRISA) since most of this work was done when the first author were there as a visiting researcher.

REFERENCES

[1] M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neural Computation*, vol. 13, no. 4, pp. 863–882, 2001.

[2] R. Gribonval and S. Lesage, "A survey of sparse component analysis for blind source separation: principles, perspectives, and new challenges," in *Proceedings of ESANN'06*, pp. 323–330, April 2006.

[3] E. J. Candès and T. Tao, "Decoding by Linear Programming," *IEEE Trans. Info. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec 2005.

[4] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. on Image Proc.*, vol. 15, no. 12, pp. 3736–3745, 2006.

[5] D. L. Donoho, "Compressed Sensing," *IEEE Trans. Info. Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[6] R. Baraniuk, "Compressive sensing," *IEEE Signal. Process. Magazine*, vol. 24, no. 4, pp. 118–121, July 2007.

[7] E. Larsson and Y. Selen, "Linear Regression with a sparse parameter vector," *IEEE Trans. on Signal Proc.*, vol. 55, pp. 451–460, 2007.

[8] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. on Signal Proc.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[9] I.F. Gorodnitski and B.D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: a re-weighted norm minimization algorithm," *IEEE Trans. on Signal Proc.*, vol. 45, pp. 600–616, 1997.

[10] D. L. Donoho and M. Elad "Optimally sparse representation in general (nonorthogonal) dictionaries via l^1-minimization," *Proc. Nat. Acad. Sci.*, vol. 100, no. 5, pp. 2197–2202, March 2003.

[11] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *IEEE Trans. Info. Theory*, vol. 49, no. 12, pp. 3320–3325, Dec 2003.

[12] J. A. Tropp, "Just relax: convex Programming methods for identifying sparse signals in noise," *IEEE Trans. Info. Theory*, vol. 52, no. 3, pp. 1030–1051, March 2006.

[13] D. L. Donoho, M. Elad, and V. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Trans. Info. Theory*, vol. 52, no. 1, pp. 6–18, Jan 2006.

[14] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[15] M. A. T. Figueirado, R. D. Nowak, and S. J. Wright, "Gradient Projection for Sparse Reconstruction: Application to compressed sensing and other Inverse Problems," *To appear in the IEEE Journal of selected topics in signal processing.*, 2007. Available at URL:`http://www.ece.wisc.edu/~nowak/GPSR.pdf.`

[16] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "A method for large-scale ℓ^1-regularized least squares problems with applications in signal processing and statistics," *Preprint*, 2007. Available at URL:`http://www.dsp.ece.rice.edu/CS`.

[17] H. Zayyani, M. Babaie-zadeh, and C. Jutten, "An iterative Bayesian algorithm for Sparse Component Analysis (SCA) in presence of noise," *Submitted to IEEE Trans. on Signal Proc.*, 2008.

[18] H. Zayyani, M. Babaie-Zadeh, and C. Jutten, "Decoding real-field codes by an iterative Expectation-Maximization (EM) algorithm," *Proceedings of ICASSP'08*, pp. 3169-3172, Las Vegas, USA, Mar-Apr 2008.

[19] B. D. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," *IEEE Trans. on Signal Proc.*, vol. 47, pp. 187–200, Jan 1999.

[20] B. D. Rao, K. Engan, S. F. Cotter, J. Palmer, and K. Kreutz-Delgado, "Subset selection in noise based on diversity measure minimization," *IEEE Trans. on Signal Proc.*, vol. 51, pp. 760–770, March 2003.

[21] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[22] D. Wipf and B. D. Rao, "Sparse Bayesian Learning for basis selection," *IEEE Trans. on Signal Proc.*, vol. 52, no. 8, pp. 2153–2164, 2004.

[23] S. Ji, Y. Xue, and L. Carin, "Bayesian Compressive Sensing," *IEEE Trans. on Signal Proc.*, vol. 56, no. 6, pp. 2346–2356, June 2008.

[24] G. H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed ℓ^0norm," *IEEE Trans. on Signal Proc.*, vol. 57, no. 1, pp. 289–301, January 2009.

[25] S. J. Wright, R. D. Nowak, and M. A. T. Figueirado,, "Sparse reconstruction by separable approximation," *Preprint.*, 2008. Available at URL:`http://www.dsp.ece.rice.edu/CS`.

[26] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," *Proceedings of ICASSP'08*, pp. 3869-3872, Las vegas, USA, Mar-Apr 2008.

[27] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with application to wavelet decomposition," *proceeding of the 27th Annual Asilomar Conf. Signals, systems, and Computers*, vol. 1, pp. 40–44. 1993.

[28] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit," *Preprint*, 2006. Available at URL:`http://www-stat.stanford.edu/~idrori/STOMP.pdf.`

[29] O. D. Escoda, L. Granai, and P. Vandergheynst, "On the use of a priori information for sparse signal approximations," *IEEE Trans. on Signal Proc.*, vol. 54, no. 9, pp. 3468–3482, Sep 2006.

[30] P. Just, P. Vandergheynst, and P. Frossard, "Tree-Based Pursuit: Algorithms and properties," *IEEE Trans. on Signal Proc.*, vol. 54, no. 12, pp. 4685–4697, Dec 2006.

[31] D. Needel and R. Vershynin, "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *Preprint*, 2007. Available at URL:`http://www.dsp.ece.rice.edu/CS`.

[32] T. Blumensath and M. Davies, "Gradient Pursuits," *IEEE Trans. on Signal Proc.*, vol. 56, pp. 2370–2382, June 2008.

[33] T. Blumensath and M. Davies, "Stagewise weak gradient pursuits Part I: Fundamentals and numerical studies," *Preprint*, 2008. Available at URL:`http://www.dsp.ece.rice.edu/CS`.

[34] D. Needel and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Preprint*, 2008. Available at URL:`http://www.dsp.ece.rice.edu/CS`.

[35] I. Daubechies, M. Defries, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communication on Pure and Applied Mathematics.*, vol. 57, pp. 1413–1457, 2004.

[36] A. A. Amini, M. Babaie-zadeh, and C. Jutten, "A fast method for sparse component analysis based on iterative detection-projection," *Proceedings of MaxEnt 2006* , pp. 123–130, 2006.

[37] P. Schniter, L. C. Potter, and J. Ziniel, "Fast Bayesian Matching Pursuit," *Proc. Workshop on Information Theory and Applications (ITA)*, pp. 326-333, La Jolla, Canada, Jan 2008.

[38] H. Zayyani, M. Babaie-Zadeh, and C. Jutten, "Bayesian pursuit algorithm for sparse representation," *Proceedings of ICASSP'09*, pp. 1549-1552, Taipe, Taiwan, April 2009.

[39] E. I. George and R. E. McCulloch, "Approaches for Bayesian variable selection," *Statistica Sinica.*, vol. 7, pp. 339–373, 1997.

[40] H. Zayyani, M. Babaie-Zadeh, and C. Jutten, "Source estimation in noisy sparse component analysis," *15'th Intl. Conf. on Digital Signal Processing (DSP2007)* , pp. 219-222, Cardiff, UK, July 2007.

[41] A. Papoulis and S. U. Pillai *Probability, Random variables and stochastic processes*, McGrawHill, 2002.

[42] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*, John Willey and sons, 2001.

[43] A. Bjork, *Numerical methods for least squares problems*, SIAM, 1996.

[44] R. Gribonval and P. Vandergheynst, "On the exponential convergence of matching pursuits in quasi-incoherent dictionaries," *IEEE Trans. Info. Theory*, vol. 52, no. 1, pp. 255–261, Jan 2006.

[45] E. J. Candès and J. Romberg, "ℓ^1-magic: Recovery of sparse signals via convex programming,", 2005, Available at URL:`http://www.acm.caltech.edu/l1magic/downloads/l1-magic.pdf.`