CHAPTER

# SPARSITY-AWARE LEARNING: ALGORITHMS AND APPLICATIONS

# 10

## CHAPTER OUTLINE




**Machine Learning. http://dx.doi.org/10.1016/B978-0-12-801522-3.00010-0**

## 10.1 INTRODUCTION

This chapter is the follow-up to the previous one concerning sparsity-aware learning. The emphasis now is on the algorithmic front. Following the theoretical advances concerning sparse modeling, a true scientific happening occurred in trying to derive algorithms tailored for the efficient solution of the related constrained optimization tasks. Our goal is to present the main directions that have been followed and to provide in a more explicit form some of the most popular algorithms. We will discuss batch as well as online algorithms. This chapter can also be considered as a complement to Chapter 8, where some aspects of convex optimization were introduced; a number of algorithms discussed there are also appropriate for tasks involving sparsity-related constraints/regularization.

Besides describing various algorithmic families, some variants of the basic sparsity-promoting $\ell_1$ and $\ell_0$ norms are discussed. Also, typical examples are shown and a case study concerning time-frequency analysis is given. Finally, some more recent theoretical advances concerning the discussion of "synthesis vs. analysis" models are provided.

## 10.2 SPARSITY-PROMOTING ALGORITHMS

In the previous chapter, our emphasis was on highlighting some of the most important aspects underlying the theory of sparse signal/parameter vector recovery from an underdetermined set of linear equations. We now turn our attention to the algorithmic aspects of the problem (e.g., [52, 54]). The issue now becomes that of discussing *efficient* algorithmic schemes, which can achieve the recovery of the unknown set of parameters. In Sections 9.3 and 9.5, we saw that the constrained $\ell_1$ norm minimization (basis pursuit) can be solved via linear programming techniques and the LASSO task via convex optimization schemes. However, such general purpose techniques tend to be inefficient, because they often require many iterations to converge, and the respective computational resources can be excessive for practical applications, especially in high-dimensional spaces, $\mathbb{R}^l$. As a consequence, a huge research effort has been invested with the goal of developing efficient algorithms that are tailored to these specific tasks. Our aim here is to provide the reader with some general trends and philosophies that characterize the related activity. We will focus on the most commonly used and cited algorithms, which at the same time are structurally simple, so the reader can follow them, without deeper knowledge of optimization. Moreover, these algorithms involve, in one way or another, arguments that are directly related to points and notions we have already used while presenting the theory; thus, they can also be exploited from a pedagogical point of view in order to strengthen the reader's understanding of the topic. We start our review with the class of batch algorithms, where all data are assumed to be available prior to the application of the algorithm, and then we will move on to online/time-adaptive schemes. Furthermore, our emphasis is on algorithms that are appropriate for any sensing matrix. This is stated in order to point out that in the literature, efficient algorithms have also been developed for specific forms of highly structured sensing matrices, and exploiting their particular structure can lead to reduced computational demands [61, 93].

There are three rough types of families along which this algorithmic activity has grown: (a) greedy algorithms, (b) iterative shrinkage schemes, and (c) convex optimization techniques. We have used the word rough because in some cases, it may be difficult to assign an algorithm to a specific family.

### 10.2.1 GREEDY ALGORITHMS

Greedy algorithms have a long history; see, for example, [114] for a comprehensive list of references. In the context of dictionary learning, a greedy algorithm known as *matching pursuit* was introduced in [88]. A greedy algorithm is built upon a series of *locally* optimal *single-term* updates. In our context, the goals are (a) to unveil the "active" columns of the sensing matrix $X$, that is, those columns that correspond to the nonzero locations of the unknown parameters; and (b) to estimate the respective sparse parameter vector. The set of indices that correspond to the nonzero vector components is also known as the *support*. To this end, the set of active columns of $X$ (and the support) is increased by one at each iteration step. In the sequel, an updated estimate of the unknown sparse vector is obtained. Let us assume that, at the $(i-1)$th iteration step, the algorithm has selected the columns denoted as $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \ldots, \mathbf{x}_{j_{i-1}}$, with $j_1, j_2, \ldots, j_{i-1} \in \{1, 2, \ldots, l\}$. These indices are the elements of the currently available support, $S^{(i-1)}$. Let $X^{(i-1)}$ be the $N \times (i-1)$ matrix, having $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \ldots, \mathbf{x}_{j_{i-1}}$ as its columns. Let also the current estimate of the solution be $\boldsymbol{\theta}^{(i-1)}$, which is a $(i-1)$-sparse vector, with zeros at all locations with index outside the support.

**Algorithm 10.1 (Orthogonal matching pursuit (OMP)).**

The algorithm is initialized with $\boldsymbol{\theta}^{(0)} := \mathbf{0}$, $\boldsymbol{e}^{(0)} := \boldsymbol{y}$, and $S^{(0)} = \emptyset$. At iteration step $i$, the following computational steps are performed:

1. Select the column $\mathbf{x}_{j_i}$ of $X$, which is *maximally* correlated to (forms the least angle with) the respective error vector, $\boldsymbol{e}^{(i-1)} := \boldsymbol{y} - X\boldsymbol{\theta}^{(i-1)}$, that is,
$$\mathbf{x}_{j_i} : j_i := \arg\max_{j=1,2,\ldots,l} \frac{\left|\mathbf{x}_j^{\mathrm{T}} \boldsymbol{e}^{(i-1)}\right|}{\|\mathbf{x}_j\|_2}.$$
2. Update the support and the corresponding set of active columns: $S^{(i)} = S^{(i-1)} \cup \{j_i\}$, and $X^{(i)} = [X^{(i-1)}, \mathbf{x}_{j_i}]$.
3. Update the estimate of the parameter vector: Solve the least-squares (LS) problem that minimizes the norm of the error, using the active columns of $X$ only, that is,
$$\tilde{\boldsymbol{\theta}} := \arg\min_{z \in \mathbb{R}^i} \left\|\boldsymbol{y} - X^{(i)}\boldsymbol{z}\right\|_2^2.$$
Obtain $\boldsymbol{\theta}^{(i)}$ by inserting the elements of $\tilde{\boldsymbol{\theta}}$ in the respective locations $(j_1, j_2, \ldots, j_i)$, which comprise the support (the rest of the elements of $\boldsymbol{\theta}^{(i)}$ retain their zero values).
4. Update the error vector
$$\boldsymbol{e}^{(i)} := \boldsymbol{y} - X\boldsymbol{\theta}^{(i)}.$$

The algorithm terminates if the norm of the error becomes less than a preselected user-defined constant, $\epsilon_0$. The following observations are in order.

*Remarks 10.1.*

- Because $\boldsymbol{\theta}^{(i)}$, in Step 3, is the result of an LS task, we know from Chapter 6 that the error vector is orthogonal to the subspace spanned by the active columns involved, that is,
$$\boldsymbol{e}^{(i)} \perp \text{span}\left\{\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_i}\right\}.$$

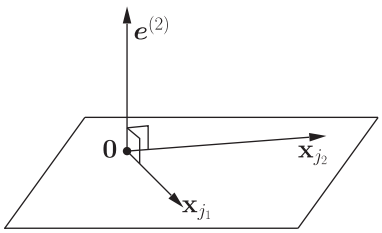


**FIGURE 10.1**

The error vector at the $i$th iteration is orthogonal to the subspace spanned by the currently available set of active columns. Here is an illustration for the case of the three-dimensional Euclidean space $\mathbb{R}^3$, and for $i = 2$.

This guarantees that in the next step, taking the correlation of the columns of $X$ with $\boldsymbol{e}^{(i)}$, none of the previously selected columns will be reselected; they result to zero correlation, being orthogonal to $\boldsymbol{e}^{(i)}$; see Figure 10.1.

- The column, which has maximal correlation (maximum absolute value of the inner product) with the currently available error vector, is the one that maximally reduces (compared to any other column) the $\ell_2$ norm of the error, when $\boldsymbol{y}$ is approximated by linearly combining the currently available active columns. This is the point where the heart of the greedy strategy beats. This minimization is with respect to a *single term*, keeping the rest fixed, as they have been obtained from the previous iteration steps (Problem 10.1).
- Starting with all the components being zero, if the algorithm stops after $k_0$ iteration steps, the result will be a $k_0$-sparse solution.
- Note that there is no optimality in this searching strategy. The only guarantee is that the $\ell_2$ norm of the error vector is decreased at every iteration step. In general, there is no guarantee that the algorithm can obtain a solution close to the true one; (see, for example, [38]). However, under certain constraints on the structure of $X$, performance bounds can be obtained; see, for example, [37, 115, 123].
- The complexity of the algorithm amounts to $\mathcal{O}(k_0 lN)$ operations, which are contributed by the computations of the correlations, plus the demands raised by the solution of the LS task in Step 3, whose complexity depends on the specific algorithm used. The $k_0$ is the sparsity level of the delivered solution and, hence, the total number of iteration steps that are performed.

Another more qualitative argument that justifies the selection of the columns based on their correlation with the error vector is the following. Assume that the matrix $X$ is orthonormal. Let $\boldsymbol{y} = X\boldsymbol{\theta}$. Then, $\boldsymbol{y}$ lies in the subspace spanned by the active columns of X, that is, those that correspond to the nonzero components of $\boldsymbol{\theta}$. Hence, the rest of the columns are orthogonal to $\boldsymbol{y}$, because $X$ is assumed to be orthonormal. Taking the correlation of $\boldsymbol{y}$, at the first iteration step, with all the columns, it is certain that one among the active columns will be chosen. The inactive columns result in zero correlation. A similar argument holds true for all subsequent steps, because all the activity takes place in a subspace that is orthogonal to all the inactive columns of $X$. In the more general case, where $X$ is not orthonormal, we can still use the correlation as a measure that quantifies geometric similarity. The smaller the

correlation/magnitude of the inner product, the more orthogonal the two vectors. This brings us back to the notion of mutual coherence, which is a measure of the maximum correlation (least angle) among the columns of $X$.

### *OMP can recover optimal sparse solutions: sufficiency condition*

We have already stated that, in general, there are no guarantees that OMP will recover optimal solutions. However, when the unknown vector is sufficiently sparse, with respect to the structure of the sensing matrix $X$, then OMP can exactly solve the $\ell_0$ minimization task in (9.20) and recover the solution in $k_0$ steps, where $k_0$ is the sparsest solution that satisfies the associated linear set of equations.

**Theorem 10.1.** *Let the mutual coherence (Section 9.6.1) of the sensing matrix, $X$, be $\mu(X)$. Assume, also, that the linear system, $\boldsymbol{y} = X\boldsymbol{\theta}$, accepts a solution such as*

$$\|\boldsymbol{\theta}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(X)}\right). \tag{10.1}$$

*Then, OMP guarantees recovery of the sparsest solution in $k_0 = \|\boldsymbol{\theta}\|_0$ steps.*

We know from Section 9.6.1 that under the previous condition, any other solution will be necessarily less sparse. Hence, there is a unique way to represent $\boldsymbol{y}$ in terms of $k_0$ columns of $X$. Without harming generality, let us assume that the true support corresponds to the first $k_0$ columns of $X$, that is,

$$\boldsymbol{y} = \sum_{j=1}^{k_0} \theta_j \mathbf{x}_j, \quad \theta_j \neq 0, \quad \forall j \in \{1, \ldots, k_0\}.$$

The theorem is a direct consequence of the following proposition.

**Proposition 10.1.** *If the condition (10.1) holds true, then the OMP algorithm will never select a column with index outside the true support; see, for example, [115], (Problem 10.2). In a more formal way, this is expressed as*

$$j_i = \arg\max_{j=1,2,\ldots,l} \frac{\left|\mathbf{x}_j^{\mathrm{T}} \boldsymbol{e}^{(i-1)}\right|}{\|\mathbf{x}_j\|_2} \in \{1, \ldots, k_0\}.$$

A geometric interpretation of this proposition is the following: if the angles formed between all the possible pairs among the columns of $X$ close to $90^o$ (columns almost orthogonal) in the $\mathbb{R}^l$ space, which guarantees that $\mu(X)$ is small enough, then $\boldsymbol{y}$ will lean more (form a smaller angle) toward any one of the active columns that contribute to its formation, compared to the rest that are inactive and do not participate in the linear combination that generates $\boldsymbol{y}$. Figure 10.2 illustrates the geometry, for the extreme case of mutually orthogonal vectors (Figure 10.2a), and for the more general case where the vectors are not orthogonal, yet the angle between any pair of columns is close enough to $90^o$ (Figure 10.2b).

In a nutshell, the previous proposition guarantees that, during the first iteration, a column corresponding to the true support will be selected. In a similar way, this is also true for all subsequent iterations. In the second step, another column, different from the previously selected column (as has already been stated), will be chosen. At step $k_0$, the last remaining active column corresponding to the true support is selected, and this necessarily results to zero error. To this end, it suffices to set $\epsilon_0$ equal to zero.

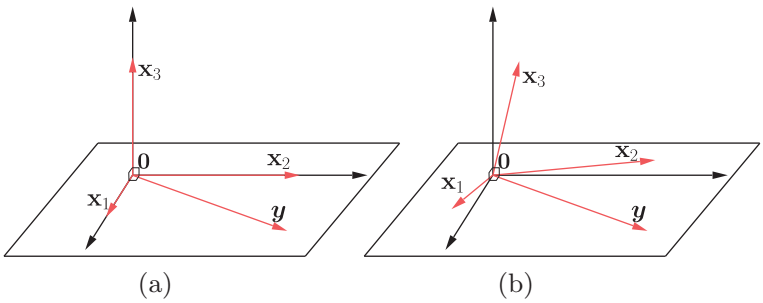


**FIGURE 10.2**

(a) In the case of an orthogonal matrix, the measurement vector $\boldsymbol{y}$ will be orthogonal to any inactive column; here, $\mathbf{x}_3$. (b) In the more general case, it is expected to "lean" closer (form smaller angles) to the active than to the inactive columns.

### *The LARS algorithm*

The least angle regression (LARS) algorithm, [48], shares the first two steps with OMP. It selects $j_i$ to be an index outside the currently available active set in order to maximize the correlation with the residual vector. However, instead of performing an LS fit to compute the nonzero components of $\boldsymbol{\theta}^{(i)}$, these are computed so that the residual will be equicorrelated with all the columns in the active set, that is,

$$|\mathbf{x}_j^{\mathrm{T}}(\boldsymbol{y} - X\boldsymbol{\theta}^{(i)})| = \text{constant}, \quad \forall j \in S^{(i)},$$

where we have assumed that the columns of $X$ are normalized, as is common in practice (recall, also, the Remarks 9.4). In other words, in contrast to the OMP, where the error vector is forced to be orthogonal to the active columns, LARS demands this error form equal angles with each one of them. Like OMP, it can be shown that, provided the target vector is sufficiently sparse and under incoherence of the columns of $X$, LARS can exactly recover the sparsest solution, [116].

A further small modification leads to the LARS-LASSO algorithm. According to this version, a previously selected index in the active set can be removed at a later stage. This gives the algorithm the potential to "recover" from a previously bad decision. Hence, this modification departs from the strict rationale that defines the greedy algorithms. It turns out that this version solves the LASSO optimization task. This algorithm is the same as the one suggested in [99] and it is known as a *homotopy* algorithm. Homotopy methods are based on a continuous transformation from one optimization task to another. The solutions to this sequence of tasks lie along a continuous parameterized path. The idea is that while the optimization tasks may be difficult to solve by themselves, one can trace this path of solutions by slowly varying the parameters. For the LASSO task, it is the $\lambda$ parameter that is varying; see, for example, [4, 86, 104]. Take as an example the LASSO task in its regularized version in (9.6). For $\lambda = 0$, the task minimizes the $\ell_2$ error norm and for $\lambda \longrightarrow \infty$ the task minimizes the parameter vector's $\ell_1$ norm, and for this case the solution tends to zero. It turns out that the solution path, as $\lambda$ changes from large to small values, is polygonal. Vertices on this solution path correspond to vectors having nonzero elements only on a subset of entries. This subset remains unchanged until $\lambda$ reaches the next critical value, which corresponds to a new vertex of the polygonal path and to a new subset of potential nonzero values. Thus, the solution is obtained via this sequence of steps along this polygonal path.

### *Compressed sensing matching pursuit (CSMP) algorithms*

Strictly speaking, the algorithms to be discussed here algorithms are not greedy, yet as stated in [93], they are at heart greedy algorithms. Instead of performing a single term optimization per iteration step, in order to increase the support by one, as is the case with OMP, these algorithms attempt to obtain first an estimate of the support and then use this information to compute a LS estimate of the target vector, constrained on the respective active columns. The quintessence of the method lies in the near-orthogonal nature of the sensing matrix, assuming that this obeys the RIP condition.

Assume that $X$ obeys the RIP for some small enough value $\delta_k$ and sparsity level, $k$, of the unknown vector. Let, also, that the measurements are exact, that is, $\boldsymbol{y} = X\boldsymbol{\theta}$. Then, $X^{\mathrm{T}}\boldsymbol{y} = X^{\mathrm{T}}X\boldsymbol{\theta} \approx \boldsymbol{\theta}$, due to the near-orthogonal nature of $X$. Therefore, intuition indicates that it is not unreasonable to select, in the first iteration step, the $t$ (a user-defined parameter) largest in magnitude components of $X^{\mathrm{T}}\boldsymbol{y}$ as indicative of the nonzero positions of the sparse target vector. This reasoning carries on for all subsequent steps, where, at the $i$th iteration, the place of $\boldsymbol{y}$ is taken by the residual $\boldsymbol{e}^{(i-1)} := \boldsymbol{y} - X\boldsymbol{\theta}^{(i-1)}$, where $\boldsymbol{\theta}^{(i-1)}$ indicates the estimate of the target vector at the $(i-1)$th iteration. Basically, this could be considered as a generalization of the OMP. However, as we will soon see, the difference between the two mechanisms is more substantial.

**Algorithm 10.2 (The CSMP scheme).**

1. Select the value of $t$.
2. Initialize the algorithm: $\boldsymbol{\theta}^{(0)} = \mathbf{0}$, $\boldsymbol{e}^{(0)} = \boldsymbol{y}$.
3. For $i = 1, 2, \ldots$, execute the following;
   - **(a)** Obtain the current support:

$$S^{(i)} := \mathrm{supp}\left(\boldsymbol{\theta}^{(i-1)}\right) \cup \left\{ \begin{array}{c} \text{indices of the } t \text{ largest in magnitude} \\ \text{components of } X^{\mathrm{T}}\boldsymbol{e}^{(i-1)} \end{array} \right\}.$$

   - **(b)** Select the active columns: Construct $X^{(i)}$ to comprise the active columns of $X$ in accordance to $S^{(i)}$. Obviously, $X^{(i)}$ is an $N \times r$ matrix, where $r$ denotes the cardinality of the support set $S^{(i)}$.
   - **(c)** Update the estimate of the parameter vector: solve the LS task

$$\tilde{\boldsymbol{\theta}} := \arg\min_{z \in \mathbb{R}^r} \left\| \boldsymbol{y} - X^{(i)}\boldsymbol{z} \right\|_2^2.$$

     Obtain $\hat{\boldsymbol{\theta}}^{(i)} \in \mathbb{R}^l$ having the $r$ elements of $\tilde{\boldsymbol{\theta}}$ in the respective locations, as indicated by the support, and the rest of the elements being zero.
   - **(d)** $\boldsymbol{\theta}^{(i)} := H_k\left(\hat{\boldsymbol{\theta}}^{(i)}\right)$. The mapping $H_k$ denotes the *hard thresholding* function; that is, it returns a vector with the $k$ largest in magnitude components of the argument, and the rest are forced to zero.
   - **(e)** Update the error vector: $\boldsymbol{e}^{(i)} = \boldsymbol{y} - X\boldsymbol{\theta}^{(i)}$.

The algorithm requires as input the sparsity level $k$. Iterations carry on until a halting criterion is met. The value of $t$, which determines the largest in magnitude values in Steps 1 and 3a, depends on the specific algorithm. In CoSaMP (compressive sampling matching pursuit [93]), $t = 2k$ (Problem 10.3), and in the SP (subspace pursuit [33]), $t = k$.

Having stated the general scheme, a major difference with OMP becomes readily apparent. In OMP, only one column is selected per iteration step. Moreover, this remains in the active set for all subsequent steps. If, for some reason, this was not a good choice, the scheme cannot recover from such a bad

decision. In contrast, the support and, hence, the active columns of $X$ are continuously updated in CSMP, and the algorithm has the ability to correct a previously bad decision, as more information is accumulated and iterations progress. In [33], it is shown that if the measurements are exact ($\boldsymbol{y} = X\boldsymbol{\theta}$), then SP can recover the $k$-sparse true vector in a finite number of iteration steps, provided that $X$ satisfies the RIP with $\delta_{3k} < 0.205$. If the measurements are noisy, performance bounds have been derived, which hold true for $\delta_{3k} < 0.083$. For the CoSaMP, performance bounds have been derived for $\delta_{4k} < 0.1$.

### 10.2.2 ITERATIVE SHRINKAGE/THRESHOLDING (IST) ALGORITHMS

This family of algorithms also have a long history; see, for example, [44, 69, 70, 73]. However, in the "early" days, most of the developed algorithms had some sense of heuristic flavor, without establishing a clear bridge with optimizing a cost function. Later attempts were substantiated by sound theoretical arguments concerning issues such as convergence and convergence rate [31, 34, 50, 56].

The general form of this algorithmic family has a striking resemblance to the classical linear algebra iterative schemes for approximating the solution of large linear systems of equations, known as *stationary iterative* or *iterative relaxation* methods. The classical Gauss-Seidel and Jacobi algorithms (e.g., [65]), in numerical analysis can be considered members of this family. Given a linear system of $l$ equations with $l$ unknowns, $\boldsymbol{z} = A\boldsymbol{x}$, the basic iteration at step $i$ has the following form:

$$\begin{aligned}\boldsymbol{x}^{(i)} &= (I - QA)\,\boldsymbol{x}^{(i-1)} + Q\boldsymbol{z}\\ &= \boldsymbol{x}^{(i-1)} + Q\boldsymbol{e}^{(i-1)}, \qquad \boldsymbol{e}^{(i-1)} := \boldsymbol{z} - A\boldsymbol{x}^{(i-1)},\end{aligned}$$

which does not come as a surprise. It is of the same form as most of the iterative schemes for numerical solutions! The matrix $Q$ is chosen in order to guarantee convergence, and different choices lead to different algorithms with their pros and cons. It turns out that this algorithmic form can also be applied to underdetermined systems of equations, $\boldsymbol{y} = X\boldsymbol{\theta}$, with a "minor" modification, which is imposed by the sparsity constraint of the target vector. This leads to the following general form of iterative computation:

$$\boldsymbol{\theta}^{(i)} = T_i\left(\boldsymbol{\theta}^{(i-1)} + Q\boldsymbol{e}^{(i-1)}\right), \qquad \boldsymbol{e}^{(i-1)} = \boldsymbol{y} - X\boldsymbol{\theta}^{(i-1)},$$

starting from an initial guess of $\boldsymbol{\theta}^{(0)}$ (usually $\boldsymbol{\theta}^{(0)} = \boldsymbol{0}$, $\boldsymbol{e}^{(0)} = \boldsymbol{y}$). In certain cases, $Q$ can be made to be iteration-dependent. The function $T_i(\cdot)$ is a nonlinear thresholding function that is applied *entrywise*, that is, *component-wise*. Depending on the specific scheme, this can be either the hard thresholding function, denoted as $H_k$, or the soft thresholding function, denoted as $S_\alpha$. Hard thresholding, as we already know, keeps the $k$ largest components of a vector unaltered and sets the rest equal to zero. Soft thresholding was introduced in Section 9.3. All components with magnitude less than a threshold value, $\alpha$, are forced to zero and the rest are reduced in magnitude by $\alpha$; that is, the $j$th component of a vector, $\boldsymbol{\theta}$, after soft thresholding becomes

$$(S_\alpha(\boldsymbol{\theta}))_j = \text{sgn}(\theta_j)(|\theta_j| - \alpha)_+.$$

Depending on (a) the choice of $T_i$, (b) the specific value of the parameter $k$ or $\alpha$, and (c) the matrix $Q$, different instances occur. The most common choice for $Q$ is $\mu X^{\mathrm{T}}$, and the generic form of the main iteration becomes

$$\boxed{\boldsymbol{\theta}^{(i)} = T_i\left(\boldsymbol{\theta}^{(i-1)} + \mu X^{\mathrm{T}}\boldsymbol{e}^{(i-1)}\right),} \tag{10.2}$$

where $\mu$ is a relaxation (user-defined) parameter, which can also be left to vary with each iteration step. The choice of $X^{\mathrm{T}}$ is intuitively justified, once more, by the near-orthogonal nature of $X$. For the first iteration step and for a linear system of the form $\boldsymbol{y} = X\boldsymbol{\theta}$, starting from a zero initial guess, we have $X^{\mathrm{T}}\boldsymbol{y} = X^{\mathrm{T}}X\boldsymbol{\theta} \approx \boldsymbol{\theta}$ and we are close to the solution.

Although intuition is most important in scientific research, it is not enough, by itself, to justify decisions and actions. The generic scheme in (10.2) has been reached from different paths, following different perspectives that lead to different choices of the involved parameters. Let us spend some more time on that, with the aim of making the reader more familiar with techniques that address optimization tasks of nondifferentiable loss functions. The term in the parenthesis in (10.2) coincides with the gradient descent iteration step if the cost function were the unregularized LS loss, that is,

$$J(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{y} - X\boldsymbol{\theta}\|_2^2.$$

In this case, the gradient descent rationale leads to

$$\begin{aligned}\boldsymbol{\theta}^{(i-1)} - \mu\frac{\partial J\left(\boldsymbol{\theta}^{(i-1)}\right)}{\partial \boldsymbol{\theta}} &= \boldsymbol{\theta}^{(i-1)} - \mu X^{\mathrm{T}}(X\boldsymbol{\theta}^{(i-1)} - \boldsymbol{y}) \\ &= \boldsymbol{\theta}^{(i-1)} + \mu X^{\mathrm{T}}\boldsymbol{e}^{(i-1)}.\end{aligned}$$

The gradient descent can alternatively be viewed as the result of minimizing a regularized version of the linearized cost function (verify it),

$$\begin{aligned}\boldsymbol{\theta}^{(i)} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^l}\Bigg\{J\left(\boldsymbol{\theta}^{(i-1)}\right) + \left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\right)^{\mathrm{T}}\frac{\partial J\left(\boldsymbol{\theta}^{(i-1)}\right)}{\partial \boldsymbol{\theta}} \\ + \frac{1}{2\mu}\left\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\right\|_2^2\Bigg\}.\end{aligned} \tag{10.3}$$

One can adopt this view of the gradient descent philosophy as a kick-off point to minimize iteratively the following LASSO task,

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^l}\left\{L(\boldsymbol{\theta},\lambda) = \frac{1}{2}\|\boldsymbol{y} - X\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1 = J(\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}\|_1\right\}.$$

The difference now is that the loss function comprises two terms: one that is smooth (differentiable) and a nonsmooth one. Let the current estimate be $\boldsymbol{\theta}^{(i-1)}$. The updated estimate is obtained by

$$\begin{aligned}\boldsymbol{\theta}^{(i)} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^l}\Bigg\{J\left(\boldsymbol{\theta}^{(i-1)}\right) + \left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\right)^{\mathrm{T}}\frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}} \\ + \frac{1}{2\mu}\left\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\right\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1\Bigg\},\end{aligned}$$

which, after ignoring constants, is equivalently written as

$$\boldsymbol{\theta}^{(i)} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^l}\left\{\frac{1}{2}\left\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\right\|_2^2 + \lambda\mu\|\boldsymbol{\theta}\|_1\right\} \tag{10.4}$$

where

$$\tilde{\boldsymbol{\theta}} := \boldsymbol{\theta}^{(i-1)} - \mu\frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}}. \tag{10.5}$$

Following exactly the same steps as those that led to the derivation of (9.13) from (9.6) (after replacing $\hat{\boldsymbol{\theta}}_{LS}$ with $\tilde{\boldsymbol{\theta}}$), we obtain

$$\boldsymbol{\theta}^{(i)} = S_{\lambda\mu}(\tilde{\boldsymbol{\theta}}) = S_{\lambda\mu}\left(\boldsymbol{\theta}^{(i-1)} - \mu\frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial\boldsymbol{\theta}}\right) \tag{10.6}$$

$$= S_{\lambda\mu}\left(\boldsymbol{\theta}^{(i-1)} + \mu X^{\mathrm{T}}\boldsymbol{e}^{(i-1)}\right). \tag{10.7}$$

This is very interesting and practically useful. The only effect of the presence of the nonsmooth $\ell_1$ norm in the loss function is an extra simple thresholding operation, which as we know is an operation performed *individually* on each component. It can be shown (e.g., [11, 95]), that this algorithm converges to a minimizer $\boldsymbol{\theta}_*$ of the LASSO (9.6), provided that $\mu \in (0, 1/\lambda_{\max}(X^{\mathrm{T}}X))$, where $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of $X^{\mathrm{T}}X$. The convergence rate is dictated by the rule

$$L(\boldsymbol{\theta}^{(i)}, \lambda) - L(\boldsymbol{\theta}_*, \lambda) \approx O(1/i),$$

which is known as *sublinear* global rate of convergence. Moreover, it can be shown that

$$L(\boldsymbol{\theta}^{(i)}, \lambda) - L(\boldsymbol{\theta}_*, \lambda) \leq \frac{C\left\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}_*\right\|_2^2}{2i}.$$

The latter result indicates that if one wants to achieve an accuracy of $\epsilon$, then this can be obtained by at most $\left\lfloor \frac{C\left\|\boldsymbol{\theta}^{(0)}-\boldsymbol{\theta}_*\right\|_2^2}{2\epsilon} \right\rfloor$ iterations, where $\lfloor\cdot\rfloor$ denotes the floor function.

In [34], (10.2) was obtained from a nearby corner, building upon arguments from the classical *proximal-point* methods in optimization theory (e.g., [105]). The original LASSO regularized cost function is modified to the *surrogate objective*,

$$J(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = \frac{1}{2}\|\boldsymbol{y} - X\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1 + \frac{1}{2}d(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}),$$

where

$$d(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) := c\left\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\right\|_2^2 - \left\|X\boldsymbol{\theta} - X\tilde{\boldsymbol{\theta}}\right\|_2^2.$$

If $c$ is appropriately chosen (larger than the largest eigenvalue of $X^{\mathrm{T}}X$), the surrogate objective is guaranteed to be strictly convex. Then it can be shown (Problem 10.4) that the minimizer of the surrogate objective is given by

$$\hat{\boldsymbol{\theta}} = S_{\lambda/c}\left(\tilde{\boldsymbol{\theta}} + \frac{1}{c}X^{\mathrm{T}}(\boldsymbol{y} - X\tilde{\boldsymbol{\theta}})\right). \tag{10.8}$$

In the iterative formulation, $\tilde{\boldsymbol{\theta}}$ is selected to be the previously obtained estimate; in this way, one tries to keep the new estimate close to the previous one. The procedure readily results to our generic scheme in (10.2), using soft thresholding with parameter $\lambda/c$. It can be shown that such a strategy converges to a minimizer of the original LASSO problem. The same algorithm was reached in [56], using *majorization-minimization* techniques from optimization theory. So, from this perspective, the IST family has strong ties with algorithms that belong to the convex optimization category.

In [118], the *sparse reconstruction by separable approximation* (SpaRSA) algorithm is proposed, which is a modification of the standard IST scheme. The starting point is (10.3); however, the

multiplying factor, $\frac{1}{2\mu}$, instead of being constant is now allowed to change from iteration to iteration according to a rule. This results in a speedup in the convergence of the algorithm. Moreover, inspired by the homotopy family of algorithms, where $\lambda$ is allowed to vary, SpaRSA can be extended to solve a sequence of problems that are associated with a corresponding sequence of values of $\lambda$. Once a solution has been obtained for a particular value of $\lambda$, it can be used as a "warm-start" for a nearby value. Solutions can therefore be computed for a range of values, at a small extra computational cost, compared to solving for a single value from a "cold start." This technique abides with the *continuation strategy*, which has been used in the context of other algorithms as well (e.g., [66]). Continuation has been shown to be a very successful tool to increase the speed of convergence.

An interesting variation of the basic IST scheme has been proposed in [11], which improves the convergence rate to $O(1/t^2)$, by only a simple modification with almost no extra computational burden. The scheme is known as *fast iterative shrinkage-thresholding algorithm* (FISTA). This scheme is an evolution of [96], which introduced the basic idea for the case of differentiable costs, and consists of the following steps:

$$\boldsymbol{\theta}^{(i)} = S_{\lambda\mu}\left(\boldsymbol{z}^{(i)} + \mu X^{\mathrm{T}}\left(\boldsymbol{y} - X\boldsymbol{z}^{(i)}\right)\right),$$

$$\boldsymbol{z}^{(i+1)} := \boldsymbol{\theta}^{(i)} + \frac{t_i - 1}{t_{i+1}}\left(\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}^{(i-1)}\right),$$

where

$$t_{i+1} := \frac{1 + \sqrt{1 + 4t_i^2}}{2},$$

with initial points $t_1 = 1$ and $\boldsymbol{z}^{(1)} = \boldsymbol{\theta}^{(0)}$. In words, in the thresholding operation, $\boldsymbol{\theta}^{(i-1)}$ is replaced by $\boldsymbol{z}^{(i)}$, which is a specific linear combination of two successive updates of $\boldsymbol{\theta}$. Hence, at a marginal increase of the computational cost, a substantial increase in convergence speed is achieved.

In [17] the hard thresholding version has been used, with $\mu = 1$, and the thresholding function $H_k$ uses the sparsity level $k$ of the target solution that is assumed to be known. In a later version, [19], the relaxation parameter is left to change so that, at each iteration step, the error is maximally reduced. It has been shown that the algorithm converges to a local minimum of the cost function $\|\boldsymbol{y} - X\boldsymbol{\theta}\|_2$, under the constraint that $\boldsymbol{\theta}$ is a $k$-sparse vector. Moreover, the latter version is a stable one and it results to a near optimal solution if a form of RIP is fulfilled.

A modified version of the generic scheme given in (10.2), which evolves along the lines of [84], obtains the updates component-wise, one vector component at a time. Thus, a "full" iteration consists of $l$ steps. The algorithm is known as *coordinate descent* and its basic iteration has the form (Problem 10.5),

$$\theta_j^{(i)} = S_{\lambda/\|\mathbf{x}_j\|_2^2}\left(\theta_j^{(i-1)} + \frac{\mathbf{x}_j^{\mathrm{T}}\boldsymbol{e}^{(i-1)}}{\|\mathbf{x}_j\|_2^2}\right), \quad j = 1, 2, \ldots, l. \tag{10.9}$$

This algorithm replaces the constant $c$, in the previously reported soft thresholding algorithm, with the norm of the respective column of $X$, if the columns of $X$ are not normalized to unit norm. It has been shown that the parallel coordinate descent algorithm also converges to a LASSO minimizer of (9.6) [50]. Improvements of the algorithm, using line search techniques to determine the most descent direction for each iteration, have also been proposed; see [124].

The main contribution to the complexity for the iterative shrinkage algorithmic family comes from the two matrix-vector products, which amounts to $\mathcal{O}(Nl)$, unless $X$ has a special structure (e.g., DFT), that can be exploited to reduce the load.

In [85], the two stage thresholding (TST) scheme is presented, which brings together arguments from the iterative shrinkage family and the OMP. This algorithmic scheme involves two stages of thresholding. The first step is exactly the same as in (10.2). However, this is now used only for determining "significant" nonzero locations, just as in compressed sensing matching pursuit (CSMP) algorithms, presented in the previous subsection. Then, an LS problem is solved to provide the updated estimate, under the constraint of the available support. This is followed by a second step of thresholding. The thresholding operations in the two stages can be different. If hard thresholding, $H_k$, is used in both steps, this results to the algorithm proposed in [58]. For this latter scheme, convergence and performance bounds are derived if the RIP holds for $\delta_{3k} < 0.58$. In other words, the basic difference between the TST and CSMP approaches is that, in the latter case, the most significant nonzero coefficients are obtained by looking at the correlation term $X^{\mathrm{T}}\boldsymbol{e}^{(i-1)}$ and in the TST family at $\boldsymbol{\theta}^{(i-1)} + \mu X^{\mathrm{T}}\boldsymbol{e}^{(i-1)}$. The differences among different approaches can be minor and the crossing lines between the different algorithmic categories are not necessarily crispy and clear. However, from a practical point of view, sometimes small differences may lead to substantially improved performance.

In [41], the IST algorithmic framework was treated as a *message passing* algorithm in the context of graphical models (Chapter 15), and the following modified recursion was obtained:

$$\boldsymbol{\theta}^{(i)} = T_i\left(\boldsymbol{\theta}^{(i-1)} + X^{\mathrm{T}}\boldsymbol{z}^{(i-1)}\right), \tag{10.10}$$

$$\boldsymbol{z}^{(i-1)} = \boldsymbol{y} - X\boldsymbol{\theta}^{(i-1)} + \frac{1}{\alpha}\boldsymbol{z}^{(i-2)}\overline{T_i'\left(\boldsymbol{\theta}^{(i-2)} + X^{\mathrm{T}}\boldsymbol{z}^{(i-2)}\right)}, \tag{10.11}$$

where $\alpha = \frac{N}{l}$, the overbar denotes the average over all the components of the corresponding vector and $T_i'$ denotes the respective derivative of the component-wise thresholding rule. The extra term on the right-hand side in (10.11), which now appears, turns out to provide a performance improvement of the algorithm, compared to the IST family, with respect to the undersampling-sparsity trade-off (Section 10.2.3). Note that $T_i$ is iteration-dependent and it is controlled via the definition of certain parameters. A parameterless version of it has been proposed in [91]. A detailed treatment on the message passing algorithms can be found in [2].

*Remarks 10.2.*

- The iteration in (10.6) bridges the IST algorithmic family with another powerful tool in convex optimization, which builds upon the notion of *proximal mapping* or *Moreau envelopes* (see Chapter 8 and, e.g., [32, 105]). Given a convex function $h : \mathbb{R}^l \to \mathbb{R}$, and a $\mu > 0$, the proximal mapping, $\mathrm{Prox}_{\mu h} : \mathbb{R}^l \longmapsto \mathbb{R}^l$, with respect to $h$, and of index $\mu$, is defined as the (unique) minimizer

$$\mathrm{Prox}_{\mu h}(\boldsymbol{x}) := \arg\min_{\boldsymbol{v}\in\mathbb{R}^l}\left\{h(\boldsymbol{v}) + \frac{1}{2\mu}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2\right\}, \quad \forall \boldsymbol{x} \in \mathbb{R}^l. \tag{10.12}$$

  Let us now assume that we want to minimize a convex function, which is given as the sum

$$f(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + h(\boldsymbol{\theta}),$$

  where $J$ is convex and differentiable, and $h$ is also convex, but not necessarily smooth. Then, it can be shown (Section 8.14) that the following iterations converge to a minimizer of $f$,

$$\boldsymbol{\theta}^{(i)} = \text{Prox}_{\mu h}\left(\boldsymbol{\theta}^{(i-1)} - \mu \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}}\right), \tag{10.13}$$

where $\mu > 0$, and it can also be made iteration dependent, that is, $\mu_i > 0$. If we now use this scheme to minimize our familiar cost,

$$J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1,$$

we obtain (10.6); this is so because the proximal operator of $h(\boldsymbol{\theta}) := \lambda \|\boldsymbol{\theta}\|_1$ is shown [31, 32], Section 8.13 to be identical to the soft thresholding operator, that is,

$$\text{Prox}_h(\boldsymbol{\theta}) = S_\lambda(\boldsymbol{\theta}).$$

In order to feel more comfortable with this operator, note that if $h(\boldsymbol{x}) \equiv 0$, its proximal operator is equal to $\boldsymbol{x}$, and in this case (10.13) becomes our familiar gradient descent algorithm.

- All the nongreedy algorithms that have been discussed so far have been developed to solve the task defined in the formulation (9.6). This is mainly because this is an easier task to solve; once $\lambda$ has been fixed, it is an unconstrained optimization task. However, there are algorithms that have been developed to solve the alternative formulations.
  The NESTA algorithm has been proposed in [12] and solves the task in its (9.8) formulation. Adopting this path can have an advantage because $\epsilon$ may be given as an estimate of the uncertainty associated with the noise, which can readily be obtained in a number of practical applications. In contrast, selecting a priori the value for $\lambda$ is more intricate. In [28], the value $\lambda = \sigma_\eta \sqrt{2 \ln l}$, where $\sigma_\eta$ is the noise standard deviation, is argued to have certain optimality properties; however, this argument hinges on the assumption of the orthogonality of $X$. NESTA relies heavily on Nesterov's generic scheme [96], hence its name. The original Nesterov's algorithm performs a constrained minimization of a smooth convex function $f(\boldsymbol{\theta})$, that is,

$$\min_{\boldsymbol{\theta} \in Q} f(\boldsymbol{\theta}),$$

where $Q$ is a convex set, and in our case this is associated with the quadratic constraint in (9.8). The algorithm consists of three basic steps. The first one, involves an auxiliary variable, and is similar with the step in (10.3), i.e.,

$$\boldsymbol{w}^{(i)} = \arg\min_{\boldsymbol{\theta} \in Q} \left\{ \left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\right)^T \frac{\partial f\left(\boldsymbol{\theta}^{(i-1)}\right)}{\partial \boldsymbol{\theta}} + \frac{L}{2} \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)} \right\|_2^2 \right\}, \tag{10.14}$$

where $L$ is an upper bound on the Lipschitz coefficient, which the gradient of $f$ has to satisfy. The difference with (10.3) is that the minimization is now a constrained one. However, Nesterov has also added a second step involving another auxiliary variable, $\boldsymbol{z}^{(i)}$, which is computed in a similar way as $\boldsymbol{w}^{(i)}$, but the linearized term is now replaced by a weighted cumulative gradient,

$$\sum_{k=0}^{i-1} \alpha_k \left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\right)^T \frac{\partial f\left(\boldsymbol{\theta}^{(k)}\right)}{\partial \boldsymbol{\theta}}.$$

The effect of this term is to smooth out the "zigzagging" of the path toward the solution, whose effect is to increase significantly the convergence speed. The final step of the scheme involves an averaging of the previously obtained variables,

$$\boldsymbol{\theta}^{(i)} = t_i \boldsymbol{z}^{(i)} + (1 - t_i)\boldsymbol{w}^{(i)}.$$

The values of the parameters $\alpha_k$, $k = 0, \ldots, i-1$, and $t_i$ result from the theory so that convergence is guaranteed. As was the case with its close relative FISTA, the algorithm enjoys an $O(1/i^2)$ convergence rate. In our case, where the function to be minimized, $\|\boldsymbol{\theta}\|_1$, is not smooth, NESTA uses a smoothed prox-function of it. Moreover, it turns out that close-form updates are obtained for $\boldsymbol{z}^{(i)}$ and $\boldsymbol{w}^{(i)}$. If $X$ is chosen in order to have orthonormal rows, the complexity per iteration is $O(l)$ plus the computations needed for performing the product $X^{\mathrm{T}}X$, which is the most computationally thirsty part. However, this complexity can substantially be reduced if the sensing matrix is chosen to be a submatrix of a unitary transform, which admits fast matrix-vector product computations (e.g., a subsampled DFT matrix). For example, for the case of a subsampled DFT matrix, the complexity amounts to $O(l)$ plus the load to perform the two fast fourier transforms (FFT). Moreover, the continuation strategy can also be employed to accelerate convergence. In [12], it is demonstrated that NESTA exhibits good accuracy results, while retaining a complexity that is competitive with algorithms developed around the (9.6) formulation and scales in an affordable way for large-size problems. Furthermore, NESTA, and in general Nesterov's scheme, enjoy a generality that allows their use for other optimization tasks as well.

- The task in (9.7) has been considered in [14] and [99]. In the former, the algorithm comprises a projection on the $\ell_1$ ball $\|\boldsymbol{\theta}\|_1 \leq \rho$ (see also Section 10.4.4) per iteration step. The most computationally dominant part of the algorithm consists of matrix-vector products. In [99], a homotopy algorithm is derived for the same task, where now the bound $\rho$ becomes the homotopy parameter that is left to vary. This algorithm is also referred to as the LARS-LASSO, as has already been reported.

### 10.2.3 WHICH ALGORITHM?: SOME PRACTICAL HINTS

We have already discussed a number of algorithmic alternatives to obtain solutions to the $\ell_0$ or $\ell_1$ norm minimization tasks. Our focus was on schemes whose computational demands are rather low and that scale well to very large problem sizes. We have not touched more expensive methods such as interior point methods for solving the $\ell_1$ convex optimization task. A review of such methods is provided in [72]. Interior point methods evolve along the Newton-type recursion and their complexity per iteration step is at least of the order $\mathcal{O}(l^3)$. As is most often the case, there is a trade-off. Schemes of higher complexity tend to result in enhanced performance. However, such schemes become impractical in problems of large size. Some examples of other algorithms that were not discussed can be found in [14, 35, 118, 121]. Talking about complexity, it has to be pointed out that what really matters at the end is not so much the complexity per iteration step, but the overall required resources in computer time/memory for the algorithm to converge to a solution within a specified accuracy. For example, an algorithm may be of low complexity per iteration step, but it may need an excessive number of iterations to converge.

Computational load is only one among a number of indices that characterize the performance of an algorithm. Throughout the book so far, we have considered a number of other performance measures, such as convergence rate, tracking speed (for the adaptive algorithms), and stability with respect to the presence of noise and/or finite word length computations. No doubt, all these performance measures are also of interest here, too. However, there is an additional aspect that is of particular importance when quantifying performance of sparsity-promoting algorithms. This is related to the *undersampling-sparsity trade-off* or the *phase transition curve*.

One of the major issues on which we focused in Chapter 9 was to derive and present the conditions that guarantee uniqueness of the $\ell_0$ minimization and its equivalence with the $\ell_1$ minimization task, under an underdetermined set of measurements/observations, $\boldsymbol{y} = X\boldsymbol{\theta}$, for the recovery of sparse enough signals/vectors. While discussing the various algorithms in this section, we reported a number of different RIP-related conditions that some of the algorithms have to satisfy in order to recover the target sparse vector. As a matter of fact, it has to be admitted that this was quite confusing, because each algorithm had to satisfy its own conditions. In addition, in practice, these conditions are not easily to be verified. Although such results are no doubt important to establish convergence, and make us more confident, and help us better understand why and how an algorithm works, one needs further experimental evidence in order to establish good performance bounds for an algorithm. Moreover, all the conditions we have dealt with, including coherence and RIP, are sufficient conditions. In practice, it turns out that sparse signal recovery is possible with sparsity levels much higher than those predicted by the theory, for given $N$ and $l$. Hence, proposing a new algorithm or selecting an algorithm from an available palette, one has to demonstrate experimentally the range of sparsity levels that can be recovered by the algorithm, as a percentage of the number of measurements and the dimensionality. Thus, in order to select an algorithm, one should cast her/his vote for the algorithm that, for given $l$ and $N$, has the potential to recover $k$-sparse vectors with $k$ being as high as possible for most of the cases, that is, with *high probability*.

Figure 10.3 illustrates the type of curve that is expected to result in practice. The vertical axis is the probability of exact recovery of a target $k$-sparse vector and the horizontal axis shows the ratio $k/N$, for a given number of measurements, $N$, and the dimensionality of the ambient space, $l$. Three curves are shown. The red ones correspond to the same algorithm, for two different values of the dimensionality, $l$, and the gray one corresponds to another algorithm. Curves of this shape are expected to result from experiments of the following setup. Assume that we are given a sparse vector, $\boldsymbol{\theta}_o$, with $k$ nonzero components in the $l$-dimensional space. Using a sensing matrix $X$, we generate $N$ measurements $\boldsymbol{y} = X\boldsymbol{\theta}_o$. The experiment is repeated a number of $M$ times, each time using a different realization of the sensing matrix and a different $k$-sparse vector. For each instance, the algorithm is run to recover

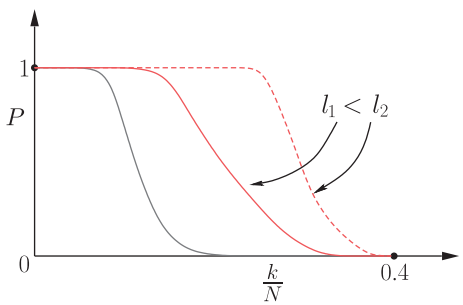


**FIGURE 10.3**

For any algorithm, the transition between the regions of 100% success and of complete failure is very sharp. For the algorithm corresponding to the red curve, this transition occurs at higher sparsity values and, from this point of view, it is a better algorithm than the one associated with the gray curve. Also, given an algorithm, the higher the dimensionality the higher the sparsity level where this transition occurs, as indicated by the two red curves.

the target sparse vector. This is not always possible. We count the number, $m$, of successful recoveries, and compute the corresponding percentage of successful recovery (probability), $m/M$, which is plotted on the vertical axis of Figure 10.3. The procedure is repeated for a different value of $k$, $1 \leq k \leq N$. A number of issues now jump onto the stage: (a) how one selects the ensemble of sensing matrices and (b) how one selects the ensemble of sparse vectors. There are different scenarios, and some typical examples are described next.

1. The $N \times l$ sensing matrices $X$ are formed by:
   - **(a)** Different i.i.d. realizations with elements drawn from a Gaussian $\mathcal{N}(0, 1/N)$.
   - **(b)** Different i.i.d. realizations from the uniform distribution on the unit sphere in $\mathbb{R}^N$, which is also known as the uniform spherical ensemble.
   - **(c)** Different i.i.d. realizations with elements drawn from Bernoulli type distributions.
   - **(d)** Different i.i.d. realizations of partial Fourier matrices, each time using a different set of $N$ rows.
2. The $k$-sparse target vector $\boldsymbol{\theta}_o$ is formed by selecting the locations of (at most) $k$ nonzero elements randomly, by "tossing a coin" with probability $p = k/l$, and filling the values of the nonzero elements according to a statistical distribution (e.g., Gaussian, uniform, double exponential, Cauchy).

Other scenarios are also possible. Some authors set all nonzero values to one [16], or to $\pm 1$, with the randomness imposed on the choice of the sign. It must be stressed that the performance of an algorithm may vary significantly under different experimental scenarios, and this may be indicative of the stability of an algorithm. In practice, a user may be interested in a specific scenario that is more representative of the available data.

Looking at Figure 10.3, the following conclusions are in order. In all curves, there is a sharp transition between two levels, from the 100% success to the 0% success. Moreover, the higher the dimensionality, the sharper the transition. This has also been shown theoretically in [40]. For the algorithm corresponding to the red curves, this transition occurs at higher values of $k$, compared to the algorithm that generates the curve drawn in gray. Provided that the computational complexity of the "red" algorithm can be accommodated by the resources that are available for a specific application, this seems to be the more sensible choice between the two algorithms. However, if the resources are limited, concessions are unavoidable.

Another way to "interrogate" and demonstrate the performance of an algorithm, with respect to its robustness to the range of values of sparsity levels that can be successfully recovered, is via the *phase transition curve*. To this end define

- $\alpha := \frac{N}{l}$, which is a normalized measure of the problem indeterminacy,
- $\beta := \frac{k}{N}$, which is a normalized measure of sparsity.

In the sequel, plot a graph having $\alpha \in [0, 1]$ in the horizontal axis and $\beta \in [0, 1]$ in the vertical one. For each point, $(\alpha, \beta)$, in the $[0, 1] \times [0, 1]$ region, compute the probability of the algorithm to recover a $k$-sparse target vector. In order to compute the probability, one has to adopt one of the previously stated scenarios. In practice, one has to form a grid of points that cover densely enough the region $[0, 1] \times [0, 1]$ in the graph. Use a varying intensity level scale to color the corresponding $(\alpha, \beta)$ point. Black corresponds to probability one and red to probability zero. Figure 10.4 illustrates the type of graph that is expected to be recovered in practice for large values of $l$; that is, the transition from the

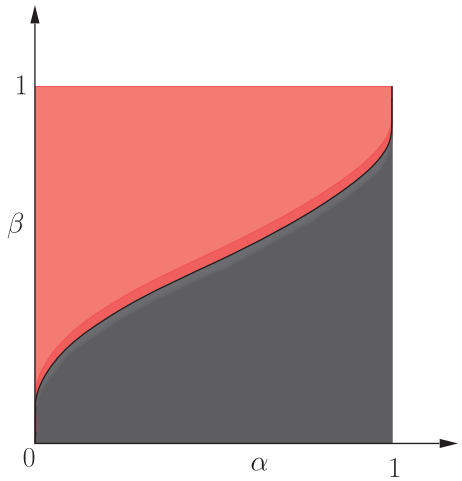


**FIGURE 10.4**

Typical phase transition behavior of a sparsity-promoting algorithm. Black corresponds to 100% success of recovering the sparsest solution, and red to 0%. For high-dimensional spaces, the transition is very sharp, as is the case in the figure. For lower dimensionality values, the transition from black to red is smoother and involves a region of varying color intensity.

region (phase) of "success" (black) to that of "fail" (red) is very sharp. As a matter of fact, there is a curve that separates the two regions. The theoretical aspects of this curve have been studied in the context of combinatorial geometry in [40] for the asymptotic case, $l \longrightarrow \infty$, and in [42] for finite values of $l$. Observe that the larger the value of $\alpha$ (larger percentage of measurements), the larger the value of $\beta$ at which the transition occurs. This is in line with what we have said so far in this chapter, and the problem gets increasingly difficult as one moves up and to the left in the graph. In practice, for smaller values of $l$, the transition region from red to black is smoother, and it gets narrower as $l$ increases. In such cases, one can draw an approximate curve that separates the "success" and "fail" regions, using regression techniques (see, e.g., [85]).

The reader may already be aware of the fact that, so far, we have avoided talking about the performance of individual algorithms. We have just discussed some "typical" behavior that algorithms tend to exhibit in practice. What the reader might have expected is to discuss comparative performance tests and draw related conclusions. We have not done that because we feel that it is too early in time to have "definite" performance conclusions, and this field is still in an early stage. Most authors compare their newly suggested algorithm with a few other algorithms, usually within a certain algorithmic family and, more important, under some specific scenarios, where the advantages of the newly suggested algorithm are documented. However, the performance of an algorithm can change significantly by changing the experimental scenario under which the tests are carried out. The most comprehensive comparative performance study so far has been carried out in [85]. However, even in this one, the scenario of exact measurements has been considered and there are no experiments concerning the robustness of individual algorithms to the presence of noise. It is important to say that this study involved

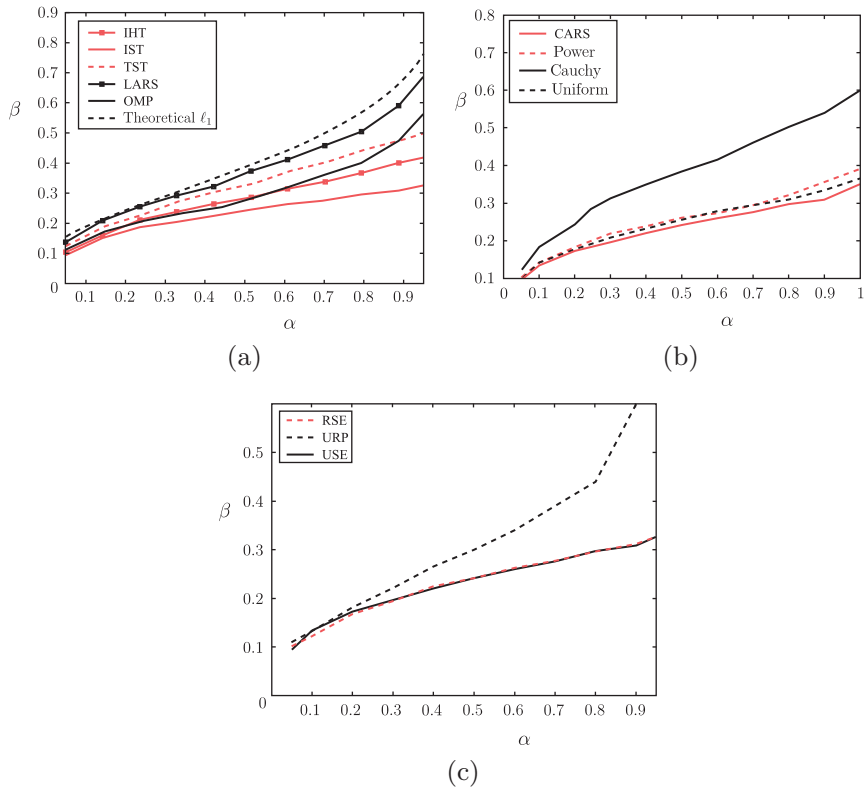


**FIGURE 10.5**

(a) The obtained phase transition curves for different algorithms under the same experimental scenario, together with the theoretical one. (b) Phase transition curve for the IST algorithm under different experimental scenarios for generating the target sparse vector. (c) The phase transition for the IST algorithms under different experimental scenarios for generating the sensing matrix $X$.

a huge effort of computation. We will comment on some of the findings from this study, which will also reveal to the reader that different experimental scenarios can significantly affect the performance of an algorithm.

Figure 10.5a shows the obtained phase transition curves for (a) the iterative hard thresholding (IHT); (b) the iterative soft thresholding (IST) scheme of (10.2); (c) the two-stage-thresholding (TST) scheme, as discussed earlier; (d) the LARS algorithm; and (e) the OMP algorithm, together with the theoretically obtained one using $\ell_1$ minimization. All algorithms were tuned with the optimal values, with respect to the required user-defined parameters, after extensive experimentation. The results in the figure correspond to the uniform spherical scenario for the generation of the sensing matrices. Sparse vectors were generated according to the $\pm 1$ scenario for the nonzero coefficients. The interesting observation is that, although the curves deviate from each other as they move to larger values of $\beta$, for smaller values, the differences in their performance become less and less. This is also true for computationally simple

schemes such as the IHT one. The performance of LARS is close to the optimal one. However, this comes at the cost of computational increase. The required computational time for achieving the same accuracy, as reported in [85], favors the TST algorithm. In some cases, LARS required excessively longer time to reach the same accuracy, in particular when the sensing matrix was the partial Fourier one and fast schemes to perform matrix vector products can be exploited. For such matrices, the thresholding schemes (IHT, IST, TST) exhibited a performance that scales very well to large-size problems.

Figure 10.5b indicates the phase transition curve for one of the algorithms (IST) as we change the scenarios for generating the sparse (target) vectors, using different distributions: (a) $\pm 1$, with equiprobable selection of signs (constant amplitude random selection (CARS)); (b) double exponential (power); (c) Cauchy; and (d) uniform in $[-1, 1]$. This is indicative and typical for other algorithms as well, with some of them being more sensitive than others. Finally, Figure 10.5c shows the transition curves for the IST algorithm by changing the sensing matrix generation scenario. Three curves are shown corresponding to (a) uniform spherical ensemble (USE); (b) random sign ensemble (RSE), where the elements are $\pm 1$ with signs uniformly distributed; and (c) the uniform random projection (URP) ensemble. Once more, one can observe the possible variations that are expected due to the use of different matrix ensembles. Moreover, changing ensembles affects each algorithm in a different way.

Concluding this section, it must be emphasized that the field of algorithmic development is still an ongoing research field, and it is early to come up with definite and concrete comparative performance conclusions. Moreover, besides the algorithmic front, existing theories often fall short in predicting what is observed in practice, with respect to their phase transition performance. For a related discussion, see, for example, [43].

## 10.3 VARIATIONS ON THE SPARSITY-AWARE THEME

In our tour so far, we have touched a number of aspects of sparsity-aware learning that come from mainstream theoretical developments. However, a number of variants have appeared and have been developed with the goal of addressing problems of a more special structure and/or proposing alternatives, which can be beneficial in boosting the performance in practice by serving the needs of specific applications. These variants focus either on the regularization term in (9.6), on the misfit-measuring term or, both. Once more, research activity in this direction is dense, and our purpose is to simply highlight possible alternatives and make the reader aware of the various possibilities that spring from the basic theory.

In a number of tasks, it is a priori known that the nonzero coefficients in the target signal/vector occur in groups and they are not randomly spread in all possible positions. A typical example is the echo path in internet telephony, where the nonzero coefficients of the impulse response tend to cluster together; see Figure 9.5. Other examples of "structured" sparsity can be traced in DNA microarrays, MIMO channel equalization, source localization in sensor networks, magnetoencephalography, or in neuroscience problems (e.g., [1, 9, 10, 60, 101]). As is always the case in machine learning, being able to incorporate a priori information into the optimization can only be of benefit for improving performance, because the estimation task is externally assisted in its effort to search for the target solution.

The *group LASSO* [8, 59, 97, 98, 117, 122] addresses the task where it is a priori known that the nonzero components occur in groups. The unknown vector $\boldsymbol{\theta}$ is divided into $L$ groups, that is,

$$\boldsymbol{\theta}^{\mathrm{T}} = [\boldsymbol{\theta}_1^{\mathrm{T}}, \ldots, \boldsymbol{\theta}_L^{\mathrm{T}}]^{\mathrm{T}},$$

each of them of a predetermined size, $s_i$, $i = 1, 2, \ldots, L$, with $\sum_{i=1}^{L} s_i = l$. The regression model can then be written as

$$\boldsymbol{y} = X\boldsymbol{\theta} + \boldsymbol{\eta} = \sum_{i=1}^{L} X_i \boldsymbol{\theta}_i + \boldsymbol{\eta},$$

where each $X_i$ is a submatrix of $X$ comprising the corresponding $s_i$ columns. The solution of the group LASSO is given by the following LS regularized task:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^l} \left( \left\| \boldsymbol{y} - \sum_{i=1}^{L} X_i \boldsymbol{\theta}_i \right\|_2^2 + \lambda \sum_{i=1}^{L} \sqrt{s_i} \, \|\boldsymbol{\theta}_i\|_2 \right), \tag{10.15}$$

where $\|\boldsymbol{\theta}_i\|_2$ is the Euclidean norm (not the squared one) of $\boldsymbol{\theta}_i$, that is,

$$\|\boldsymbol{\theta}_i\|_2 = \sqrt{\sum_{j=1}^{s_i} |\theta_{i,j}|^2}.$$

In other words, the individual components of $\boldsymbol{\theta}$, which contribute to the formation of the $\ell_1$ norm in the standard LASSO formulation, are now replaced by the square root of the energy of each individual block. In this setting, it is not the individual components but *blocks* of them that are forced to zero, when their contribution to the LS misfit measuring term is not significant. Sometimes, this type of regularization is coined as the $\ell_1/\ell_2$ regularization. An example of an $\ell_1/\ell_2$ ball for $\boldsymbol{\theta} \in \mathbb{R}^3$ can be seen in Figure 10.6b in comparison with the corresponding $\ell_1$ ball depicted in Figure 10.6a.

Beyond the conventional group LASSO, often referred to as *block sparsity*, research effort has been dedicated to the development of learning strategies incorporating more elaborate *structured sparse* models. There are two major reasons for such directions. First, in a number of applications the unknown set of parameters, $\boldsymbol{\theta}$, exhibit a structure that cannot be captured by the block sparse model. Second, even for cases where $\boldsymbol{\theta}$ is block sparse, standard grouped $\ell_1$ norms require information about the partitioning of $\boldsymbol{\theta}$. This can be rather restrictive in practice. The adoption of overlapping groups has been proposed as a possible solution. Assuming that every coefficient belongs to at least one group, such models lead to optimization tasks that, in many cases, are not hard to solve, for example, by resorting to proximal methods [6, 7]. Moreover, by using properly defined overlapping groups [71], the allowed sparsity patterns can be constrained to form *hierarchical* structures, such as connected and rooted trees and subtrees that are met for example, in, multiscale (wavelet) decompositions. In Figure 10.6c, an example of an $\ell_1/\ell_2$ ball for overlapping groups is shown.

Besides the previously stated directions, extensions of the compressed sensing principles to cope with structured sparsity led to the *model-based* compressed sensing [10, 26]. The $(k, C)$ model allows the significant coefficients of a $k$-sparse signal to appear in at most $C$ clusters, whose size is unknown. In Section 9.9, it was commented that searching for a $k$-sparse solution takes place in a union of subspaces, each one of dimensionality $k$. Imposing a certain structure on the target solution restricts the searching in a subset of these subspaces and leaves a number of these out of the game. This obviously facilitates the optimization task. In [27], structured sparsity is considered in terms of graphical models, and in [110] the C-HiLasso group sparsity model was introduced, which allows each block to have a sparse structure itself. Theoretical results that extend the RIP to the block RIP have been developed and reported, see,

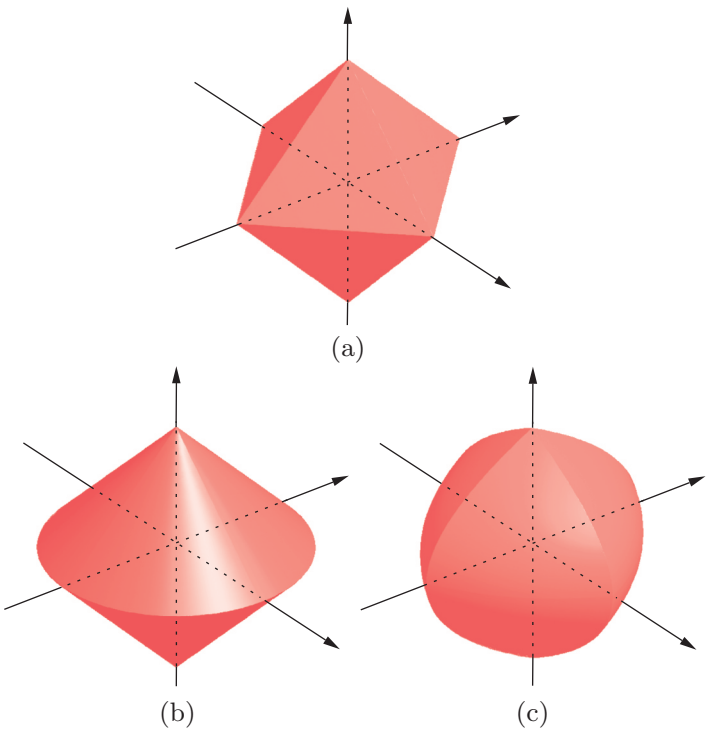


**FIGURE 10.6**

Representation of balls of $\boldsymbol{\theta} \in \mathbb{R}^3$ corresponding to: (a) the $\ell_1$ norm; (b) an $\ell_1/\ell_2$ with nonoverlapping groups; one group comprises $\{\theta_1, \theta_2\}$, and the other one $\{\theta_3\}$; (c) the $\ell_1/\ell_2$ with overlapping groups comprising $\{\theta_1, \theta_2, \theta_3\}$, $\{\theta_1\}$, and $\{\theta_3\}$, [6].

for example, [18, 83] and in the algorithmic front, proper modifications of greedy algorithms have been proposed in order to provide structured sparse solutions [53].

In [24], it is suggested to replace the $\ell_1$ norm by a weighted version of it. To justify such a choice, let us recall Example 9.2 and the case where the "unknown" system was sensed using $\boldsymbol{x} = [2, 1]^{\mathrm{T}}$. We have seen that by "blowing" up the $\ell_1$ ball, the wrong sparse solution was obtained. Let us now replace the $\ell_1$ norm in (9.21) with its weighted version

$$\|\boldsymbol{\theta}\|_{1,w} := w_1|\theta_1| + w_2|\theta_2|, \quad w_1, w_2 > 0,$$

and set $w_1 = 4$ and $w_1 = 1$. Figure 10.7a shows the isovalue curve $\|\boldsymbol{\theta}\|_{1,w} = 1$, together with that resulting from the standard $\ell_1$ norm. The weighted one is sharply "pinched" around the vertical axis, and the larger the value of $w_1$, compared to that of $w_2$, the sharper the corresponding ball will be. Figure 10.7b shows what happens when "blowing" the weighted $\ell_1$ ball. It will first touch the point $(0, 1)$, which is the true solution. Basically, what we have done is "squeeze" the $\ell_1$ ball to be aligned more to the axis that contains the (sparse) solution. For the case of our example, any weight $w_1 > 2$ would do the job.

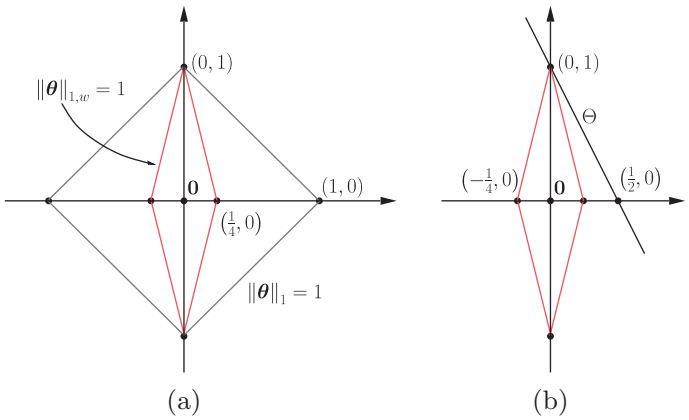


**FIGURE 10.7**

(a) The isovalue curves for the $\ell_1$ and the weighted $\ell_1$ norms for the same value. The weighted $\ell_1$ is sharply pinched around one of the axes, depending on the weights. (b) Adopting to minimize the weighted $\ell_1$ norm for the setup of Figure 9.9c, the correct sparse solution is obtained.

Consider now the general case of a weighted norm

$$\|\boldsymbol{\theta}\|_{1,w} := \sum_{j=1}^{l} w_j|\theta_j|, \quad w_j > 0, : \quad \text{Weighted } \ell_1 \text{ Norm.} \tag{10.16}$$

The ideal choice of the weights would be

$$w_j = \begin{cases} \frac{1}{|\theta_{o,j}|}, & \theta_{o,j} \neq 0, \\ \infty, & \theta_{o,j} = 0, \end{cases}$$

where $\boldsymbol{\theta}_o$ is the target true vector, and where we have silently assumed that $0 \cdot \infty = 0$. In other words, the smaller a coefficient, the larger the respective weight becomes. This is justified, because large weighting will force respective coefficients toward zero during the minimization process. Of course, in practice the values of the true vector are not known, so it is suggested to use their estimates during each iteration of the minimization procedure. The resulting scheme is of the following form.

**Algorithm 10.3.**

1. Initialize weights to unity, $w_j^{(0)} = 1, j = 1, 2, \ldots, l$.
2. Minimize the weighted $\ell_1$ norm,

$$\boldsymbol{\theta}^{(i)} = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^l} \|\boldsymbol{\theta}\|_{1,w}$$
$$\text{s.t.} \quad \boldsymbol{y} = X\boldsymbol{\theta}.$$

3. Update the weights

$$w_j^{(i+1)} = \frac{1}{\left|\theta_j^{(i)}\right| + \epsilon}, \quad j = 1, 2, \ldots, l.$$

4. Terminate when a stopping criterion is met, otherwise return to step 2.

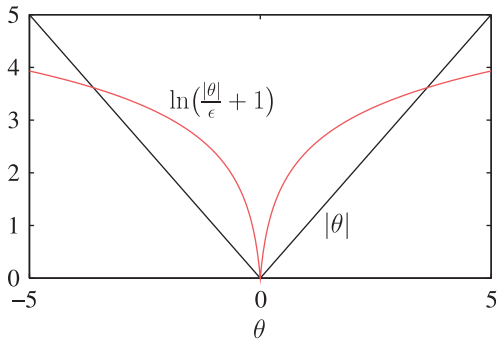


**FIGURE 10.8**

One-dimensional graphs of the $\ell_1$ norm and the logarithmic regularizer $\ln\left(\frac{|\theta|}{\epsilon}+1\right) = \ln(|\theta|+\epsilon) - \ln\epsilon$, with $\epsilon = 0.1$. The term $\ln\epsilon$ was subtracted for illustration purposes only and does not affect the optimization. Notice the nonconvex nature of the logarithmic regularizer.

The constant $\epsilon$ is a small user-defined parameter to guarantee stability when the estimates of the coefficients take very small values. Note that if the weights have constant preselected values, the task retains its convex nature; this is no longer true when the weights are changing. It is interesting to point out that this intuitively motivated weighting scheme can result if the $\ell_1$ norm is replaced by $\sum_{j=1}^{l} \ln\left(|\theta_j| + \epsilon\right)$ as the regularizing term of (9.6). Figure 10.8 shows the respective graph, in the one-dimensional space together with that of the $\ell_1$ norm. The graph of the logarithmic function reminds us of the $\ell_p$, $p < 0 < 1$ "norms" and the comments made in Section 9.2. This is no longer a convex function, and the iterative scheme given before is the result of a majorization-minimization procedure in order to solve the resulting nonconvex task [24] (Problem 10.6).

The concept of iterative weighting, as used before, has also been applied in the context of the *iterative reweighted LS algorithm*. Observe that the $\ell_1$ norm can be written as

$$\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{l} |\theta_j| = \boldsymbol{\theta}^{\mathrm{T}} \mathcal{W}_\theta \boldsymbol{\theta},$$

where

$$\mathcal{W}_\theta = \begin{bmatrix} \frac{1}{|\theta_1|} & 0 & \cdots & 0 \\ 0 & \frac{1}{|\theta_2|} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{|\theta_l|} \end{bmatrix},$$

and where in the case of $\theta_i = 0$, for some $i \in \{1, 2, \ldots, l\}$, the respective coefficient of $\mathcal{W}_\theta$ is defined to be 1. If $\mathcal{W}_\theta$ were a constant weighting matrix, that is, $\mathcal{W}_\theta := \mathcal{W}_{\tilde{\theta}}$, for some fixed $\tilde{\boldsymbol{\theta}}$, then obtaining the minimum

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^l} \|\boldsymbol{y} - X\boldsymbol{\theta}\|_2^2 + \lambda \boldsymbol{\theta}^{\mathrm{T}} \mathcal{W}_{\tilde{\theta}} \boldsymbol{\theta},$$

is straightforward and similar to the ridge regression. In the iterative reweighted scheme, $\mathcal{W}_\theta$ is replaced by $\mathcal{W}_{\theta^{(i)}}$, formed by using the respected estimates of the coefficients, which have been obtained from the previous iteration, that is, $\tilde{\boldsymbol{\theta}} := \boldsymbol{\theta}^{(i)}$, as we did before. In the sequel, each iteration solves a weighted ridge regression task.

The *focal underdetermined system solver* (FOCUSS) algorithm, [64], was the first one to use the concept of iterative-reweighted-least-squares (IRLS) to represent $\ell_p$, $p \leq 1$ as a weighted $\ell_2$ norm in order to find a sparse solution to an underdetermined system of equations. This algorithm is also of historical importance, because it is among the very first ones to emphasize the importance of sparsity; moreover, it provides comprehensive convergence analysis as well as a characterization of the stationary points of the algorithm. Variants of this basic iterative weighting scheme have also been proposed (see, e.g., [35] and the references therein).

In [126], the *elastic net* regularization penalty was introduced, which combines the $\ell_2$ and $\ell_1$ concepts together in a trade-off fashion, that is,

$$\lambda \sum_{i=1}^{l} \left( \alpha \theta_i^2 + (1-\alpha)|\theta_i| \right),$$

where $\alpha$ is a user-defined parameter controlling the influence of each individual term. The idea behind the elastic net is to combine the advantages of the LASSO and the ridge regression. In problems where there is a group of variables in $\boldsymbol{x}$ that are highly correlated, LASSO tends to select one of the corresponding coefficients in $\boldsymbol{\theta}$, and set the rest to zero in a rather arbitrary fashion. This can be understood by looking carefully at how the greedy algorithms work. When sparsity is used to select the most important of the variables in $\boldsymbol{x}$ (feature selection), it is better to select all the relevant components in the group. If one knew which of the variables are correlated, he/she could form a group and then use the group LASSO. However, if this is not known, involving the ridge regression offers a remedy to the problem. This is because the $\ell_2$ penalty in ridge regression tends to shrink the coefficients associated with correlated variables toward each other (e.g., [68]). In such cases, it would be better to work with the elastic net rationale that involves LASSO and ridge regression in a combined fashion.

In [23], the LASSO task is modified by replacing the squared error term with one involving correlations, and the minimization task becomes

$$\begin{aligned} \hat{\boldsymbol{\theta}}: \quad & \min_{\boldsymbol{\theta} \in \mathbb{R}^l} \|\boldsymbol{\theta}\|_1 \\ \text{s.t.} \quad & \left\| X^{\mathrm{T}} (\boldsymbol{y} - X\boldsymbol{\theta}) \right\|_\infty \leq \epsilon, \end{aligned}$$

where $\epsilon$ is related to $l$ and the noise variance. This task is known as the *Dantzig selector*. That is, instead of constraining the energy of the error, the constraint now imposes an upper limit to the correlation of the error vector with any of the columns of $X$. In [5, 15], it is shown that under certain conditions, the LASSO estimator and the Dantzig selector become identical.

*Total variation (TV)* [107] is a closely related to $\ell_1$ sparsity-promoting notion that has been widely used in image processing. Most of the grayscale image arrays, $I \in \mathbb{R}^{l \times l}$, consist of slowly varying pixel intensities except at the edges. As a consequence, the discrete gradient of an image array will be approximately sparse (compressible). The discrete directional derivatives of an image array are defined pixel-wise as

$$\nabla_x(I)(i,j) := I(i+1,j) - I(i,j), \quad \forall i \in \{1,2,\ldots,l-1\}, \tag{10.17}$$

$$\nabla_y(I)(i,j) := I(i,j+1) - I(i,j), \quad \forall j \in \{1,2,\ldots,l-1\}, \tag{10.18}$$

and

$$\nabla_x(I)(l,j) := \nabla_y(I)(i,l) := 0, \quad \forall i,j \in \{1,2,\ldots,l-1\}. \tag{10.19}$$

The discrete gradient transform

$$\nabla : \mathbb{R}^{l\times l} \to \mathbb{R}^{l\times 2l},$$

is defined in terms of a matrix form as

$$\nabla(I)(i,j) := [\nabla_x(i,j), \nabla_y(i,j)], \quad \forall i,j \in \{1,2,\ldots l\}. \tag{10.20}$$

The total variation of the image array is defined as the $\ell_1$ norm of the *magnitudes* of the elements of the discrete gradient transform, that is,

$$\|I\|_{\text{TV}} := \sum_{i=1}^{l}\sum_{j=1}^{l} \|\nabla(I)(i,j)\|_2 = \sum_{i=1}^{l}\sum_{j=1}^{l} \sqrt{\nabla_x(I)^2(i,j) + \nabla_y(I)^2(i,j)}. \tag{10.21}$$

Note that this is a mixture of $\ell_2$ and $\ell_1$ norms. The sparsity promoting optimization around the total variation is defined as

$$\begin{aligned} & I_* \in \arg\min_{I} \quad \|I\|_{TV} \\ & \text{s.t.} \qquad \|\boldsymbol{y} - \mathcal{F}(I)\|_2 \le \epsilon, \end{aligned} \tag{10.22}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ is the observations vector and $\mathcal{F}(I)$ denotes the result in vectorized form of the application of a linear operator on $I$. For example, this could be the result of the action of a partial two-dimensional DFT on the image. Subsampling of the DFT matrix as a means of forming sensing matrices has already been discussed in Section 9.7.2. The task in (10.22) retains its convex nature and it basically expresses our desire to reconstruct an image, that is as smooth as possible, given the available observations. The NESTA algorithm can be used for solving the total variation minimization task; besides it, other efficient algorithms for this task can be found in, for example, [63, 120].

It has been shown in [22] for the exact measurements case ($\epsilon = 0$), and in [94] for the erroneous measurements case, that conditions and bounds that guarantee recovery of an image array from the task in (10.22) can be derived and are very similar to those we have discussed for the case of the $\ell_1$ norm.

**Example 10.1** (*Magnetic resonance imaging (MRI)*)**.** In contrast to ordinary imaging systems, which directly acquire pixel samples, MRI scanners sense the image in an encoded form. Specifically, MRI scanners sample components in the spatial frequency domain, known as "$k$-space" in MRI nomenclature. If all the components in this transform domain were available, one could apply the inverse 2D-DFT to recover the exact MR image in the pixel domain. Sampling in the $k$-space is realized along particular trajectories in a number of successive acquisitions. This process is time consuming, merely due to physical constraints. As a result, techniques for efficient image recovery from a *limited number of observations* is of high importance, because they can reduce the required acquisition time for performing

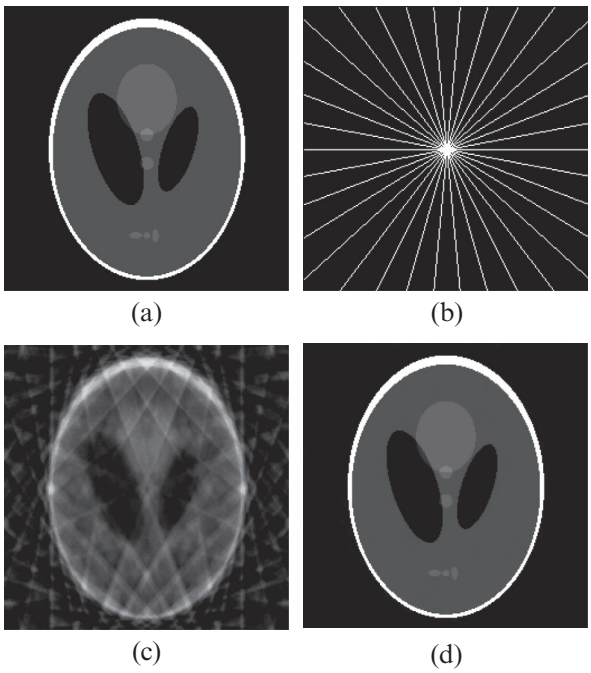


**FIGURE 10.9**

(a) The original Shepp-Logan image phantom. (b) The white lines indicate the directions across which the sampling in the spatial Fourier transform were obtained. (c) The recovered image after applying the inverse DFT, having first filled with zeros the missing values in the DFT transform. (d) The recovered image using the total variation minimization approach.

the measurements. Long acquisition times are not only inconvenient but even impossible, because the patients have to stay still for long time intervals. Thus, MRI was among the very first applications where compressed sensing found its way to offering its elegant solutions.

Figure 10.9a shows the "famous" Shepp-Logan phantom, and the goal is to recover it via a limited number of (measurements) samples in its frequency domain. The MRI measurements are taken across 17 radial lines in the spatial frequency domain, as shown in Figure 10.9b. A "naive" approach to recovering the image from this limited number of measuring samples would be to adopt a zero-filling rationale for the missing components. The recovered image according to this technique is shown in Figure 10.9c. Figure 10.9d shows the recovered image using the approach of minimizing the total variation, as explained before. Observe that the results for this case are astonishingly good. The original image is almost perfectly recovered. The constrained minimization was performed via the NESTA algorithm. Note that if the minimization of the $\ell_1$ norm of the image array were used in place of the total variation, the results would not be as good; the phantom image is sparse in the discrete gradient domain, because it contains large sections that share constant intensities.

## 10.4 ONLINE SPARSITY-PROMOTING ALGORITHMS

In this section, online schemes for sparsity-aware learning are presented. There are a number of reasons that one has to resort to such schemes. As has already been noted in previous chapters, in various signal processing tasks the data arrive sequentially. Under such a scenario, using batch processing techniques to obtain an estimate of an unknown target parameter vector would be highly inefficient, because the number of training points keeps increasing. Such an approach is prohibited for real-time applications. Moreover, time-recursive schemes can easily incorporate the notion of adaptivity, when the learning environment is not stationary but undergoes changes as time evolves. Besides signal processing applications, there are an increasing number of machine learning applications where online processing is of paramount importance, such as bioinformatics, hyperspectral imaging, and data mining. In such applications, the number of training points easily amounts to a few thousand up to hundreds of thousand points. Concerning the dimensionality of the ambient (feature) space, one can claim numbers that lie in similar ranges. For example, in [82], the task is to search for sparse solutions in feature spaces with dimensionality as high as $10^9$ having access to data sets as large as $10^7$ points. Using batch techniques, in a single computer is out of the question with today's technology.

The setting that we have adopted for this section is the same as that used in previous chapters (e.g., Chapters 5 and 6). We assume that there is an unknown parameter vector that generates data according to the standard regression model

$$y_n = \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{\theta} + \eta_n, \quad \forall n,$$

and the training samples are received sequentially $(y_n, \boldsymbol{x}_n)$, $n = 1, 2, \ldots$. In the case of a stationary environment, we would expect our algorithm to converge asymptotically, as $n \longrightarrow \infty$, to or "near to" the true parameter vector that gives birth to the observations, $y_n$, when it is sensed by $\boldsymbol{x}_n$. For time varying environments, the algorithms should be able to track the underlying changes as time goes by. Before we proceed, a comment is important. Because the time index, $n$, is left to grow, all we have said in the previous sections with respect to underdetermined systems of equations, loses its meaning. Sooner or later we are going to have more observations than the dimension of the space in which the data live. Our major concern here becomes the issue of asymptotic convergence for the case of stationary environments. The obvious question that is now raised is why not use a standard algorithm (e.g., LMS, RLS, or APSM), because we know that these algorithms converge to, or near enough in some sense, the solution (i.e., the algorithm will identify the zeros asymptotically)? The answer is that if such algorithms are modified to be aware of the underlying sparsity, convergence is significantly speeded up; in real-life applications, one does not have the "luxury" of waiting a long time for the solution. In practice, a good algorithm should be able to provide a good enough solution, and in the case of sparse solutions to *obtain the support*, after a reasonably small number of iteration steps.

In Chapter 5, we commented on attempts to modify classical online schemes (for example, the proportionate LMS) in order to consider sparsity. However, these algorithms were of a rather ad hoc nature. In this section, the powerful theory around the $\ell_1$ norm regularization will be used to obtain sparsity-promoting time adaptive schemes.

### 10.4.1 LASSO: ASYMPTOTIC PERFORMANCE

When presenting the basic principles of parameter estimation in Chapter 3, the notions of bias, variance, and consistency, which are related to the performance of an estimator, were introduced. In a number

of cases, such performance measures were derived asymptotically. For example, we have seen that the maximum likelihood estimator is asymptotically unbiased and consistent. In Chapter 6, we saw that the LS estimator is also asymptotically consistent. Moreover, under the assumption that the noise samples are i.i.d., the LS estimator, $\hat{\boldsymbol{\theta}}_n$, based on $n$ observations, is itself a random vector that satisfies the $\sqrt{n}$-estimation consistency, that is,

$$\sqrt{n}\left(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_o\right) \xrightarrow{d} \mathcal{N}\left(\mathbf{0}, \sigma^2 \Sigma^{-1}\right),$$

where $\boldsymbol{\theta}_o$ is the true vector that generates the observations, $\Sigma$ denotes the variance of the noise source, and $\Sigma$ is the covariance matrix $\mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$ of the input sequence, which has been assumed to be zero mean and the limit denotes convergence in distribution.

The LASSO in its (9.6) formulation is the task of minimizing the $\ell_1$ norm regularized version of the LS cost. However, nothing has been said so far about the statistical properties of this estimator. The only performance measure that we referred to was the error norm bound given in (9.36). However, this bound, although important in the context for which it was proposed, does not provide much statistical information. Since the introduction of the LASSO estimator, a number of papers have addressed problems related to its statistical performance (see, e.g., [45, 55, 74, 127]).

When dealing with sparsity-promoting estimators such as the LASSO, two crucial issues emerge: (a) whether the estimator, even asymptotically, can obtain the support, if the true vector parameter is a sparse one; and (b) to quantify the performance of the estimator with respect to the estimates of the nonzero coefficients, that is, those coefficients whose index belongs to the support. Especially for LASSO, the latter issue becomes to study whether LASSO behaves as well as the unregularized LS with respect to these nonzero components. This task was addressed for the first time and in a more general setting in [55]. Let the support of the true, yet unknown, $k$-sparse parameter vector $\boldsymbol{\theta}_o$ be denoted as $S$. Let also $\Sigma_{|S}$ be the $k \times k$ covariance matrix $\mathbb{E}[\mathbf{x}_{|S}\mathbf{x}_{|S}^{\mathrm{T}}]$, where $\mathbf{x}_{|S} \in \mathbb{R}^k$ is the random vector that contains only the $k$ components of $\mathbf{x}$, with indices in the support $S$. Then, we say that an estimator satisfies asymptotically the *oracle properties* if:

- $\lim_{n\to\infty} \text{Prob}\left\{S_{\hat{\boldsymbol{\theta}}_n} = S\right\} = 1$. This is known as *support consistency*.
- $\sqrt{n}\left(\hat{\boldsymbol{\theta}}_{n|S} - \boldsymbol{\theta}_{o|S}\right) \xrightarrow{d} \mathcal{N}\left(\mathbf{0}, \sigma^2 \Sigma_{|S}^{-1}\right)$. This is the *$\sqrt{n}$-estimation consistency*.

We denote as $\boldsymbol{\theta}_{o|S}$ and $\hat{\boldsymbol{\theta}}_{n|S}$ the $k$-dimensional vectors that result from $\boldsymbol{\theta}_o$, $\hat{\boldsymbol{\theta}}_n$, respectively, if we keep the components whose indices lie in the support $S$. In other words, according to the oracle properties, a good sparsity-promoting estimator should be able to predict, asymptotically, the true support and its performance with respect to the nonzero components should be as good as that of a genie-aided LS estimator, which is informed in advance of the positions of the nonzero coefficients.

Unfortunately, the LASSO estimator *cannot* satisfy simultaneously both conditions. It has been shown [55, 74, 127] that

- For support consistency, the regularization parameter $\lambda := \lambda_n$ should be time varying such that

$$\lim_{n\to\infty} \frac{\lambda_n}{\sqrt{n}} = \infty, \quad \lim_{n\to\infty} \frac{\lambda_n}{n} = 0.$$

  That is, $\lambda_n$ must grow faster than $\sqrt{n}$, but slower than $n$.

- For $\sqrt{n}$-consistency, $\lambda_n$ must grow as

$$\lim_{n\to\infty} \frac{\lambda_n}{\sqrt{n}} = 0,$$

that is, it grows slower than $\sqrt{n}$.

The previous two conditions are conflicting and the LASSO estimator cannot comply with the two oracle conditions simultaneously. The proofs of the previous two points are somewhat technical and are not given here. The interested reader can obtain them from the previously given references. However, before we proceed, it is instructive to see why the regularization parameter has to grow more slowly than $n$, in any case. Without being too rigorous mathematically, recall that the LASSO solution comes from Eq. (9.6). This can be written as

$$\mathbf{0} \in -\frac{2}{n}\sum_{i=1}^{n} \boldsymbol{x}_i y_i + \frac{2}{n}\left(\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}}\right)\boldsymbol{\theta} + \frac{\lambda_n}{n}\partial \|\boldsymbol{\theta}\|_1 , \tag{10.23}$$

where we have divided both sides by $n$. Taking the limit as $n\longrightarrow\infty$, if $\lambda_n/n\longrightarrow 0$, then we are left with the first two terms; this is exactly what we would have if the unregularized LS had been chosen as the cost function. Recall from Chapter 6 that in this case, the solution asymptotically converges[1] (under some general assumptions, which are assumed to hold true here) to the true parameter vector; that is, we have strong consistency.

### 10.4.2 THE ADAPTIVE NORM-WEIGHTED LASSO

There are two ways to get out of the previously stated conflict. One is to replace the $\ell_1$ norm with a nonconvex function, which can lead to an estimator that satisfies the oracle properties simultaneously [55]. The other is to modify the $\ell_1$ norm by replacing it with a *weighted* version. Recall that the weighted $\ell_1$ norm was discussed in Section 10.3 as a means to assist the optimization procedure to unveil the sparse solution. Here the notion of weighted $\ell_1$ norm comes as a necessity imposed by our willingness to satisfy the oracle properties. This gives rise to the *adaptive time-and-norm-weighted LASSO* (TNWL) cost estimate defined as

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^l} \left\{ \sum_{j=1}^{n} \beta^{n-j}\left(y_j - \boldsymbol{x}_j^{\mathrm{T}}\boldsymbol{\theta}\right)^2 + \lambda_n \sum_{i=1}^{l} w_{i,n}|\theta_i| \right\}, \tag{10.24}$$

where $\beta \leq 1$ is used as the forgetting factor to allow for tracking slow variations. The time varying weighting sequences is denoted as $w_{i,n}$. There are different options. In [127] and under a stationary environment with $\beta = 1$, it is shown that if

$$w_{i,n} = \frac{1}{|\theta_i^{\mathrm{est}}|^{\gamma}},$$

where $\theta_i^{\mathrm{est}}$ is the estimate of the $i$th component obtained by *any* $\sqrt{n}$-consistent estimator, such as the unregularized LS, then for specific choices of $\lambda_n$ and $\gamma$ the corresponding estimator satisfies the oracle

[1] Recall that this convergence is with probability 1.

properties simultaneously. The main reasoning behind the weighted $\ell_1$ norm is that as time goes by, and the $\sqrt{n}$-consistent estimator provides better and better estimates, then the weights corresponding to indices outside the true support (zero values) are inflated and those corresponding to the true support converge to a finite value. This helps the algorithm, simultaneously to locate the support and obtain unbiased (asymptotically) estimates of the large coefficients.

Another choice for the weighting sequence is related to the *smoothly clipped absolute deviation (SCAD)* [55, 128]. This is defined as

$$w_{i,n} = \chi_{(0,\mu_n)}(|\theta_i^{\text{est}}|) + \frac{\left(\alpha\mu_n - |\theta_i^{\text{est}}|\right)_+}{(\alpha-1)\mu_n}\chi_{(\mu_n,\infty)}(|\theta_i^{\text{est}}|),$$

where $\chi(\cdot)$ stands for the characteristic function, $\mu_n = \lambda_n/n$, and $\alpha > 2$. Basically, this corresponds to a quadratic spline function. It turns out [128] that if $\lambda_n$ is chosen to grow faster than $\sqrt{n}$ and slower than $n$, the adaptive LASSO with $\beta = 1$ satisfies both oracle conditions simultaneously.

A time adaptive scheme for solving the TNWL LASSO was presented in [3]. The cost function of the adaptive LASSO in (10.24) can be written as

$$J(\boldsymbol{\theta}) = \boldsymbol{\theta}^T R_n \boldsymbol{\theta} - \boldsymbol{r}_n^T \boldsymbol{\theta} + \lambda_n \|\boldsymbol{\theta}\|_{1,w_n},$$

where

$$R_n := \sum_{j=1}^{n} \beta^{n-j} \boldsymbol{x}_j \boldsymbol{x}_j^{\mathrm{T}}, \quad \boldsymbol{r}_n := \sum_{j=1}^{n} \beta^{n-j} y_j \boldsymbol{x}_j,$$

and $\|\boldsymbol{\theta}\|_{1,\boldsymbol{w}_n}$ is the weighted $\ell_1$ norm. We know from Chapter 6, and it is straightforward to see, that

$$R_n = \beta R_{n-1} + \boldsymbol{x}_n \boldsymbol{x}_n^{\mathrm{T}}, \quad \boldsymbol{r}_n = \beta \boldsymbol{r}_{n-1} + y_n \boldsymbol{x}_n.$$

The complexity for both of the previous updates, for matrices of a general structure, amounts to $\mathcal{O}(l^2)$ multiply/add operations. One alternative is to update $R_n$ and $\boldsymbol{r}_n$ and then solve a convex optimization task for each time instant, $n$, using any standard algorithm. However, this is not appropriate for real-time applications, due to its excessive computational cost. In [3], a time-recursive version of a coordinate descent algorithm has been developed. As we saw in Section 10.2.2, coordinate descent algorithms update one component at each iteration step. In [3], iteration steps are associated with time updates, as is always the case with the online algorithms. As each new training pair $(y_n, \boldsymbol{x}_n)$ is received, a single component of the unknown vector is updated. Hence, at each time instant, a scalar optimization task has to be solved, and its solution is given in closed form, which results in a simple soft thresholding operation. One of the drawbacks of the coordinate techniques is that each coefficient is updated every $l$ time instants, which, for large values of $l$, can slow down convergence. Variants of the basic scheme that cope with this drawback are also addressed in [3], referred to as online cyclic coordinate descent time weighted LASSO (OCCD-TWL). The complexity of the scheme is of the order of $\mathcal{O}(l^2)$. Computational savings are possible if the input sequence is a time series and fast schemes for the updates of $R_n$ and the RLS can then be exploited. However, if an RLS-type algorithm is used in parallel, the convergence of the overall scheme may be slowed down, because the RLS-type algorithm has to converge first in order to provide reliable estimates for the weights, as pointed out before.

### 10.4.3 ADAPTIVE CoSaMP (AdCoSaMP) ALGORITHM

In [90], an adaptive version of the CoSaMP algorithm, which is summarized in Algorithm 10.2, was proposed. Iteration steps, $i$, now coincide with time updates, $n$, and the LS solver in Step 3c of the general CSMP scheme is replaced by an LMS one.

Let us focus first on the quantity $X^{\mathrm{T}}\boldsymbol{e}^{(i-1)}$ in Step 3a of the CSMP scheme, which is used to compute the support at iteration $i$. In the online setting and at (iteration) time $n$, this quantity is now "rephrased" as

$$X^{\mathrm{T}}\boldsymbol{e}_{n-1} = \sum_{j=1}^{n-1} \boldsymbol{x}_j e_j.$$

In order to make the algorithm flexible to adapt to variations of the environment as the time index, $n$, increases, the previous correlation sum is modified to

$$\boldsymbol{p}_n := \sum_{j=1}^{n-1} \beta^{n-1-j}\boldsymbol{x}_j e_j = \beta \boldsymbol{p}_{n-1} + \boldsymbol{x}_{n-1}e_{n-1}.$$

The LS task, constrained on the active columns that correspond to the indices in the support $S$ in Step 3c, is performed in an online rationale by involving the basic LMS recursions, that is,[2]

$$\tilde{e}_n := y_n - \boldsymbol{x}_{n|S}^{\mathrm{T}}\tilde{\boldsymbol{\theta}}_{|S}(n-1)$$

$$\tilde{\boldsymbol{\theta}}_{|S}(n) := \tilde{\boldsymbol{\theta}}_{|S}(n-1) + \mu \boldsymbol{x}_{n|S}\tilde{e}_n,$$

where $\tilde{\boldsymbol{\theta}}_{|S}(\cdot)$ and $\boldsymbol{x}_{n|S}$ denote the respective subvectors corresponding to the indices in the support $S$. The resulting algorithm is summarized as follows.

**Algorithm 10.4 (The AdCoSaMP scheme).**

1. Select the value of $t = 2k$.
2. Initialize the algorithm: $\boldsymbol{\theta}(1) = \mathbf{0}$, $\tilde{\boldsymbol{\theta}}(1) = \mathbf{0}$, $\boldsymbol{p}_1 = \mathbf{0}$, $e_1 = y_1$.
3. Choose $\mu$ and $\beta$.
4. For $n = 2, 3, \ldots$, execute the following steps:
   - **(a)** $\boldsymbol{p}_n = \beta \boldsymbol{p}_{n-1} + \boldsymbol{x}_{n-1}e_{n-1}$.
   - **(b)** Obtain the current support:

$$S = \mathrm{supp}\{\boldsymbol{\theta}(n-1)\} \cup \left\{ \begin{array}{c} \text{indices of the } t \text{ largest in} \\ \text{magnitude components of } \boldsymbol{p}_n \end{array} \right\}.$$

   - **(c)** Perform the LMS update:

$$\tilde{e}_n = y_n - \boldsymbol{x}_{n|S}^{\mathrm{T}}\tilde{\boldsymbol{\theta}}_{|S}(n-1),$$
$$\tilde{\boldsymbol{\theta}}_{|S}(n) = \tilde{\boldsymbol{\theta}}_{|S}(n-1) + \mu \boldsymbol{x}_{n|S}\tilde{e}_n.$$

   - **(d)** Obtain the set $S_k$ of the indices of the $k$ largest components of $\tilde{\boldsymbol{\theta}}_{|S}(n)$.
   - **(e)** Obtain $\boldsymbol{\theta}(n)$ such that:

$$\boldsymbol{\theta}_{|S_k}(n) = \tilde{\boldsymbol{\theta}}_{|S_k}, \quad \text{and } \boldsymbol{\theta}_{|S_k^c}(n) = \mathbf{0},$$

     where $S_k^c$ is the complement set of $S_k$.
   - **(f)** Update the error: $e_n = y_n - \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{\theta}(n)$.

[2] The time index for the parameter vector is given in parentheses, due to the presence of the other subscripts.

In place of the standard LMS, its normalized version can alternatively be adopted. Note that Step 4e is directly related to the hard thresholding operation.

In [90], it is shown that if the sensing matrix, which is now time dependent and keeps increasing in size, satisfies a condition similar to RIP for each time instant, called *exponentially weighted isometry property* (ERIP), which depends on $\beta$, then the algorithm asymptotically satisfies an error bound, which is similar to the one that has been derived for CoSaMP in [93], plus an extra term that is due to the excess mean-square error (see Chapter 5), which is the price paid by replacing the LS solver by the LMS.

### 10.4.4 SPARSE ADAPTIVE PROJECTION SUBGRADIENT METHOD (SpAPSM)

The APSM family of algorithms was introduced in Chapter 8, as one among the most popular techniques for online/adaptive learning. As pointed out there, a major advantage of this algorithmic family is that one can readily incorporate convex constraints. In Chapter 8, APSM was used as an alternative to methods that build around the LS loss function, such as the LMS and the RLS. The rationale behind APSM is that because our data are assumed to be generated by a regression model, then the unknown vector could be estimated by finding a point in the intersection of a sequence of hyperslabs that are defined by the data points, that is, $S_n[\epsilon] := \left\{\boldsymbol{\theta} \in \mathbb{R}^l : \left|y_n - \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{\theta}\right| \leq \epsilon\right\}$. Also, it was pointed out that such a model was very natural when the noise is bounded. When dealing with sparse vectors, there is an additional constraint that we want our solution to satisfy, that is, $\|\boldsymbol{\theta}\|_1 \leq \rho$ (see also the LASSO formulation (9.7)). This task fits nicely in the APSM rationale and the basic recursion can be readily written, without much thought or derivation, as follows: for any arbitrarily chosen initial point $\boldsymbol{\theta}_0$, define $\forall n$,

$$\boxed{\boldsymbol{\theta}_n = P_{B_{\ell_1}[\delta]}\left(\boldsymbol{\theta}_{n-1} + \mu_n\left(\sum_{i=n-q+1}^{n}\omega_i^{(n)}P_{S_i[\epsilon]}\left(\boldsymbol{\theta}_{n-1}\right) - \boldsymbol{\theta}_{n-1}\right)\right)}, \tag{10.25}$$

where $q \geq 1$ is the number of hyperslabs that are considered each time, $\mu_n$ is an extrapolation parameter, and it is a user-defined variable. In order for convergence to be guaranteed, theory dictates that it must lie in the interval $(0, 2\mathcal{M}_n)$, where

$$\mathcal{M}_n := \begin{cases} \dfrac{\sum_{i=n-q+1}^{n}\omega_i^{(n)}\left\|P_{S_i[\epsilon]}(\boldsymbol{\theta}_{n-1}) - \boldsymbol{\theta}_{n-1}\right\|^2}{\left\|\sum_{i=n-q+1}^{n}\omega_i^{(n)}P_{S_i[\epsilon]}(\boldsymbol{\theta}_{n-1}) - \boldsymbol{\theta}_{n-1}\right\|^2}, \\ \quad \text{if } \left\|\sum_{i=n-q+1}^{n}\omega_i^{(n)}P_{S_i[\epsilon]}(\boldsymbol{\theta}_{n-1}) - \boldsymbol{\theta}_{n-1}\right\| \neq 0, \\ 1, \text{otherwise}, \end{cases} \tag{10.26}$$

and $P_{B_{\ell_1}[\rho]}(\cdot)$ is the projection operator onto the $\ell_1$ ball $B_{\ell_1}[\rho] := \left\{\boldsymbol{\theta} \in \mathbb{R}^l : \|\boldsymbol{\theta}\|_1 \leq \rho\right\}$, because the solution is constrained to live within this ball. Note that recursion (10.25) is analogous to the iterative soft thresholding shrinkage algorithm in the batch processing case (10.7). There, we saw that the only difference the sparsity imposes on an iteration, with respect to its unconstrained counterpart, is an extra soft thresholding. This is exactly the case here. The term in the parentheses is the iteration for the unconstrained task. Moreover, as has been shown in [46], projection on the $\ell_1$ ball is equivalent to a soft thresholding operation. Following the general arguments given in Chapter 8, the previous iteration converges arbitrarily close to a point in the intersection

$$B_{\ell_1}[\delta] \cap \bigcap_{n \geq n_0} S_n[\epsilon],$$

for some finite value of $n_0$. In [76, 77], the weighted $\ell_1$ ball (denoted here as $B_{\ell_1[\boldsymbol{w}_n,\rho]}$) has been used to improve convergence as well as the tracking speed of the algorithm, when the environment is time varying. The weights were adopted in accordance with what was discussed in Section 10.3, that is,

$$w_{i,n} := \frac{1}{|\theta_{i,n-1}| + \acute{\epsilon}_n}, \quad \forall i \in \{1, 2, \ldots, l\},$$

where $(\acute{\epsilon}_n)_{n \geq 0}$ is a sequence (can be also constant) of small numbers to avoid division by zero. The basic time iteration becomes as follows: for any arbitrarily chosen initial point $\boldsymbol{\theta}_0$, define $\forall n$,

$$\boxed{\boldsymbol{\theta}_n = P_{B_{\ell_1}[\boldsymbol{w}_n,\rho]}\left(\boldsymbol{\theta}_{n-1} + \mu_n\left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} P_{S_i[\epsilon]}(\boldsymbol{\theta}_{n-1}) - \boldsymbol{\theta}_{n-1}\right)\right)}, \tag{10.27}$$

where $\mu_n \in (0, 2\mathcal{M}_n)$ and $\mathcal{M}_n$ is given in (10.26). Figure 10.10 illustrates the associated geometry of the basic iteration in $\mathbb{R}^2$ and for the case of $q = 2$. It comprises two parallel projections on the

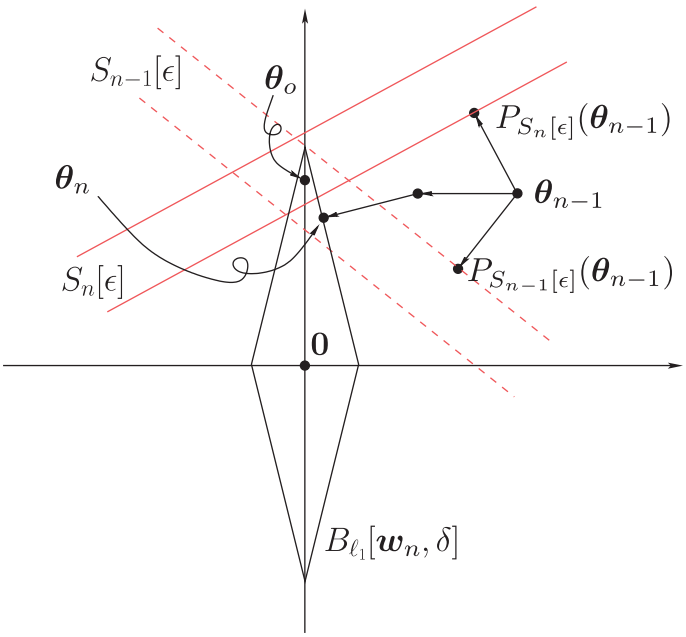


**FIGURE 10.10**

Geometric illustration of the update steps involved in the SpAPSM algorithm, for the case of $q = 2$. The update at time $n$ is obtained by first convexly combining the projections onto the current and previously formed hyperslabs, $S_n[\epsilon]$, $S_{n-1}[\epsilon]$, and then projecting onto the weighted $\ell_1$ ball. This brings the update closer to the target solution $\boldsymbol{\theta}_o$.

hyperslabs, followed by one projection onto the weighted $\ell_1$ ball. In [76], it is shown (Problem 10.7) that a good bound for the weighted $\ell_1$ norm is the sparsity level $k$ of the target vector, which is assumed to be known and is a user-defined parameter. In [76], it is shown that asymptotically, and under some general assumptions, this algorithmic scheme converges arbitrarily close to the intersection of the hyperslabs with the weighted $\ell_1$ balls, that is,

$$\bigcap_{n\geq n_0}\left(P_{B_{\ell_1}[\boldsymbol{w}_n,\rho]}\cap S_n[\epsilon]\right),$$

for some nonnegative integer $n_0$. It has to be pointed out that in the case of weighted $\ell_1$ norms, the constraint is *time varying* and the convergence analysis is not covered by the standard analysis used for APSM, and had to be extended to this more general case.

The complexity of the algorithm amounts to $\mathcal{O}(ql)$. The larger the $q$ the faster the convergence rate, at the expense of higher complexity. In [77], in order to reduce the dependence of the complexity on $q$, the notion of the *subdimensional* projection is introduced, where projections onto the $q$ hyperslabs could be restricted along the directions of the most significant coefficients of the currently available estimates. The dependence on $q$ now becomes $\mathcal{O}(qk_n)$, where $k_n$ is the sparsity level of the currently available estimate, which after a few steps of the algorithm gets much lower than $l$. The total complexity amounts to $\mathcal{O}(l)+\mathcal{O}(qk_n)$ per iteration step. This allows the use of large values of $q$, which (at only a small extra computational cost compared to $\mathcal{O}(l)$) drives the algorithm to a performance close to that of the adaptive weighted LASSO.

### *Projection onto the weighted $\ell_1$ ball*

Projecting onto an $\ell_1$ ball is equivalent to a soft thresholding operation. Projection onto the weighted $\ell_1$ norm results in a slight variation of the soft thresholding, with different threshold values per component. In the sequel, we give the iteration steps for the more general case of the weighted $\ell_1$ ball. The proof is a bit technical and lengthy and it will not be given here. It was derived, for the first time, via purely geometric arguments, and without the use of the classical Lagrange multipliers, in [76]. Lagrange multipliers have been used instead in [46], for the case of the $\ell_1$ ball. The efficient computation of the projection on the $\ell_1$ ball was treated earlier, in a more general context, in [100].

Recall from the definition of a projection, discussed in Chapter 8, that given a point outside the ball, $\boldsymbol{\theta}\in\mathbb{R}^l\setminus B_{\ell_1}[\boldsymbol{w},\rho]$, its projection onto the weighted $\ell_1$ ball is the point $P_{B_{\ell_1}[\boldsymbol{w},\rho]}(\boldsymbol{\theta})\in B_{\ell_1}[\boldsymbol{w},\rho]:=\{\boldsymbol{z}\in\mathbb{R}^l:\sum_{i=1}^l w_i|z_i|\leq\rho\}$ that lies closest to $\boldsymbol{\theta}$ in the Euclidean sense. If $\boldsymbol{\theta}$ lies within the ball then it coincides with its projection. Given the weights and the value of $\rho$, the following iterations provide the projection.

**Algorithm 10.5 (Projection onto the weighted $\ell_1$ ball $B_{\ell_1}[\boldsymbol{w},\rho]$).**

1. Form the vector $[|\theta_1|/w_1,\ldots,|\theta_l|/w_l]^{\mathrm{T}}\in\mathbb{R}^l$.
2. Sort the previous vector in a nonascending order, so that $|\theta_{\tau(1)}|/w_{\tau(1)}\geq\ldots\geq|\theta_{\tau(l)}|/w_{\tau(l)}$. The notation $\tau$ stands for the permutation, which is implicitly defined by the sorting operation. Keep in mind the inverse $\tau^{-1}$, which is the index of the position of the element in the original vector.
3. $r_1:=l$.
4. Let $m=1$. While $m\leq l$, do
   (a) $m_*:=m$.
   (b) Find the maximum $j_*$ among those $j\in\{1,2,\ldots,r_m\}$ such that $\dfrac{|\theta_{\tau(j)}|}{w_{\tau(j)}}>\dfrac{\sum_{i=1}^{r_m}w_{\tau(i)}|\theta_{\tau(i)}|-\rho}{\sum_{i=1}^{r_m}w_{\tau(i)}^2}$.

**(c)** If $j_* = r_m$ then break the loop.
**(d)** Otherwise set $r_{m+1} := j_*$.
**(e)** Increase $m$ by 1 and go back to Step 4a.

**5.** Form the vector $\hat{\boldsymbol{p}} \in \mathbb{R}^{r_{m_*}}$ whose $j$-th component, $j = 1, \ldots, r_{m_*}$, is given by

$$\hat{p}_j := |\theta_{\tau(j)}| - \frac{\sum_{i=1}^{r_{m_*}} w_{\tau(i)} |\theta_{\tau(i)}| - \rho}{\sum_{i=1}^{r_{m_*}} w_{\tau(i)}^2} w_{\tau(j)}.$$

**6.** Use the inverse mapping $\tau^{-1}$ to insert the element $\hat{p}_j$ into the $\tau^{-1}(j)$ position of the $l$-dimensional vector $\boldsymbol{p}$, $\forall j \in \{1, 2, \ldots r_{m_*}\}$, and fill in the rest with zeros.

**7.** The desired projection is $P_{B_{\ell_1}[\boldsymbol{w},\rho]}(\boldsymbol{\theta}) = [\text{sgn}(\theta_1)p_1, \ldots, \text{sgn}(\theta_l)p_l]^{\text{T}}$.

*Remarks 10.3.*

- *Generalized Thresholding Rules*: Projections onto both $\ell_1$ and weighted $\ell_1$ balls impose convex sparsity inducing constraints via properly performed soft thresholding operations. More recent advances within the SpAPSM framework [78, 109] allow the substitution of $P_{B_{\ell_1}[\rho]}$ and $P_{B_{\ell_1}[\boldsymbol{w},\rho]}$ with a *generalized thresholding*, built around the notions of SCAD, nonnegative garrote, and a number of thresholding functions corresponding to the *nonconvex*, $\ell_p$, $p < 1$ penalties. Moreover, it is shown that such generalized thresholding (GT) operators are nonlinear mappings with *their fixed point set being a union of subspaces*, that is, the nonconvex object that lies at the heart of any sparsity-promoting technique. Such schemes are very useful for low values of $q$, where one can improve upon the performance obtained by the LMS-based AdCoSAMP, at comparable complexity levels.
- A comparative study of various online sparsity-promoting low complexity schemes, including the proportionate LMS, in the context of the echo cancelation task, is given in [80]. It turns out that the SpAPSM-based schemes outperform LMS-based sparsity-promoting algorithms.
- More algorithms and methods that involve sparsity-promoting regularization, in the context of more general convex loss functions, compared to the squared error, are discussed in Chapter 8, where related references are provided.
- *Distributed Sparsity-Promoting Algorithms*: Besides the algorithms reported so far, a number of algorithms in the context of distributed learning have also appeared in the literature. As pointed out in Chapter 5, algorithms complying to the consensus as well as the diffusion rationale have been proposed (see, e.g., [29, 39, 89, 102]). A review of such algorithms appears in [30].

**Example 10.2** (Time-varying signal)**.** In this example, the performance curves of the most popular online algorithms, mentioned before, are studied in the context of a time varying environment. A typical simulation setup, which is commonly adopted by the adaptive filtering community in order to study the tracking agility of an algorithm, is that of an unknown vector that undergoes an abrupt change after a number of observations. Here, we consider a signal, $\boldsymbol{s}$, with a sparse wavelet representation, that is, $\boldsymbol{s} = \Psi\boldsymbol{\theta}$, where $\Psi$ is the corresponding transformation matrix. In particular, we set $l = 1024$ with 100 nonzero wavelet coefficients. After 1500 measurements (observations), 10 arbitrarily picked wavelet coefficients change their values to new ones, selected uniformly at random from the interval $[-1, 1]$. Note that this may affect the sparsity level of the signal, and we can now end with up to 110 nonzero coefficients. A total of $N = 3000$ sensing vectors are used, which result from the wavelet transform of the input vectors $\boldsymbol{x}_n \in \mathbb{R}^l$, $n = 1, 2, \ldots, 3000$, having elements drawn from $\mathcal{N}(0, 1)$. In this way,

the online algorithms do not estimate the signal itself, but its sparse wavelet representation, $\boldsymbol{\theta}$. The observations are corrupted by additive white Gaussian noise of variance $\sigma_n^2 = 0.1$. Regarding SpAPSM, the extrapolation parameter $\mu_n$ is set equal to $1.8 \times \mathcal{M}_n$, $\omega_i^{(n)}$ are all getting the same value $1/q$, the hyperslabs parameter $\epsilon$ was set equal to $1.3\sigma_n$, and $q = 390$. The parameters for all algorithms were selected in order to optimize their performance. Because the sparsity level of the signal may change (from $k = 100$ up to $k = 110$) and because in practice it is not possible to know in advance the exact value of $k$, we feed the algorithms with an overestimate, $k$, of the true sparsity value, and in particular we used $\hat{k} = 150$ (i.e., 50% overestimation up to the 1500th iteration).

The results are shown in Figure 10.11. Note the enhanced performance obtained via the SpAPSM algorithm. However, it has to be pointed out that the complexity of the AdCoSAMP is much lower compared to the other two algorithms, for the choice of $q = 390$ for the SpAPSM. The interesting observation is that SpAPSM achieves a better performance compared to OCCD-TWL, albeit at significantly lower complexity. If on the other hand complexity is of major concern, use of SpAPSM offers the flexibility to use generalized thresholding operators, which lead to improved performance for small values of $q$, at complexity comparable to that of LMS-based sparsity promoting algorithms [79, 80].

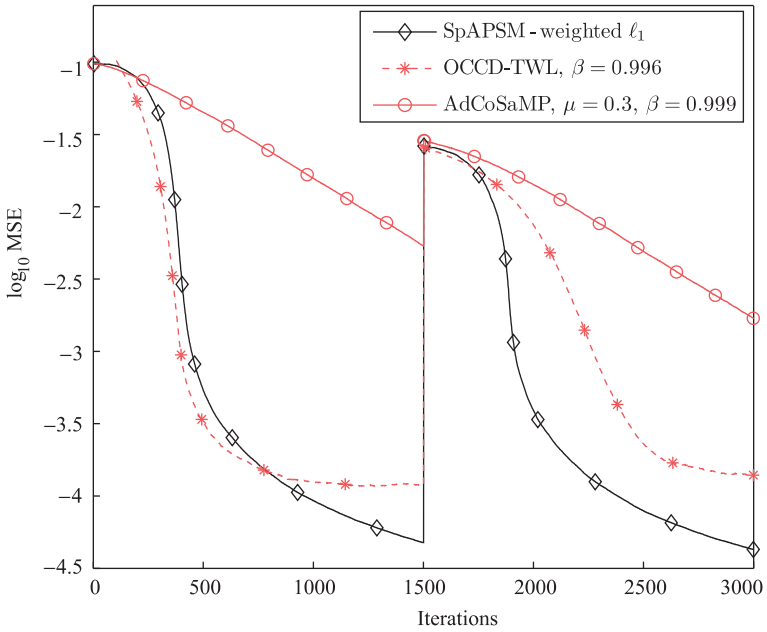


**FIGURE 10.11**

MSE learning curves for AdCoSAMP, SpAPSM, and OCCD-TWL for the simulation in Example 10.2. The vertical axis shows the $\log_{10}$ of the mean-square, that is, $\log_{10}\frac{1}{2}||\boldsymbol{s} - \boldsymbol{\Psi}\boldsymbol{\theta}_n||^2$, and the horizontal shows the time index. At time $n = 1500$, the system undergoes a sudden change.

## 10.5 LEARNING SPARSE ANALYSIS MODELS

All our discussion so far has been spent in the terrain of signals that are either sparse themselves or that can be sparsely represented in terms of the atoms of a dictionary in a synthesis model, as introduced in (9.16), that is,

$$\boldsymbol{s} = \sum_{i \in \mathcal{I}} \theta_i \boldsymbol{\psi}_i.$$

As a matter of fact, most of the research activity has been focused on the synthesis model. This may be partly due to the fact that the synthesis modeling path may provide a more intuitively appealing structure to describe the generation of the signal in terms of the elements (atoms) of a dictionary. Recall from Section 9.9 that the sparsity assumption was imposed on $\boldsymbol{\theta}$ in the synthesis model and the corresponding optimization task was formulated in (9.38) and (9.39) for the exact and noisy cases, respectively.

However, this is not the only way to approach the task of sparse modeling. Very early in this chapter, in Section 9.4, we referred to the analysis model

$$\boldsymbol{S} = \Phi^H \boldsymbol{s}$$

and pointed out that in a number of real-life applications, the resulting transform $\boldsymbol{S}$ is sparse. To be fair, the most orthodox way to deal with the underlying model sparsity would be to consider $\left\| \Phi^H \boldsymbol{s} \right\|_0$. Thus, if one wants to estimate $\boldsymbol{s}$, a very natural way would be to cast the related optimization task as

$$\begin{aligned} \min_{\boldsymbol{s}} \quad & \left\| \Phi^H \boldsymbol{s} \right\|_0, \\ \text{s.t.} \quad & \boldsymbol{y} = X\boldsymbol{s}, \text{ or } \|\boldsymbol{y} - X\boldsymbol{s}\|_2^2 \leq \epsilon, \end{aligned} \tag{10.28}$$

depending on whether the measurements via a sensing matrix, $X$, are exact or noisy. Strictly speaking, the total variation minimization approach, which was used in Example 10.1, falls under this analysis model formulation umbrella, because what is minimized is the $\ell_1$ norm of the gradient transform of the image.

The optimization tasks in either of the two formulations given in (10.28) build around the assumption that the signal of interest has *sparse analysis representation*. The obvious question that is now raised is whether the optimization tasks in (10.28) and their counterparts in (9.38) or (9.39) are any different. One of the first efforts to shed light on this problem was in [51]. There, it was pointed out that the two tasks, though related, are in general different. Moreover, their comparative performance depends on the specific problem at hand. However, it is fair to say that this is a new field of research and more definite conclusions are currently being shaped. An easy answer can be obtained for the case where the involved dictionary corresponds to an orthonormal transformation matrix (e.g., DFT). In this case, we already know that the analysis and synthesis matrices are related as

$$\Phi = \Psi = \Psi^{-H},$$

which leads to an equivalence between the two previously stated formulations. Indeed, for such a transform we have

$$\underbrace{\boldsymbol{S} = \Phi^H \boldsymbol{s}}_{\text{Analysis}} \quad \Leftrightarrow \quad \underbrace{\boldsymbol{s} = \Phi \boldsymbol{S}}_{\text{Synthesis}}.$$

Using the last formula into the (10.28), the tasks in (9.38) or (9.39) are readily obtained by replacing $\boldsymbol{\theta}$ by $\boldsymbol{s}$. However, this reasoning cannot be extended to the case of overcomplete dictionaries; in these cases, the two optimization tasks may lead to different solutions.

The previous discussion concerning the comparative performance between the synthesis or analysis-based sparse representations is not only of "philosophical" value. It turns out that, often in practice, the nature of certain overcomplete dictionaries does not permit the use of the synthesis-based formulation. These are the cases where the columns of the overcomplete dictionary exhibit a high degree of dependence; that is, the coherence of the matrix, as defined in Section 9.6.1, has large values. Typical examples of such overcomplete dictionaries are the Gabor frames, the curvelet frames, and the oversampled DFT. The use of such dictionaries lead to enhanced performance in a number of applications (e.g., [111, 112]). Take as an example the case of our familiar DFT transform. This transform provides a representation of our signal samples in terms of sampled exponential sinusoids, whose integral frequencies are multiples of $\frac{2\pi}{l}$, that is,

$$\boldsymbol{s} := \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{l-1} \end{bmatrix} = \sum_{i=0}^{l-1} S_i \boldsymbol{\psi}_i, \tag{10.29}$$

where $S_i$ are the DFT coefficients and $\boldsymbol{\psi}_i$ is the sampled sinusoid with frequency equal to $\frac{2\pi}{l} i$, that is,

$$\boldsymbol{\psi}_i = \begin{bmatrix} 1 \\ \exp\left(-j\frac{2\pi}{l} i\right) \\ \vdots \\ \exp\left(-j\frac{2\pi}{l} i(l-1)\right) \end{bmatrix}. \tag{10.30}$$

However, this is not necessarily the most efficient representation. For example, it is highly unlikely that a signal comprises only integral frequencies and only such signals can result in a sparse representation using the DFT basis. Most probably, in general, there will be frequencies lying in between the frequency samples of the DFT basis that result in nonsparse representations. Using these extra frequencies, a much better representation of the frequency content of the signal can be obtained. However, in such a dictionary, the atoms are no longer linearly independent, and the coherence of the respective (dictionary) matrix increases.

Once a dictionary exhibits high coherence, then there is no way of finding a sensing matrix, $X$, so that $X\Psi$ obeys the RIP. Recall that at the heart of sparsity-aware learning lies the concept of stable embedding, that allows the recovery of a vector/signal after projecting it on a lower dimensional space; which is what all the available conditions (e.g., RIP), guarantee. However, no stable embedding is possible with highly coherent dictionaries. Take as an extreme example the case where the first and second atoms are identical. Then no sensing matrix $X$ can achieve a signal recovery that distinguishes the vector $[1, 0, \ldots, 0]^{\mathrm{T}}$ from $[0, 1, 0, \ldots, 0]^{\mathrm{T}}$. Can then one conclude that for highly coherent overcomplete dictionaries, compressed sensing techniques are not possible? Fortunately, the answer to this is negative. After all, our goal in compressed sensing has always been the recovery of the signal $\boldsymbol{s} = \Psi\boldsymbol{\theta}$ and not the identification of the sparse vector $\boldsymbol{\theta}$ in the synthesis model representation. The latter was just a means to an end. While the unique recovery of $\boldsymbol{\theta}$ cannot be guaranteed for highly coherent dictionaries, this does

not necessarily cause any problems for the recovery of $\boldsymbol{s}$, using a small set of measurement samples. The escape route will come by considering the analysis model formulation.

### 10.5.1 COMPRESSED SENSING FOR SPARSE SIGNAL REPRESENTATION IN COHERENT DICTIONARIES

Our goal in this subsection is to establish conditions that guarantee recovery of a signal vector, which accepts a sparse representation in a redundant and coherent dictionary, using a small number of signal-related measurements. Let the dictionary at hand be a tight frame, $\Psi$ (Appendix 10.7). Then, our signal vector is written as

$$\boldsymbol{s} = \Psi\boldsymbol{\theta}, \tag{10.31}$$

where $\boldsymbol{\theta}$ is assumed to be $k$-sparse. Recalling the properties of a tight frame, as they are summarized in Appendix 10.7, the coefficients in the expansion (10.31) can be written as $\langle \boldsymbol{\psi}_i, \boldsymbol{s} \rangle$, and the respective vector as

$$\boldsymbol{\theta} = \Psi^{\mathrm{T}}\boldsymbol{s},$$

because a tight frame is self-dual. Then, the analysis counterpart of the synthesis formulation in (9.39) can be cast as

$$\begin{aligned} \min_{\boldsymbol{s}} \quad & \left\| \Psi^{\mathrm{T}}\boldsymbol{s} \right\|_1, \\ \text{s.t.} \quad & \|\boldsymbol{y} - X\boldsymbol{s}\|_2^2 \leq \epsilon. \end{aligned} \tag{10.32}$$

The goal now is to investigate the accuracy of the recovered solution to this convex optimization task. It turns out that similar strong theorems are also valid for this problem, as with the case of the synthesis formulation, which was studied in Chapter 9.

**Definition 10.1.** Let $\Sigma_k$ be the union of all subspaces spanned by all subsets of $k$ columns of $\Psi$. A sensing matrix, $X$, obeys the restricted isometry property adapted to $\Psi$, ($\Psi$-RIP) with $\delta_k$, if

$$\boxed{(1-\delta_k)\,\|\boldsymbol{s}\|_2^2 \leq \|X\boldsymbol{s}\|_2^2 \leq (1+\delta_k)\,\|\boldsymbol{s}\|_2^2 : \quad \Psi - \text{RIP Condition},} \tag{10.33}$$

for all $\boldsymbol{s} \in \Sigma_k$.

The union of subspaces, $\Sigma_k$, is the image under $\Psi$ of all $k$-sparse vectors. This is the difference with the RIP definition given in Section 9.7.2. All the random matrices discussed earlier in this chapter can be shown to satisfy this form of RIP, with overwhelming probability, provided the number of observations, $N$, is at least of the order of $k\ln(l/k)$. We are now ready to establish the main theorem concerning our $\ell_1$ minimization task.

**Theorem 10.2.** *Let $\Psi$ be an arbitrary tight frame and $X$ a sensing matrix that satisfies the $\Psi$-RIP with $\delta_{2k} \leq 0.08$, for some positive $k$. Then the solution, $\boldsymbol{s}_*$, of the minimization task in (10.32) satisfies the property*

$$\|\boldsymbol{s} - \boldsymbol{s}_*\|_2 \leq C_0 k^{-\frac{1}{2}} \left\| \Psi^{\mathrm{T}}\boldsymbol{s} - (\Psi^{\mathrm{T}}\boldsymbol{s})_k \right\|_1 + C_1\sqrt{\epsilon}, \tag{10.34}$$

*where $C_0, C_1$ are constants depending on $\delta_{2k}$, and $(\Psi^{\mathrm{T}}\boldsymbol{s})_k$ denotes the best $k$-sparse approximation of $\Psi^{\mathrm{T}}\boldsymbol{s}$, that results by setting all but the $k$ largest in magnitude components of $\Psi^{\mathrm{T}}\boldsymbol{s}$ equal to zero.*

The bound in (10.34) is the counterpart of that given in (9.36). In other words, the previous theorem states that if $\Psi^{\mathrm{T}}\boldsymbol{s}$ decays rapidly, then $\boldsymbol{s}$ can be reconstructed from just a few (compared to the signal length $l$) observations. The theorem was first given in [25] and it is the first time that such a theorem provides results for the sparse analysis model formulation in a general context.

### 10.5.2 COSPARSITY

In the sparse synthesis formulation, one searches for a solution in a union of subspaces that are formed by all possible combinations of $k$ columns of the dictionary, $\Psi$. Our signal vector lies in one of these subspaces—the one that is spanned by the columns of $\Psi$ whose indices lie in the support set (Section 10.2.1). In the sparse analysis approach, things get different. The kick-off point is the sparsity of the transform $\boldsymbol{S} := \Phi^{\mathrm{T}}\boldsymbol{s}$, where $\Phi$ defines the transformation matrix or analysis operator. Because $\boldsymbol{S}$ is assumed to be sparse, there exists an index set $\mathcal{I}$ such that $\forall i \in \mathcal{I}$, $S_i = 0$. In other words, $\forall i \in \mathcal{I}$, $\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{s} := \langle \boldsymbol{\phi}_i, \boldsymbol{s} \rangle = 0$, where $\boldsymbol{\phi}_i$ stands for the $i$th column of $\Phi$. Hence, the subspace in which $\boldsymbol{s}$ lives is the orthogonal complement of the subspace formed by those columns of $\Phi$ that correspond to a zero in the transform vector $\boldsymbol{S}$. Assume, now, that $\mathrm{card}(\mathcal{I}) = C_o$. The signal, $\boldsymbol{s}$, can be identified by searching on the *orthogonal complements* of the subspaces formed by all possible combinations of $C_o$ columns of $\Phi$, that is,

$$\langle \boldsymbol{\phi}_i, \boldsymbol{s} \rangle = 0, \quad \forall i \in \mathcal{I}.$$

The difference between the synthesis and analysis problems is illustrated in Figure 10.12. To facilitate the theoretical treatment of this new setting, the notion of *cosparsity* was introduced in [92].

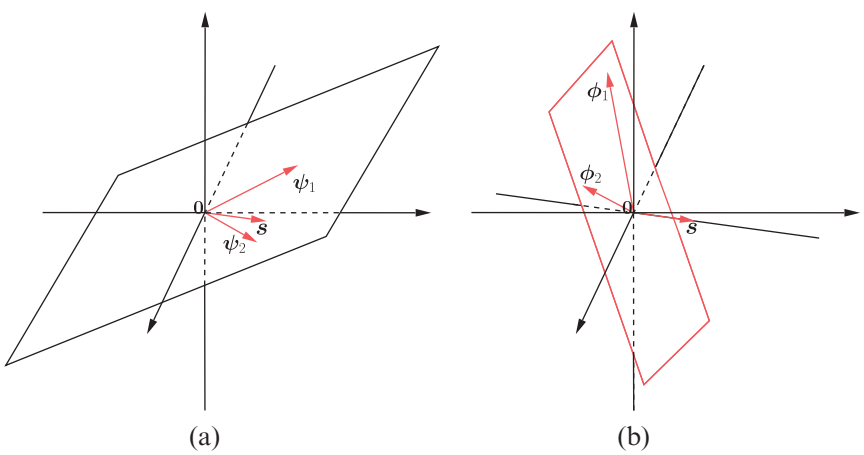


**FIGURE 10.12**

Searching for a sparse vector $\boldsymbol{s}$. (a) In the synthesis model, the sparse vector lies in subspaces formed by combinations of $k$ (in this case $k = 2$) columns of the dictionary $\Psi$. (b) In the analysis model, the sparse vector lies in the orthogonal complement of the subspace formed by $C_o$ (in this case $C_o = 2$) columns of the transformation matrix $\Phi$.

**Definition 10.2.** The cosparsity of a signal $\boldsymbol{s} \in \mathbb{R}^l$ with respect to a $p \times l$ matrix $\Phi^{\mathrm{T}}$ is defined as

$$C_o := p - \left\| \Phi^{\mathrm{T}} \boldsymbol{s} \right\|_0 . \tag{10.35}$$

In words, the cosparsity is the number of zeros in the obtained transform vector $\boldsymbol{S} = \Phi^{\mathrm{T}} \boldsymbol{s}$; in contrast, the sparsity measures the number of the nonzero elements of the respective sparse vector. If one assumes that $\Phi$ has "full spark,"[3] that is, $l + 1$, then any $l$ of the columns of $\Phi$, and thus any $l$ rows of $\Phi^{\mathrm{T}}$, are guaranteed to be independent. This indicates that for such matrices, the maximum value that cosparsity can take is equal to $C_o = l - 1$. Otherwise, the existence of $l$ zeros will necessarily correspond to a zero signal vector. Higher cosparsity levels are possible by relaxing the full spark requirement.

Let now the cosparsity of our signal with respect to a matrix $\Phi^{\mathrm{T}}$ be $C_o$. Then, in order to dig out the signal from the subspace in which it is hidden, one must form all possible combinations of $C_o$ columns of $\Phi$ and search in their orthogonal complements. In the case that $\Phi$ is full rank, we have seen previously that $C_o < l$, and, hence, any set of $C_o$ columns of $\Phi$ are linearly independent. In other words, the dimension of the span of those columns is $C_o$. As a result, the dimensionality of the orthogonal complement, into which we search for $\boldsymbol{s}$, is $l - C_o$.

We have by now accumulated enough information to elaborate a bit more on the statement made before concerning the different nature of the synthesis and analysis tasks. Let us consider a synthesis task using an $l \times p$ dictionary, and let $k$ be the sparsity level in the corresponding expansion of a signal in terms of this dictionary. The dimensionality of the subspaces in which the solution is sought is $k$ ($k$ is assumed to be less than the spark of the respective matrix). Let us keep the same dimensionality for the subspaces in which we are going to search for a solution in an analysis task. Hence, in this case $C_o = l - k$ (assuming a full spark matrix). Also, for the sake of comparison assume that the analysis matrix is $p \times l$. Solving the synthesis task, one has to search $\binom{p}{k}$ subspaces, while solving the analysis task one has to search $\binom{p}{C_o = l-k}$ subspaces. These are two different numbers; assuming that $k \ll l$ and also that $l < p/2$, which are natural assumptions for overcomplete dictionaries, then the latter of the two numbers is much larger than the former (use your computer to play with some typical values). In other words, there are many more analysis than synthesis low-dimensional subspaces for which to search. The large number of low-dimensional subspaces makes the algorithmic recovery of a solution from the analysis model a tougher task [92]. However, it might reveal a much stronger descriptive power of the analysis model compared to the synthesis one.

Another interesting aspect that highlights the difference between the two approaches is the following. Assume that the synthesis and analysis matrices are related as $\Phi = \Psi$, as was the case for tight frames. Under this assumption, $\Phi^{\mathrm{T}} \boldsymbol{s}$ provides a set of coefficients for the synthesis expansion in terms of the atoms of $\Phi = \Psi$. Moreover, if $\left\| \Phi^{\mathrm{T}} \boldsymbol{s} \right\|_0 = k$, then the $\Phi^{\mathrm{T}} \boldsymbol{s}$ is a possible $k$-sparse solution for the synthesis model. However, there is no guarantee that this is the sparsest one.

It is now time to investigate whether conditions that guarantee uniqueness of the solution for the sparse analysis formulation can be derived. The answer is in the affirmative, and it has been established in [92] for the case of exact measurements.

**Lemma 10.1.** *Let $\Phi$ be a transformation matrix of full spark. Then, for almost all $N \times l$ sensing matrices and for $N > 2(l - C_o)$, the equation*

$$\boldsymbol{y} = X\boldsymbol{s},$$

*has at most one solution with cosparsity at least $C_o$.*

[3] Recall by Definition 9.2 that spark($\Phi$) is defined for an $l \times p$ matrix $\Phi$ with $p \geq l$ and of full rank.

The above lemma guarantees the uniqueness of the solution, if one exists, of the optimization

$$\begin{aligned} \min_{\boldsymbol{s}} \quad & \left\|\Phi^{\mathrm{T}}\boldsymbol{s}\right\|_0 \\ \text{s.t.} \quad & \boldsymbol{y} = X\boldsymbol{s}. \end{aligned} \tag{10.36}$$

However, solving the previous $\ell_0$ minimization task is a difficult one, and we know that its synthesis counterpart has been shown to be NP-hard, in general. Its relaxed convex relative is the $\ell_1$ minimization

$$\begin{aligned} \min_{\boldsymbol{s}} \quad & \left\|\Phi^{\mathrm{T}}\boldsymbol{s}\right\|_1 \\ \text{s.t.} \quad & \boldsymbol{y} = X\boldsymbol{s}. \end{aligned} \tag{10.37}$$

In [92], conditions are derived that guarantee the equivalence of the $\ell_0$ and $\ell_1$ tasks, in (10.36) and (10.37), respectively; this is done in a way similar to that for the sparse synthesis modeling. Also, in [92], a greedy algorithm inspired by the orthogonal matching pursuit, discussed in Section 10.2.1, has been derived. A thorough study on greedy-like algorithms applicable to the cosparse model can be found in [62]. In [103] an iterative analysis thresholding is proposed and theoretically investigated. Other algorithms that solve the $\ell_1$ optimization in the analysis modeling framework can be found in, for example, [21, 49, 108]. NESTA can also be used for the analysis formulation. Moreover, a critical aspect affecting the performance of algorithms obeying the cosparse analysis model is the choice of the analysis matrix $\Phi$. It turns out that it is not always the best practice to use fixed and predefined matrices. As a promising alternative, problem-tailored analysis matrices can be learned using the available data (e.g., [106, 119]).

## 10.6 A CASE STUDY: TIME-FREQUENCY ANALYSIS

The goal of this section is to demonstrate how all the previously stated theoretical findings can be exploited in the context of a real application. Sparse modeling has been applied to almost everything. So picking up a typical application would not be easy. We preferred to focus on a less "publicized" application, that of analyzing echolocation signals emitted by bats. However, the analysis will take place within the framework of time-frequency representation, which is one of the research areas that significantly inspired the evolution of compressed sensing theory. Time-frequency analysis of signals has been a field of intense research for a number of decades, and it is one of the most powerful signal processing tools. Typical applications include speech processing, sonar sounding, communications, biological signals, and EEG processing, to name but a few (see, e.g., [13, 20, 57]).

### *Gabor transform and frames*

It is not our intention to present the theory behind the Gabor transform. Our goal is to outline some basic related notions and use them as a vehicle for the less familiar reader to better understand how redundant dictionaries are used and to get better acquainted with their potential performance benefits.

The Gabor transform was introduced in the mid-1940s by Dennis Gabor (1900–1979), a Hungarian-British engineer. His most notable scientific achievement was the invention of holography, for which he won the Nobel Prize for Physics in 1971.

The discrete version of the Gabor transform can be seen as a special case of the short time Fourier transform (STFT) (e.g., [57, 87]). In the standard DFT transform, the full length of a time sequence, comprising $l$ samples, is used all in "one go" in order to compute the corresponding frequency content. However, the latter can be time varying, so the DFT will provide an average information, which cannot be of much use. The Gabor transform (and the STFT in general) introduces time localization via the use of a window function, which slides along the signal segment in time, and at each time instant focuses on a different part of the signal. This is a way that allows one to follow the slow time variations, which take place in the frequency domain. The time localization in the context of the Gabor transform is achieved via a Gaussian window function, that is,

$$g(n) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{n^2}{2\sigma^2}\right). \tag{10.38}$$

Figure 10.13a shows the Gaussian window, $g(n-m)$, centered at time instant $m$. The choice of the window spreading factor, $\sigma$, will be discussed later on.

Let us now construct the atoms of the Gabor dictionary. Recall that in the case of the signal representation in terms of the DFT in (10.29), each frequency is represented only once, by the corresponding sampled sinusoid, (10.30). In the Gabor transform, each frequency appears $l$ times; the corresponding sampled sinusoid is multiplied by the Gaussian window sequence, each time shifted by one sample. Thus, at the $i$th frequency bin, we have $l$ atoms, $\boldsymbol{g}^{(m,i)}, m = 0, 1, \ldots, l-1$, with elements given by

$$g^{(m,i)}(n) = g(n-m)\psi_i(n), \quad n, m, i = 0, 1, \ldots, l-1, \tag{10.39}$$

where $\psi_i(n)$ is the $n$th element of the vector $\boldsymbol{\psi}_i$ in (10.30). This results to an overcomplete dictionary comprising $l^2$ atoms in the $l$-dimensional space. Figure 10.13b illustrates the effect of multiplying

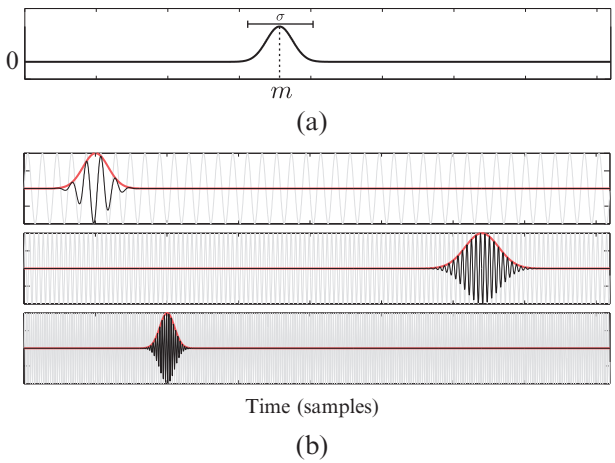


**FIGURE 10.13**

(a) The Gaussian window with spreading factor $\sigma$ centered at time instant $m$. (b) Pulses obtained by windowing three different sinusoids with Gaussian windows of different spread and applied at different time instants.

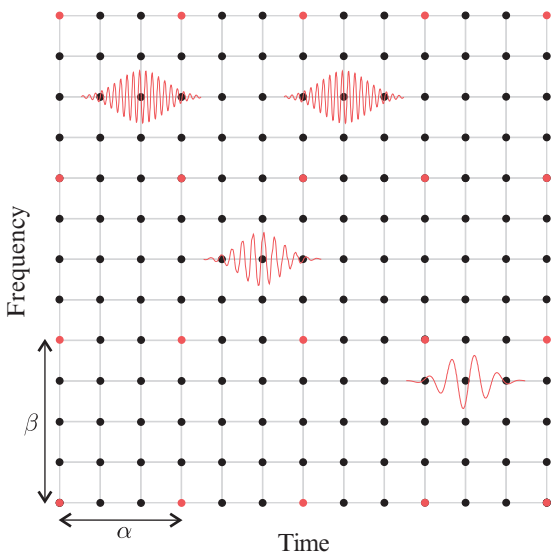


**FIGURE 10.14**

Each atom of the Gabor dictionary corresponds to a node in the time-frequency grid. That is, it is a sampled windowed sinusoid whose frequency and location in time are given by the coordinates of the respective node. In practice, this grid may be subsampled by factors $\alpha$ and $\beta$ for the two axes respectively, in order to reduce the number of the involved atoms.

different sinusoids with Gaussian pulses of different spread and at different time delays. Figure 10.14 is a graphical interpretation of the atoms involved in the Gabor dictionary. Each node, $(m, i)$, in this time-frequency plot, corresponds to an atom of frequency equal to $\frac{2\pi}{l}i$ and delay equal to $m$.

Note that the windowing of a signal of finite duration inevitably introduces boundary effects, especially when the delay $m$ gets close to the time segment edges, 0 and $l-1$. A solution that facilitates the theoretical analysis is to use a modulo $l$ arithmetic to wrap around at the edge points (this is equivalent to extending the signal periodically); see, for example, [113].

Once the atoms have been defined, they can be stacked one next to the other to form the columns of the $l \times l^2$ Gabor dictionary, $G$. It can be shown that the Gabor dictionary is a tight frame, [125].

### *Time-frequency resolution*

By definition of the Gabor dictionary, it is readily understood that the choice of the window spread, as measured by $\sigma$, must be a critical factor, since it controls the localization in time. As known from our Fourier transform basics, when the pulse becomes short, in order to increase the time resolution, its corresponding frequency content spreads out, and vice versa. From Heisenberg's principle, we know that we can never achieve high time and frequency resolution simultaneously; one is gained at the expense of the other. It is here where the Gaussian shape in the Gabor transform is justified. It can be shown that the Gaussian window gives the optimal trade-off between time and frequency resolution, [57, 87]. The time-frequency resolution trade-off is demonstrated in Figure 10.15, where three sinusoids are shown windowed with different pulse durations. The diagram shows the corresponding spread in

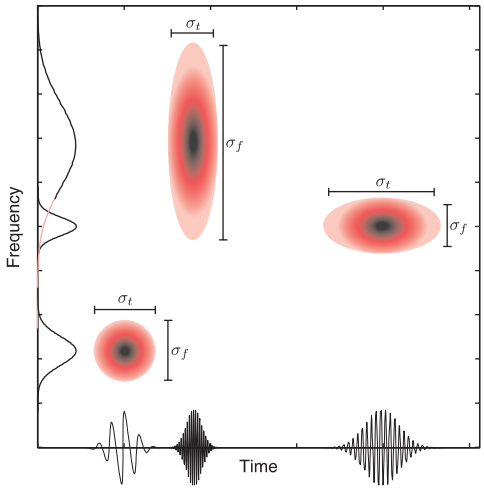


**FIGURE 10.15**

The shorter the width of the pulsed (windowed) sinusoid is in time, the wider the spread of its frequency content around the frequency of the sinusoid. The Gaussian-like curves along the frequency axis indicate the energy spread in frequency of the respective pulses. The values of $\sigma_t$ and $\sigma_f$ indicate the spread in time and frequency, respectively.

the time-frequency plot. The value of $\sigma_t$ indicates the time spread and $\sigma_f$ the spread of the respective frequency content around the basic frequency of each sinusoid.

### *Gabor frames*

In practice, $l^2$ can take large values, and it is desirable to see whether one can reduce the number of the involved atoms without sacrificing the frame-related properties. This can be achieved by an appropriate subsampling, as illustrated in Figure 10.14. We only keep the atoms that correspond to the red nodes. That is, we subsample by keeping every $\alpha$ nodes in time and every $\beta$ nodes in frequency in order to form the dictionary, that is,

$$G_{(\alpha,\beta)} = \{\boldsymbol{g}^{(m\alpha,i\beta)}\}, \quad m = 0, 1, \ldots, \frac{l}{\alpha} - 1, \; i = 0, 1, \ldots, \frac{l}{\beta} - 1,$$

where $\alpha$ and $\beta$ are divisors of $l$. Then, it can be shown (e.g., [57]) that if $\alpha\beta < l$ the resulting dictionary retains its frame properties. Once $G_{(\alpha,\beta)}$ is obtained, the canonical dual frame is readily available via (10.46) (adjusted for complex data), from which the corresponding set of expansion coefficients, $\boldsymbol{\theta}$, results.

### *Time-frequency analysis of echolocation signals emitted by bats*

Bats are using echolocation for navigation (flying around at night), for prey detection (small insects), and for prey approaching and catching; each bat adaptively changes the shape and frequency content of its calls in order to better serve the previous tasks. Echolocation is used in a similar way for sonars.

Bats emit calls as they fly, and "listen" to the returning echoes in order to build up a sonic map of their surroundings. In this way, bats can infer the distance and the size of obstacles as well as of other flying creatures/insects. Moreover, all bats emit special types of calls, called social calls, which are used for socializing, flirting, and so on. The fundamental characteristics of the echolocation calls, for example the frequency range and average time duration, differ from species to species because, thanks to evolution, bats have adapted their calls in order to become better suited to the environment in which a species operates.

Time-frequency analysis of echolocation calls provides information about the species (species identification) as well as the specific task and behavior of the bats in certain environments. Moreover, the bat-biosonar system is studied in order for humans to learn more about nature and get inspired for subsequent advances in applications such as sonar navigation systems, radars, medical ultrasonic devices, and more.

Figure 10.16 shows a case of a recorded echolocation signal from bats. Zooming at two different parts of the signal, we can observe that the frequency is changing with time. In Figure 10.17, the DFT of the signal is shown, but there is not much information that can be drawn from it except that the signal is compressible in the frequency domain; most of the activity takes place within a short range of frequencies.

Our echolocation signal was a recording of total length $T = 21.845$ ms, [75]. Samples were taken at the sampling frequency $f_s = 750$ KHz, which results in a total of $l = 16384$ samples. Although the signal itself is not sparse in the time domain, we will take advantage of the fact that it is sparse in a transformed domain. We will assume that the signal is sparse in its expansion in terms of the Gabor dictionary.

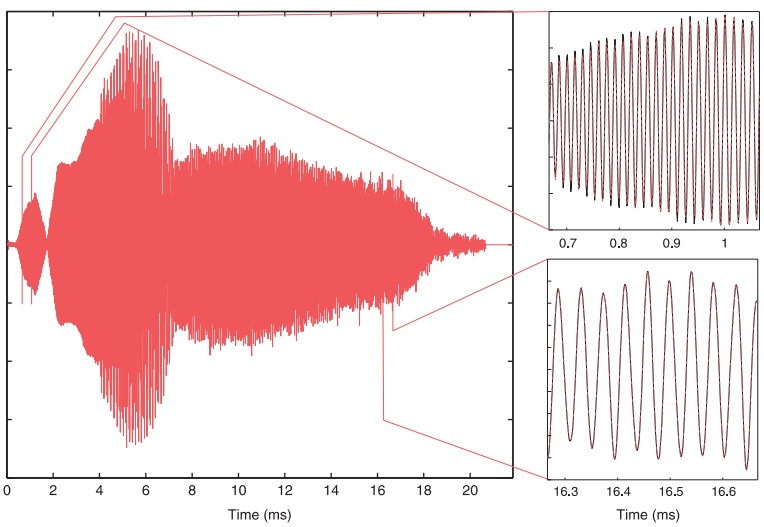


**FIGURE 10.16**

The recorded echolocation signal. The frequency of the signal is time varying, which is indicated by focusing on two different parts of the signal.

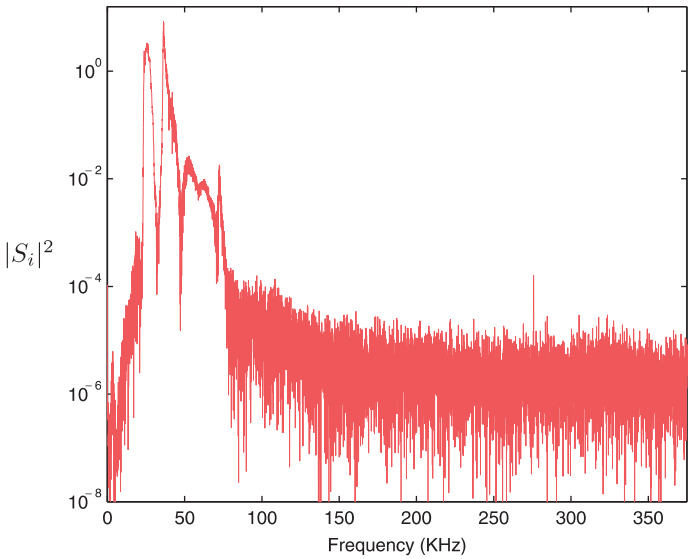


**FIGURE 10.17**

Plot of the energy of the DFT transform coefficients, $S_i$. Observe that most of the frequency activity takes place within a short frequency range.

Our goal in this example is to demonstrate that one does not really need all 16,384 samples to perform time-frequency analysis; all the processing can be carried out using a reduced number of observations, by exploiting the theory of compressed sensing. To form the measurements vector, $\boldsymbol{y}$, the number of observations was chosen to be $N = 2048$. This amounts to a reduction of eight times with respect to the number of available samples. The observations vector was formed as

$$\boldsymbol{y} = X\boldsymbol{s},$$

where $X$ is an $N \times l$ sensing matrix comprising $\pm 1$ generated in a random way. This means that once we obtain $\boldsymbol{y}$, we do not need to store the original samples anymore, leading to a savings in memory. Ideally, one could have obtained the reduced number of observations by sampling directly the analog signal at sub-Nyquist rates, as has already been discussed at the end of Section 9.9. Another goal is to use both the analysis and synthesis models and demonstrate their difference.

Three different spectrograms were computed. Two of them, shown in Figure 10.18b and c, correspond to the reconstructed signals obtained by the analysis (10.37) and the synthesis (9.37) formulations, respectively. In both cases, the NESTA algorithm was used and the $G_{(128,64)}$ frame was employed. Note that the latter dictionary is redundant by a factor of 2. The spectrograms are the result of plotting the time-frequency grid and coloring each node $(t, i)$ according to the energy $|\theta|^2$ of the coefficient associated with the respective atom in the Gabor dictionary. The full Gabor transform was applied on the reconstructed signals to obtain the spectrograms, in order to get better coverage of the time-frequency grid. The scale is logarithmic and the darker areas correspond to larger values. The spectrogram of the original signal obtained via the full Gabor transform is shown

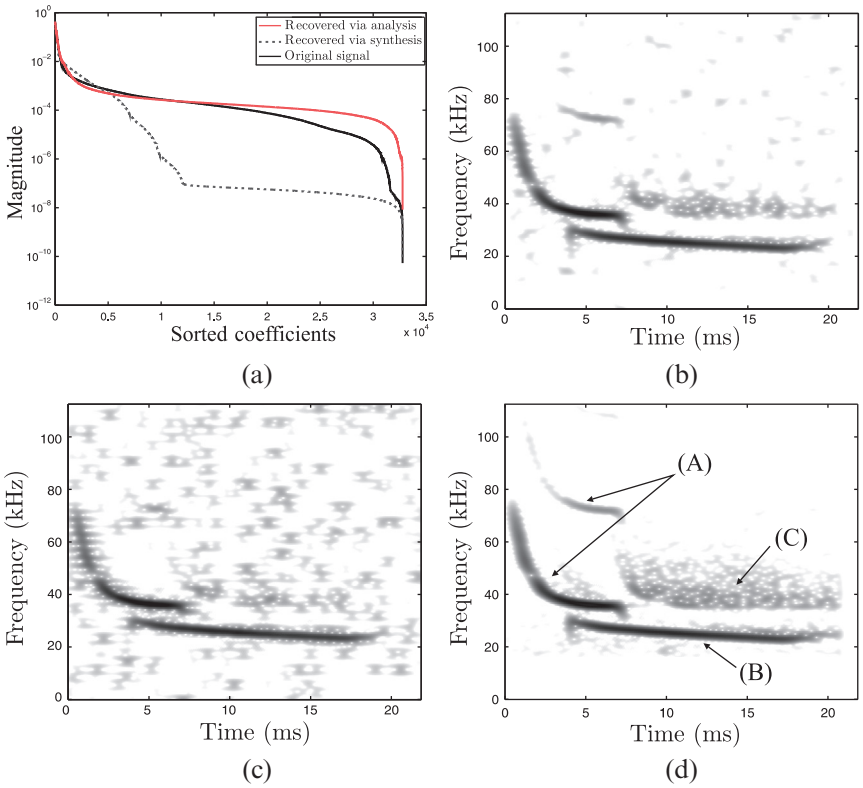


**FIGURE 10.18**

(a) Plot of the magnitude of the coefficients, sorted in decreasing order, in the expansion in terms of the $G_{(128,64)}$ Gabor frame. The results correspond to the analysis and synthesis model formulations. The third curve corresponds to the case of analyzing the original vector signal directly, by projecting it on the dual frame. (b) The spectrogram from the analysis and (c) the spectrogram from the synthesis formulations, respectively. (d) The spectrogram corresponding to $G_{(64,32)}$ frame using the analysis formulation. For all cases, the number of observations used was one eighth of the total number of signal samples. A, B, and C indicate different parts of the signal, as explained in the text.

in Figure 10.18d. It is evident that the analysis model resulted in a more clear spectrogram, which resembles the original one better. When the frame $G_{(64,32)}$ is employed, which is a highly redundant Gabor dictionary comprising $8l$ atoms, then the analysis model results in a recovered signal whose spectrogram is visually indistinguishable from the original one in Figure 10.18d.

Figure 10.18a is the plot of the magnitude of the corresponding Gabor transform coefficients, sorted in decreasing values. The synthesis model provides a sparser representation in the sense that the coefficients decrease much faster. The third curve is the one that results if we multiply the dual frame matrix $\tilde{G}_{(128,64)}$ directly with the vector of the original signal samples, and it is shown for comparison reasons.

To conclude, the curious reader may wonder what these curves in Figure 10.18d mean after all. The call denoted by (A) belongs to a Pipistrellus pipistrellus (!) and the call denoted by (B) is either a social call or belongs to a different species. The (C) is the return echo from the signal (A). The large spread in time of (C) indicates a highly reflective environment [75].

## 10.7 APPENDIX TO CHAPTER 10: SOME HINTS FROM THE THEORY OF FRAMES

In order to remain in the same framework as the one already adopted for this chapter and comply with the notation previously used, we will adhere to the real data case, although everything we will say is readily extended to the complex-valued case by replacing transposition with its Hermitian counterpart.

A frame in a vector space[4] $V \subseteq \mathbb{R}^l$ is a generalization of the notion of a basis. Recall from our linear algebra basics (see also Section 8.15) that *basis* is a set of vectors $\boldsymbol{\psi}_i$, $i \in \mathcal{I}$, with the following two properties: (a) $V = \text{span}\{\boldsymbol{\psi}_i : i \in \mathcal{I}\}$, where the cardinality $\text{card}(\mathcal{I}) = l$; and (b) $\boldsymbol{\psi}_i$, $i \in \mathcal{I}$, are linearly independent. If, in addition, $\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle = \delta_{i,j}$ then the basis is known as orthonormal. If we now relax the second condition and allow $l < \text{card}(\mathcal{I})$, we introduce redundancy in the signal representations, which, as has already been mentioned, can offer a number of advantages in a wide range of applications. However, once redundancy is introduced, we lose uniqueness in the signal representation

$$\boldsymbol{s} = \sum_{i \in \mathcal{I}} \theta_i \boldsymbol{\psi}_i, \tag{10.40}$$

due to the dependency among the vectors $\boldsymbol{\psi}_i$. The question that is now raised is whether there is a simple and systematic way to compute the coefficients $\theta_i$ in the previous expansion.

**Definition 10.3.** The set $\boldsymbol{\psi}_i$, $i \in \mathcal{I}$, which spans a vector space, $V$, is called a *frame* if there exist positive real numbers, $A$ and $B$, such that for every nonzero $\boldsymbol{s} \in V$,

$$0 < A \|\boldsymbol{s}\|_2^2 \leq \sum_{i \in \mathcal{I}} |\langle \boldsymbol{\psi}_i, \boldsymbol{s} \rangle|^2 \leq B \|\boldsymbol{s}\|_2^2, \tag{10.41}$$

where $A$ and $B$ are known as the bounds of the frame.

Note that if $\boldsymbol{\psi}_i$, $i \in \mathcal{I}$, comprise an orthonormal basis, then $A = B = 1$ and (10.41) is the celebrated Parseval's theorem. Thus, (10.41) can be considered a generalization of Parseval's theorem. Looking more carefully, we notice that this is a stability condition that closely resembles our familiar RIP condition in (9.29). Indeed, the upper bound guarantees that the expansion never diverges (this applies to infinite dimensional spaces) and the lower bound guarantees that no nonzero vector, $\|\boldsymbol{s}\| \neq 0$, will never become zero after projecting it along the atoms of the frame. To look at it from a slightly different perspective, form the dictionary matrix

$$\Psi = [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_p],$$

where we used $p$ to denote the cardinality of $\mathcal{I}$. Then, the lower bound in (10.41) guarantees that $\boldsymbol{s}$ can be reconstructed from its transform samples $\Psi^{\mathrm{T}}\boldsymbol{s}$; note that in such a case, if $\boldsymbol{s}_1 \neq \boldsymbol{s}_2$, then their respective transform values will be different.

[4] We constrain our discussion in this section to finite dimensional Euclidean spaces. The theory of frames has been developed for general Hilbert spaces.

It can be shown that if condition (10.41) is valid, then there exists another set of vectors, $\tilde{\boldsymbol{\psi}}_i$, $i \in \mathcal{I}$, known as the *dual frame*, with the following elegant property:

$$\boldsymbol{s} = \sum_{i \in \mathcal{I}} \left\langle \tilde{\boldsymbol{\psi}}_i, \boldsymbol{s} \right\rangle \boldsymbol{\psi}_i = \sum_{i \in \mathcal{I}} \langle \boldsymbol{\psi}_i, \boldsymbol{s} \rangle \tilde{\boldsymbol{\psi}}_i, \quad \forall \boldsymbol{s} \in V. \tag{10.42}$$

Once a dual frame is available, the coefficients in the expansion of a vector in terms of the atoms of a frame are easily obtained. If we form the matrix $\tilde{\Psi}$ of the dual frame vectors, then it easily checks out that since condition (10.42) is true for any $\boldsymbol{s}$, it implies that

$$\tilde{\Psi}\Psi^{\mathrm{T}} = \Psi\tilde{\Psi}^{\mathrm{T}} = I, \tag{10.43}$$

where $I$ is the $l \times l$ identity matrix. Note that all of us have used the property in (10.42), possibly in a disguised form, many times in our professional life. Indeed, consider the simple case of two linearly independent vectors in the two-dimensional space (in order to make things simple). Then, (10.40) becomes

$$\boldsymbol{s} = \theta_1 \boldsymbol{\psi}_1 + \theta_2 \boldsymbol{\psi}_2 = \Psi\boldsymbol{\theta}.$$

Solving for the unknown $\boldsymbol{\theta}$ is nothing but the solution of a linear set of equations; note that the involved matrix $\Psi$ is invertible. Let us rephrase a bit our familiar solution

$$\boldsymbol{\theta} = \Psi^{-1}\boldsymbol{s} := \tilde{\Psi}\boldsymbol{s} := \begin{bmatrix} \tilde{\boldsymbol{\psi}}_1^{\mathrm{T}} \\ \tilde{\boldsymbol{\psi}}_2^{\mathrm{T}} \end{bmatrix} \boldsymbol{s}, \tag{10.44}$$

where $\tilde{\boldsymbol{\psi}}_i^{\mathrm{T}}$, $i = 1, 2$, are the rows of the inverse matrix. Using now the previous notation, it is readily seen that

$$\boldsymbol{s} = \left\langle \tilde{\boldsymbol{\psi}}_1, \boldsymbol{s} \right\rangle \boldsymbol{\psi}_1 + \left\langle \tilde{\boldsymbol{\psi}}_2, \boldsymbol{s} \right\rangle \boldsymbol{\psi}_2.$$

Moreover, note that in this special case of independent vectors, the respective definitions imply

$$\begin{bmatrix} \tilde{\boldsymbol{\psi}}_1^{\mathrm{T}} \\ \tilde{\boldsymbol{\psi}}_2^{\mathrm{T}} \end{bmatrix} [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2] = I,$$

and the dual frame is not only unique but it also fulfills the *biorthogonality* condition, that is,

$$\left\langle \tilde{\boldsymbol{\psi}}_i, \boldsymbol{\psi}_j \right\rangle = \delta_{i,j}. \tag{10.45}$$

In the case of a general frame, the dual frames are neither biorthogonal nor uniquely defined. The latter can also be verified by the condition (10.43) that defines the respective matrices. $\Psi^{\mathrm{T}}$ is a rectangular tall matrix and its left inverse is not unique. There is, however, a uniquely defined dual frame, known as the *canonical* dual frame, given as

$$\tilde{\boldsymbol{\psi}}_i := (\Psi\Psi^{\mathrm{T}})^{-1}\boldsymbol{\psi}_i, \text{ or } \tilde{\Psi} := (\Psi\Psi^{\mathrm{T}})^{-1}\Psi. \tag{10.46}$$

Another family of frames of special type are the *tight frames*. For tight frames, the two bounds in (10.41) are equal, that is, $A = B$. Thus, once a tight frame is available, we can normalize each vector in the frame as

$$\boldsymbol{\psi}_i \longmapsto \frac{1}{\sqrt{A}}\boldsymbol{\psi}_i,$$

which then results to the *Parseval tight frame*; the condition (10.41) now becomes similar in appearance to our familiar Parseval's theorem for orthonormal bases

$$\sum_{i\in\mathcal{I}} |\langle \boldsymbol{\psi}_i, \boldsymbol{s}\rangle|^2 = \|\boldsymbol{s}\|_2^2 . \tag{10.47}$$

Moreover, it can be shown (Problem 10.9) that a Parseval tight frame coincides with its canonical dual frame (that is, it is self-dual) and we can write

$$\boldsymbol{s} = \sum_{i\in\mathcal{I}} \langle \boldsymbol{\psi}_i, \boldsymbol{s}\rangle \boldsymbol{\psi}_i,$$

or in matrix form

$$\tilde{\Psi} = \Psi, \tag{10.48}$$

which is similar with what we know for orthonormal bases; however in this case, orthogonality does not hold.

We will conclude this subsection with a simple example of a Parseval (tight) frame, known as the *Mercedes Benz (MB)*,

$$\Psi = \begin{bmatrix} 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \end{bmatrix}.$$

One can easily check that all the properties of a Parseval tight frame are fulfilled.

If constructing a frame, especially in high-dimensional spaces, sounds a bit difficult, the following theorem (from Naimark, see, for example, [67]) offers a systematic method for such constructions.

**Theorem 10.3.** *A set $\{\boldsymbol{\psi}_i\}_{i\in\mathcal{I}}$ in a Hilbert space $\mathbb{H}_s$ is a Parseval tight frame if and only if it can be obtained via an orthogonal projection, $P_{\mathbb{H}_s} : \mathbb{H} \to \mathbb{H}_s$, of an orthonormal basis $\{\boldsymbol{e}_i\}_{i\in\mathcal{I}}$ in a larger Hilbert space $\mathbb{H}$, such that $\mathbb{H}_s \subset \mathbb{H}$.*

To verify the theorem, check that the MB frame is obtained by orthogonally projecting the three-dimensional orthonormal basis

$$\boldsymbol{e}_1 = \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \boldsymbol{e}_2 = \begin{bmatrix} \sqrt{\frac{2}{3}} \\ -\frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} \end{bmatrix}, \quad \boldsymbol{e}_3 = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix},$$

using the projection matrix

$$P_{\mathbb{H}_s} := \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix}.$$

Observe that the effect of the projection

$$P_{\mathbb{H}_s}[\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3] = \Psi^{\mathrm{T}},$$

is to set $\boldsymbol{e}_3$ to the zero vector.

Frames were introduced by Duffin and Schaeffer in their study on nonharmonic Fourier series in 1952 [47] and they remained rather obscured until they were used in the context of wavelet theory (e.g., [36]). The interested reader can obtain the proofs of what has been said in this section from these references. An introductory review with a lot of engineering flavor can be found in [81], where the major references in the field are given.

## PROBLEMS

**10.1** Show that the step, in a greedy algorithm, that selects the column of the sensing matrix, in order to maximize the correlation between the column and the currently available error vector $\boldsymbol{e}^{(i-1)}$, is equivalent with selecting the column that reduces the $\ell_2$ norm of the error vector.
*Hint.* All the parameters obtained in previous steps are fixed, and the optimization is with respect to the new column as well as the corresponding weighting coefficient in the estimate of the parameter vector.

**10.2** Prove the proposition stating that if there is a sparse solution to the linear system $\boldsymbol{y} = X\boldsymbol{\theta}$ such that

$$k_0 = ||\boldsymbol{\theta}||_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(X)}\right),$$

where $\mu(X)$ is the mutual coherence of $X$, then the column selection procedure in a greedy algorithm will always select a column among the active columns of $X$, which correspond to the support of $\boldsymbol{\theta}$; that is, the columns that take part in the representation of $\boldsymbol{y}$ in terms of the columns of $X$.
*Hint.* Assume that

$$\boldsymbol{y} = \sum_{i=1}^{k_0} \theta_i \mathbf{x}_i.$$

**10.3** Give an explanation to justify why in step 4 of the CoSaMP algorithm the value of $t$ is taken to be equal to $2k$.

**10.4** Show that if

$$J(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = \frac{1}{2}||\boldsymbol{y} - X\boldsymbol{\theta}||_2^2 + \lambda||\boldsymbol{\theta}||_1 + \frac{1}{2}d(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}),$$

where

$$d(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) := c||\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}||_2^2 - ||X\boldsymbol{\theta} - X\tilde{\boldsymbol{\theta}}||_2^2.$$

then minimization results to

$$\hat{\boldsymbol{\theta}} = S_{\lambda/c}\left(\frac{1}{c}X^T(\boldsymbol{y} - X\tilde{\boldsymbol{\theta}}) + \tilde{\boldsymbol{\theta}}\right).$$

**10.5** Prove the basic recursion of the parallel coordinate descent algorithm.

*Hint.* Assume that at the $i$th iteration, it is the turn of the $j$th component to be updated, so that the following is minimized:

$$J(\theta_j) = \frac{1}{2}||\boldsymbol{y} - X\boldsymbol{\theta}^{(i-1)} + \theta_j^{(i-1)}\boldsymbol{x_j} - \theta_j\boldsymbol{x}_j||_2^2 + \lambda|\theta_j|.$$

**10.6** Derive the iterative scheme to minimize the weighted $\ell_1$ ball, using a majorization-minimization procedure to minimize $\sum_{i=1}^{l} \ln(|\theta_i| + \epsilon)$, subject to the observations set, $\boldsymbol{y} = X\boldsymbol{\theta}$.
*Hint.* Use the linearization of the logarithmic function to bound it from above, because it is a concave function and its graph is located below its tangent.

**10.7** Show that the weighted $\ell_1$ ball used in SpAPSM is upper bounded by the $\ell_0$ norm of the target vector.

**10.8** Show that the canonical dual frame minimizes the total $\ell_2$ norm of the dual frame, that is,

$$\sum_{i\in\mathcal{I}} \left\|\tilde{\boldsymbol{\psi}}_i\right\|_2^2.$$

*Hint.* Use the result of Problem 9.10.

**10.9** Show that Parseval's tight frames are self-dual.

**10.10** Prove that the bounds $A$, $B$ of a frame coincide with the maximum and minimum eigenvalues of the matrix product $\Psi\Psi^{\mathrm{T}}$.

### *MATLAB Exercises*

**10.11** Construct a multitone signal having samples

$$\theta_n = \sum_{j=1}^{3} a_j \cos\left(\frac{\pi}{2N}(2m_j - 1)n\right), n = 0, \ldots, l-1,$$

where $N = 30$, $l = 2^8$, $\boldsymbol{a} = [0.3, 1, 0.75]^{\mathrm{T}}$, and $\boldsymbol{m} = [4, 10, 30]^{\mathrm{T}}$. (a) Plot this signal in the time and in the frequency domain (use the "fft.m" Matlab function to compute the Fourier transform). (b) Build a sensing matrix $30 \times 2^8$ with entries drawn from a normal distribution, $\mathcal{N}(0, \frac{1}{\sqrt{N}})$, and recover $\boldsymbol{\theta}$ based on these observations by $\ell_1$ minimization using, for example, "solvelasso.m" (see Matlab exercise 9.21). (c) Build a sensing matrix $30 \times 2^8$, where each of its rows contains only a single nonzero component taking the value 1. Moreover, each column has at most one nonzero component. Observe that the multiplication of this sensing matrix with $\boldsymbol{\theta}$ just picks certain components of $\boldsymbol{\theta}$ (those that correspond to the position of the nonzero value of each row of the sampling matrix). Show by solving the corresponding $\ell_1$ minimization task, as in question (b), that $\boldsymbol{\theta}$ can be recovered exactly using such a sparse sensing matrix (containing only 30 nonzero components!). Observe that the unknown $\boldsymbol{\theta}$ is sparse in the frequency domain and give an explanation why the recovering is successful with the specific sparse sensing matrix.

**10.12** Implement the OMP algorithm (see 10.2.1) as well as the CSMP (see 10.2.1) with $t = 2k$. Assume a compressed sensing system using normal distributed sensing matrix. (a) Compare the two algorithms in the case where $\alpha = \frac{N}{l} = 0.2$ for $\beta = \frac{k}{N}$ taking values in the set $\{0.1, 0.2, 0.3, \cdots, 1\}$ (choose yourself a signal and a sensing matrix in order to comply with

the recommendations above). (b) Repeat the same test when $\alpha = 0.8$. Observe that this experiment, if it is performed for many different $\alpha$ values, $0 \leq \alpha \leq 1$, can be used for the estimation of phase transition diagrams such as the one depicted in Figure 10.4. (c) Repeat (a) and (b) with the obtained measurements now contaminated with noise corresponding to 20 dB SNR.

**10.13** Reproduce the MRI reconstruction experiment of Figure 10.9 by running the matlab script "MRIcs.m," which is available from the website of the book.

**10.14** Reproduce the bat echolocation Time-Frequency analysis experiment of Figure 10.18 by running the Matlab script "BATcs.m," which is available from the website of the book.

## REFERENCES


[1] M.G. Amin (Ed.), Compressive Sensing for Urban Radar, CRC Press, 2104.

[2] M.R. Andersen, Sparse inference using approximate message passing, MSc Thesis, Technical University of Denmark, Department of Applied Mathematics and Computing, 2014.

[3] D. Angelosante, J.A. Bazerque, G.B. Giannakis, Online adaptive estimation of sparse signals: where RLS meets the $\ell_1$-norm, IEEE Trans. Signal Proc. 58(7) (2010) 3436-3447.

[4] M. Asif, J. Romberg, Dynamic Updating for $\ell_1$ minimization, IEEE J. Selected Topics Signal Process. 4(2) (2010) 421-434.

[5] M. Asif, J. Romberg, On the LASSO and Dantzig selector equivalence, in: Proceedings of the Conference on Information Sciences and Systems (CISS), Princeton, NJ, March 2010.

[6] F. Bach, Optimization with sparsity-inducing penalties, Foundat. Trends Machine Learn. 4 (2012) 1-106.

[7] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, Structured sparsity through convex optimization, Stat. Sci. 27(4) (2012) 450-468.

[8] S. Bakin, Adaptive regression and model selection in data mining problems, Ph.D. thesis, Australian National University, 1999.

[9] R. Baraniuk, V. Cevher, M. Wakin, Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective, Proc. IEEE 98(6) (2010) 959-971.

[10] R.G. Baraniuk, V. Cevher, M.F. Duarte, C. Hegde, Model-based compressive sensing, IEEE Trans. Informat. Theory 56(4) (2010) 1982-2001.

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

[12] S. Becker, J. Bobin, E.J. Candès, NESTA: A fast and accurate first-order method for sparse recovery, SIAM J. Imaging Sci. 4(1) (2011) 1-39.

[13] A. Belouchrani, M.G. Amin, Blind source separation based on time-frequency signal representations, IEEE Trans. Signal Process. 46 (11) (1998) 2888–2897.

[14] E. van den Berg, M.P. Friedlander, Probing the pareto frontier for the basis pursuit solutions, SIAM J. Sci. Comput. 31(2) (2008) 890-912.

[15] P. Bickel, Y. Ritov, A. Tsybakov, Simultaneous analysis of LASSO and Dantzig selector, Ann. Stat. 37(4) (2009) 1705-1732.

[16] A. Blum, Random projection, margins, kernels and feature selection, Lecture Notes on Computer Science (LNCS), 2006, pp. 52-68.

[17] T. Blumensath, M.E. Davies, Iterative hard thresholding for compressed sensing, Appl. Comput. Harmonic Anal. 27(3) (2009) 265-274.

[18] T. Blumensath, M.E. Davies, Sampling theorems for signals from the union of finite-dimensional linear subspaces, IEEE Trans. Informat. Theory 55(4) (2009) 1872-1882.

[19] T. Blumensath, M.E. Davies, Normalized iterative hard thresholding: guaranteed stability and performance, IEEE Selected Topics Signal Process. 4(2) (2010) 298-309.
[20] B. Boashash, Time Frequency Analysis, Elsevier, 2003.
[21] J.F. Cai, S. Osher, Z. Shen, Split Bregman methods and frame based image restoration, Multiscale Model. Simulat. 8(2)(2009) 337-369.
[22] E. Candès, J. Romberg, T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete Fourier information, IEEE Trans. Informat. Theory 52(2) (2006) 489-509.
[23] E.J. Candès, T. Tao, The Dantzig selector: Statistical estimation when $p$ is much larger than $n$, Ann. Stat. 35(6) (2007) 2313-2351.
[24] E.J. Candès, M.B. Wakin, S.P. Boyd, Enhancing sparsity by reweighted $\ell_1$ minimization, J. Fourier Anal. Appl. 14(5) (2008) 877-905.
[25] E.J. Candès, Y.C. Eldar, D. Needell, P. Randall, Compressed sensing with coherent and redundant dictionaries, Appl. Comput. Harmonic Anal. 31(1) (2011) 59-73.
[26] V. Cevher, P. Indyk, C. Hegde, R.G. Baraniuk, Recovery of clustered sparse signals from compressive measurements, in: International Conference on Sampling Theory and Applications (SAMPTA), Marseille, France, 2009.
[27] V. Cevher, P. Indyk, L. Carin, R.G. Baraniuk, Sparse signal recovery and acquisition with graphical models, IEEE Signal Process. Mag. 27(6) (2010) 92-103.
[28] S. Chen, D.L. Donoho, M. Saunders, Atomic decomposition by basis pursuit, SIAM J. Sci. Comput. 20(1) (1998) 33-61.
[29] S. Chouvardas, K. Slavakis, Y. Kopsinis, S. Theodoridis, A sparsity promoting adaptive algorithm for distributed learning, IEEE Trans. Signal Process. 60(10) (2012) 5412-5425.
[30] S. Chouvardas, Y. Kopsinis, S. Theodoridis, Sparsity-aware distributed learning, in: A. Hero, J. Moura, T. Luo, S. Cui (Eds.), Big Data over Networks, Cambridge University Press, 2014.
[31] P.L. Combettes, V.R. Wajs, Signal recovery by proximal forward-backward splitting, SIAM J. Multiscale Model. Simulat. 4(4) (2005) 1168-1200.
[32] P.L. Combettes, J.-C. Pesquet, Proximal splitting methods in signal processing, in: Fixed-Point Algorithms for Inverse Problems in Science and Engineering, Springer-Verlag, 2011.
[33] W. Dai, O. Milenkovic, Subspace pursuit for compressive sensing signal reconstruction, IEEE Trans. Informat. Theory 55(5) (2009) 2230-2249.
[34] I. Daubechies, M. Defrise, C. De-Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Commun. Pure Appl. Math. 57(11) (2004) 1413-1457.
[35] I. Daubechies, R. DeVore, M. Fornasier, C.S. Güntürk, Iteratively reweighted least squares minimization for sparse recovery, Commun. Pure Appl. Math. 63(1) (2010) 1-38.
[36] I. Daubechies, A. Grossman, Y. Meyer, Painless nonorthogonal expansions, J. Math. Phys. 27 (1986) 1271-1283.
[37] M.A. Davenport, M.B. Wakin, Analysis of orthogonal matching pursuit using the restricted isometry property, IEEE Trans. Informat. Theory 56(9) (2010) 4395-4401.
[38] R.A. DeVore, V.N. Temlyakov, Some remarks on greedy algorithms, Adv. Comput. Math. 5 (1996) 173-187.
[39] P. Di Lorenzo, A.H. Sayed, Sparse distributed learning based on diffusion adaptation, IEEE Trans. Signal Process. 61(6) (2013) 1419-1433.
[40] D.L. Donoho, J. Tanner, Neighborliness of randomly-projected simplifies in high dimensions, in: Proceedings on National Academy of Sciences, 2005, pp. 9446-9451.
[41] D.A. Donoho, A. Maleki, A. Montanari, Message-passing algorithms for compressed sensing, Proc. Natl Acad. Sci. USA 106(45) (2009) 18914-18919.
[42] D.L. Donoho, J. Tanner, Counting the faces of randomly projected hypercubes and orthants, with applications, Discrete Comput. Geomet. 43(3) (2010) 522-541.

[43] D.L. Donoho, J. Tanner, Precise undersampling theorems, Proc. IEEE 98(6) (2010) 913-924.
[44] D.L. Donoho, I.M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, Biometrika 81(3) (1994) 425-455.
[45] D. Donoho, I. Johnstone, G. Kerkyacharian, D. Picard, Wavelet shrinkage: asymptopia? J. R. Stat. Soc. B 57 (1995) 301-337.
[46] J. Duchi, S.S. Shwartz, Y. Singer, T. Chandra, Efficient projections onto the $\ell_1$-ball for learning in high dimensions, in: Proceedings of the International Conference on Machine Leaning (ICML), 2008, pp. 272-279.
[47] R.J. Duffin, A.C. Schaeffer, A class of nonharmonic Fourier series, Trans. Amer. Math. Soc. 72 (1952) 341-366.
[48] B. Efron, T. Hastie, I.M. Johnstone, R. Tibshirani, Least angle regression, Ann. Stat. 32 (2004) 407-499.
[49] M. Elad, J.L. Starck, P. Querre, D.L. Donoho, Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA), Appl. Comput. Harmonic Anal. 19 (2005) 340-358.
[50] M. Elad, B. Matalon, M. Zibulevsky, Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization, Appl. Comput. Harmonic Anal. 23 (2007) 346-367.
[51] M. Elad, P. Milanfar, R. Rubinstein, Analysis versus synthesis in signal priors, Inverse Problems 23 (2007) 947-968.
[52] M. Elad, Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing, Springer, 2010.
[53] Y.C. Eldar, P. Kuppinger, H. Bolcskei, Block-sparse signals: Uncertainty relations and efficient recovery, IEEE Trans. Signal Process. 58(6) (2010) 3042-3054.
[54] Y.C. Eldar, G. Kutyniok, Compressed Sensing: Theory and Applications, Cambridge University Press, 2012.
[55] J. Fan, R. Li, Variable selection via nonconcave penalized likelihood and its oracle properties, J. Amer. Stat. Assoc. 96(456) (2001) 1348-1360.
[56] M.A. Figueiredo, R.D. Nowak, An EM algorithm for wavelet-based image restoration, IEEE Trans. Image Process. 12(8) (2003) 906-916.
[57] P. Flandrin, Time-Frequency/Time-scale Analysis, Academic Press, 1999.
[58] S. Foucart, Hard thresholding pursuit: an algorithm for compressive sensing, SIAM J. Numer. Anal. 49(6) (2011) 2543-2563.
[59] J. Friedman, T. Hastie, R. Tibshirani, A note on the group LASSO and a sparse group LASSO, arXiv:1001.0736v1[math.ST] (2010).
[60] P.J. Garrigues, B. Olshausen, Learning horizontal connections in a sparse coding model of natural images, in: Advances in Neural Information Processing Systems (NIPS), 2008.
[61] A.C. Gilbert, S. Muthukrisnan, M.J. Strauss, Improved time bounds for near-optimal sparse Fourier representation via sampling, in: Proceedings of SPIE (Wavelets XI), San Diego, CA, 2005.
[62] R. Giryes, S. Nam, M. Elad, R. Gribonval, M. Davies, Greedy-like algorithms for the cosparse analysis model, Linear Algebra Appl. 441(0) (2014) 22-60.
[63] T. Goldstein, S. Osher, The split Bregman algorithm for $\ell_1$ regularized problems, SIAM J. Imaging Sci. 2(2) (2009) 323-343.
[64] I.F. Gorodnitsky, B.D. Rao, Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm, IEEE Trans. Signal Process. 45(3) (1997) 600-614.
[65] L. Hageman, D. Young, Applied Iterative Methods. Academic Press, New York, 1981.
[66] T. Hale, W. Yin, Y. Zhang, A fixed-point continuation method for $l_1$ regularized minimization with applications to compressed sensing, Tech. Rep. TR07-07, Department of Computational and Applied Mathematics, Rice University, 2007.

[67] D. Han, D.R. Larson, Frames, Bases and Group Representations, American Mathematical Society, Providence, RI, 2000.
[68] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, second ed., Springer, 2008.
[69] J.C. Hoch, A.S. Stern, D.L. Donoho, I.M. Johnstone, Maximum entropy reconstruction of complex (phase sensitive) spectra, J. Magnet. Resonance 86(2) (1990) 236-246.
[70] P.A. Jansson, Deconvolution: Applications in Spectroscopy. Academic Press, New York, 1984.
[71] R. Jenatton, J.-Y. Audibert, F. Bach, Structured variable selection with sparsity-inducing norms, J. Machine Learn. Res. 12 (2011) 2777-2824.
[72] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, D. Gorinevsky, An interior-point method for large-scale $\ell_1$-regularized least squares, IEEE J. Selected Topics Signal Process. 1(4) (2007) 606-617.
[73] N.G. Kingsbury, T.H. Reeves, Overcomplete image coding using iterative projection-based noise shaping, in: Proceedings IEEE International Conference on Image Processing (ICIP), 2002, pp. 597-600.
[74] K. Knight, W. Fu, Asymptotics for the LASSO-type estimators, Ann. Stat. 28(5) (2000) 1356-1378.
[75] Y. Kopsinis, E. Aboutanios, D.E. Waters, S. McLaughlin, Time-frequency and advanced frequency estimation techniques for the investigation of bat echolocation calls, J. Acoust. Soc. Amer. 127(2) (2010) 1124-1134.
[76] Y. Kopsinis, K. Slavakis, S. Theodoridis, Online sparse system identification and signal reconstruction using projections onto weighted $\ell_1$ balls, IEEE Trans. Signal Process. 59(3) (2011) 936-952.
[77] Y. Kopsinis, K. Slavakis, S. Theodoridis, S. McLaughlin, Reduced complexity online sparse signal reconstruction using projections onto weighted $\ell_1$ balls, in: Digital Signal Processing (DSP), 2011 17th International Conference on, July 2011, pp. 1-8.
[78] Y. Kopsinis, K. Slavakis, S. Theodoridis, S. McLaughlin, Generalized thresholding sparsity-aware algorithm for low complexity online learning, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Kyoto, Japan, March 2012, pp. 3277-3280.
[79] Y. Kopsinis, K. Slavakis, S. Theodoridis, S. McLaughlin, Thresholding-based online algorithms of complexity comparable to sparse LMS methods, in: Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, May 2013, pp. 513-516.
[80] Y. Kopsinis, S. Chouvardas, S. Theodoridis, Sparsity-aware learning in the context of echo cancelation: A set theoretic estimation approach, in: Proceedings of the European Signal Processing Conference (EUSIPCO), Lisbon, Portugal, September 2014.
[81] J. Kovacevic, A. Chebira, Life beyond bases: the advent of frames, IEEE Signal Process. Mag. 24(4) (2007) 86-104.
[82] J. Langford, L. Li, T. Zhang, Sparse online learning via truncated gradient, J. Machine Learn. Res. 10 (2009) 777-801.
[83] Y.M. Lu, M.N. Do, Sampling signals from a union of subspaces, IEEE Signal Process. Mag. 25(2) (2008) 41-47.
[84] Z.Q. Luo, P. Tseng, On the convergence of the coordinate descent method for convex differentiable minimization, J. Optim. Theory Appl. 72(1) (1992) 7-35.
[85] A. Maleki. D.L. Donoho, Optimally tuned iterative reconstruction algorithms for compressed sensing, IEEE J. Selected Topics Signal Process. 4(2) (2010) 330-341.
[86] D.M. Malioutov, M. Cetin, A.S. Willsky, Homotopy continuation for sparse signal representation, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2005, pp. 733-736.
[87] S. Mallat, A Wavelet Tour of Signal Processing: The Sparse Way, third ed., Academic Press, 2008.
[88] S. Mallat, S. Zhang, Matching pursuit in a time-frequency dictionary, IEEE Trans. Signal Process. 41 (1993) 3397-3415.
[89] G. Mateos, J. Bazerque, G. Giannakis, Distributed sparse linear regression, IEEE Trans. Signal Process. 58(10) (2010) 5262-5276.

[90] G. Mileounis, B. Babadi, N. Kalouptsidis, V. Tarokh, An adaptive greedy algorithm with application to nonlinear communications, IEEE Trans. Signal Process. 58(6) (2010) 2998-3007.
[91] A. Mousavi, A. Maleki, R.G. Baraniuk, Parameterless optimal approximate message passing, arXiv:1311.0035v1[cs.IT] 2013.
[92] S. Nam, M. Davies, M. Elad, R. Gribonval, The cosparse analysis model and algorithms, Appl. Comput. Harmonic Anal. 34(1) (2013) 30-56.
[93] D. Needell, J.A. Tropp, COSAMP: iterative signal recovery from incomplete and inaccurate samples, Appl. Comput. Harmonic Anal. 26(3) (2009) 301-321.
[94] D. Needell, R. Ward, Stable image reconstruction using total variation minimization, SIAM J. Imaging Sci., 6(2) (2013) 1035–1058.
[95] Y. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course. Kluwer Academic Publishers, 2004.
[96] Y.E. Nesterov, A method for solving the convex programming problem with convergence rate $O(1/k^2)$, Dokl. Akad. Nauk SSSR 269 (1983) 543-547 (in Russian).
[97] G. Obozinski, B. Taskar, M. Jordan, Multi-task feature selection, Tech. Rep., Department of Statistics, University of California, Berkeley, 2006.
[98] G. Obozinski, B. Taskar, M.I. Jordan, Joint covariate selection and joint subspace selection for multiple classification problems, Stat. Comput. 20(2) (2010) 231-252.
[99] M.R. Osborne, B. Presnell, B.A. Turlach, A new approach to variable selection in least squares problems, IMA J. Numer. Anal. 20 (2000) 389-403.
[100] P.M. Pardalos, N. Kovoor, An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds, Math. Program. 46 (1990) 321-328.
[101] F. Parvaresh, H. Vikalo, S. Misra, B. Hassibi, Recovering Sparse Signals Using Sparse Measurement Matrices in Compressed DNA Microarrays, IEEE J. Selected Topics Signal Process. 2(3) (2008) 275-285.
[102] S. Patterson, Y.C. Eldar, I. Keidar, Distributed compressed sensing for static and time-varying networks, arXiv:1308.6086[cs.IT] 2014.
[103] T. Peleg M. Elad, Performance guarantees of the thresholding algorithm for the cosparse analysis model, IEEE Trans. Informat. Theory 59(3) (2013) 1832-1845.
[104] M.D. Plumbley, Geometry and homotopy for $\ell_1$ sparse representation, in: Proceedings of the International Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS), Rennes, France, 2005.
[105] R.T. Rockafellar, Monotone operators and the proximal point algorithms, SIAM J. Control Optim. 14(5) (1976) 877-898.
[106] R. Rubinstein, R. Peleg, M. Elad, Analysis KSVD: A dictionary-learning algorithm for the analysis sparse model, IEEE Trans. Signal Process. 61(3) (2013) 661-677.
[107] L.I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D Nonlinear Phenomena 60(1-4) (1992) 259-268.
[108] I.W. Selesnick, M.A.T. Figueiredo, Signal restoration with overcomplete wavelet transforms: Comparison of analysis and synthesis priors, in: Proceedings of SPIE, 2009.
[109] K. Slavakis, Y. Kopsinis, S. Theodoridis, S. McLaughlin, Generalized thresholding and online sparsity-aware learning in a union of subspaces, IEEE Trans. Signal Process. 61(15) (2013) 3760-3773.
[110] P. Sprechmann, I. Ramirez, G. Sapiro, Y.C. Eldar, CHiLasso: a collaborative hierarchical sparse modeling framework, IEEE Trans. Signal Process. 59(9) (2011) 4183-4198.
[111] J.L. Starck, E.J. Candès, D.L. Donoho, The curvelet transform for image denoising, IEEE Trans. Image Pocess. 11(6) (2002) 670-684.
[112] J.L. Starck, J. Fadili, F. Murtagh, The undecimated wavelet decomposition and its reconstruction, IEEE Trans. Signal Process. 16(2) (2007) 297-309.

[113] T. Strohmer, Numerical algorithms for discrete Gabor expansions, in: Gabor Analysis and Algorithms: Theory and Applications, Birkhauser, Boston, MA, 1998, pp. 267-294.
[114] V.N. Temlyakov, Nonlinear methods of approximation, Foundat. Comput. Math. 3(1) (2003) 33-107.
[115] J.A. Tropp, Greed is good, IEEE Trans. Informat. Theory 50 (2004) 2231-2242.
[116] Y. Tsaig, Sparse solution of underdetermined linear systems: algorithms and applications, Ph.D. thesis, Stanford University, 2007.
[117] B.A. Turlach, W.N. Venables, S.J. Wright, Simultaneous variable selection, Technometrics 47(3) (2005) 349-363.
[118] S. Wright, R. Nowak, M. Figueiredo, Sparse reconstruction by separable approximation, IEEE Trans. Signal Process. 57(7) (2009) 2479-2493.
[119] M. Yaghoobi, S. Nam, R. Gribonval, M. Davies, Constrained overcomplete analysis operator learning for cosparse signal modelling, IEEE Trans. Signal Process. 61(9) (2013) 2341-2355.
[120] J. Yang, Y. Zhang, W. Yin, A fast alternating direction method for TV $\ell_1$ - $\ell_2$ signal reconstruction from partial Fourier data, IEEE Trans. Selected Topics Signal Process. 4(2) (2010) 288-297.
[121] W. Yin, S. Osher, D. Goldfarb, J. Darbon, Bregman iterative algorithms for $\ell_1$-minimization with applications to compressed sensing, SIAM J. Imaging Sci. 1(1) (2008) 143-168.
[122] M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables, J. R. Stat. Soc. 68(1) (2006) 49-67.
[123] T. Zhang, Sparse Recovery with orthogonal matching pursuit under RIP, IEEE Trans. Informat. Theory 57(9) (2011) 6215-6221.
[124] M. Zibulevsky, M. Elad, L1-L2 optimization in signal processing, IEEE Signal Process. Mag. 27(3) (2010) 76-88.
[125] M. Zibulevsky, Y.Y. Zeevi, Frame analysis of the discrete Gabor scheme, IEEE Trans. Signal Process. 42(4) (1994) 942-945.
[126] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc. B. 67(2) (2005) 301-320.
[127] H. Zou, The adaptive LASSO and its oracle properties, J. Amer. Stat. Assoc. 101 (2006) 1418-1429.
[128] H. Zou, R. Li, One-step sparse estimates in nonconcave penalized likelihood models, Ann. Stat. 36(4) (2008) 1509-1533.