

Chapter 2

Sparse Representations

2.1 Introduction

Sparse representations intend to represent signals with as few as possible significant coefficients. This is important for many applications, like for instance compression.

Consider for example the signal $\cos(5t)$ along 7 min. For computer storage this signal could be saved as a file with hundreds of bytes, if one takes samples and save them. On the other hand, this signal can be represented with a text file such [C 5 7m] with just 4 bytes. Obviously, this last representation is more economical in terms of symbols. In many cases, it is possible to obtain this economy by choosing adequate bases combined with dictionaries.

When using wavelets it is frequently noticed that a great compression rate can be obtained, with almost unnoticeable loss of information. Supposing that the signal comes from a sensor, it would be very convenient to have the sensor yielding already compressed information. This is called '*compressed sensing*'.

Almost immediately, the topics already mentioned lead to inverse problems (related to reconstruction). In the recent times, some interesting methods have been proposed for their solution.

Due to pressing demands from digital devices and networked systems consumers, a lot of interest is being paid to finding sparsest representations. This turns out to be a not-so-standard optimization problem. It also suggests that some space should be devoted, in our book, to mathematical optimization. Since it would be too long for this chapter, an appendix has been included with selected optimization topics, leaving for our present chapter some specific aspects of optimization related to sparse representations.

In the next pages the reader will find numerous references to key publications, which may well serve for a fruitful diving into the various aspects of sparse representations. In the end, what is behind the scene is a new look to sampling [18, 34, 42, 68, 160].

2.2 Sparse Solutions

This section considers a general problem, with a well-known mathematical expression:

$$A \mathbf{x} = \mathbf{b} \quad (2.1)$$

The target is to obtain sparse solutions.

First of all, looking at the $m \times n$ matrix A and the corresponding system of linear equations, there are three cases: $m > n$, $m = n$, and $m < n$.

The case of more equations than unknowns is typical of statistics and regression problems.

On the other hand, the undetermined system is the basic scenario considered in compressive sensing (which will be studied in more detail in the next section).

Usually, in order to get sparse solutions, the question is stated in an optimization framework.

A matrix or a vector is said to be sparse if it has only a few (compared to the total size) non-zero entries. Of course, one needs to formalize this initial concept.

In the case of a vector, its l_0 norm, $\|\mathbf{x}\|_0$, is simply the number of non-zero entries in the vector.

Continuing with vectors, the l_1 norm, $\|\mathbf{x}\|_1$, is the sum of the absolute value of the entries in a vector. The l_2 norm of a vector is its Euclidean length:

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^n |x_i|^2 \right)^{1/2}.$$

From the point of view of sparsity, l_0 is the most appropriate norm, while l_2 says almost nothing. On the other hand, from the point of view of mathematical analysis, finding the sparsest solution is an optimization problem and in this context, the l_2 norm is a well know terrain while l_0 is unusual.

As we shall see, our efforts will concentrate on using the l_1 norm, which is linked to sparsity and is easier for mathematical work.

The main focus of this section is put on methods for solving sparsity optimization problems. The basic literature is [3, 27, 76, 177].

2.2.1 The Central Problem

Consider the problem of finding a sparse $n \times 1$ vector \mathbf{x} such that:

$$A \mathbf{x} = \mathbf{b} \quad (2.2)$$

where the $m \times 1$ vector \mathbf{b} and the $m \times n$ matrix A , are given.

Recall that in the case of regression problems, with $m > n$, this is an overdetermined system that usually has no solution. But one could be interested on establishing a model, based on the following problem:

$$\text{minimize } \|A\mathbf{x} - \mathbf{b}\|_2^2$$

In general this minimization would not enforce sparsity. Some suitable terms should be added, as we shall see.

In the case of undetermined system, with $m < n$, there would be in principle infinite candidate solutions. A typical proposal is to choose the solution with the minimum l_2 norm, so the problem is:

$$\text{minimize } \|\mathbf{x}\|_2^2, \text{ subject to } A\mathbf{x} = \mathbf{b}$$

Using Lagrange multipliers, and taking derivatives, it is possible to analytically get the solution:

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b} \quad (2.3)$$

Again, in general this solution would be not sparse. The problem should be stated with other terms.

But, why it would be interesting to obtain sparse solutions?

If you are trying to establish a model in a regression problem, a sparse solution will reveal the pertinent variables as those with non-zero values. This helps to identify and select the correct variables (in the specialized literature variable selection, or feature selection, is a frequently cited issue).

When dealing with signals, it is sometimes desired to transform a set of measurements \mathbf{b} into a sparse vector \mathbf{x} . Later on, the vector \mathbf{x} could be used to exactly reproduce \mathbf{b} , or, in certain applications it would be sufficient with an approximate recovery. You may remember from past chapters, that this was frequently the case with wavelets, where most information was contained in large entries of \mathbf{x} , while other entries were negligible. Then, it would make sense to consider an approximation of the form:

$$\|A\mathbf{x} - \mathbf{b}\| < \varepsilon \quad (2.4)$$

where some of the entries of \mathbf{x} were set to zero. This is a sparse approximation.

Another case of interest is the design of digital filters with a minimum number of non-zero coefficients, [17].

2.2.1.1 Basic Approaches

Suppose that it is an undetermined system, with infinite candidate solutions. One wants the sparsest solution:

- In formal terms, the problem of sparsest representation can be written as follows:

$$\text{minimize } \|\mathbf{x}\|_0, \text{ subject to } A\mathbf{x} = \mathbf{b}$$

- There is a method, called ‘*Basis Pursuit*’ (BP), which replaces l_0 with l_1 , so now the problem is:

$$\text{minimize } \|\mathbf{x}\|_1, \text{ subject to } A\mathbf{x} = \mathbf{b}$$

Under certain conditions BP will find a sparse solution or even the sparsest one. Another method, called the ‘*Lasso*’, states the following problem:

$$\text{minimize } \|A\mathbf{x} - \mathbf{b}\|_2^2, \text{ subject to } \|\mathbf{x}\|_1 < c$$

This method can be used in the regression context, and for undetermined systems to find sparse approximations. Like BP, under certain conditions it will find the sparsest solution.

2.2.1.2 Some Mathematical Concepts and Facts

Just in the phrases above, the term “norm” has been taken with somewhat excessive license. A function $\|\cdot\|$ is called a *norm* if:

- $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = \mathbf{0}$.
- $\|\lambda\mathbf{x}\| = |\lambda| \|\mathbf{x}\|$ for all λ and \mathbf{x}
- $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for all \mathbf{x} and \mathbf{y} (triangle inequality)

If only (b) and (c) hold the function is called a *semi-norm*.

If only (a) and (b) hold the function is called a *quasi-norm*.

The functions already taken as measures have the form:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} \quad (2.5)$$

If $1 \leq p < \infty$ this function is a norm. However, if $0 < p < 1$ this function is a quasi-norm.

$\|\mathbf{x}\|_0$ is the limit as $p \rightarrow 0$. It is neither a norm or a quasi-norm.

As regards sparsity, a pertinent definition is the following:

A vector \mathbf{x} is *s-sparse* if at most s of its entries are nonzero.

Suppose you have obtained a solution \mathbf{x} for $A\mathbf{x} = \mathbf{b}$ with a few nonzero entries. Then you may ask yourself if it is the sparsest solution. Here, it becomes relevant to consider the following definition:

The ‘*spark*’ of a matrix A is the smallest number of columns of A that are linearly-dependent.

According with this definition, the vectors in the null space of the matrix (i.e. $A\mathbf{x} = 0$) must satisfy: $\|\mathbf{x}\|_0 \geq \text{spark}(A)$.

Now, there is a theorem that establishes the following: if $A\mathbf{x} = \mathbf{b}$ has a solution \mathbf{x} obeying $\|\mathbf{x}\|_0 < \text{spark}(A)/2$, this solution is necessarily the sparsest possible [76].

2.2.2 Norms and Sparsity

Specialists on sparse representations do speak of norms that promote sparsity. This is an interesting aspect, which can be illustrated with some plots.

Suppose that the matrix A is a 1×2 matrix. Then, the solutions of $A\mathbf{x} = \mathbf{b}$ form a line L on the 2D plane.

Consider the following ball of radius r :

$$B_p(r) = \{\mathbf{x} \mid \|\mathbf{x}\|_p \leq r\} \quad (2.6)$$

Then the solution that minimizes the norm l_p is obtained at the intersection of the ball and the line L .

Figure 2.1 shows three cases, corresponding to the intersection of the line and the balls $B_{1/2}$, B_1 , and B_2 . Notice that the intersection of the line L with the vertical axis gives a sparsest solution (only one nonzero entry).

The left hand plot, (a), in Fig. 2.1 makes clear that quasi-norms with $0 < p < 1$ promote sparsity. The central plot, (b), shows that usually l_1 promote sparsity (unless L was parallel to a border of B_1). Finally, the right hand plot shows that l_2 do not promote sparsity.

This simple example can be generalized to problems with more dimensions.

There are other functions that promote sparsity, like for instance $1 - \exp(-|\mathbf{x}|)$, or $\log(1 + |\mathbf{x}|)$, or $|\mathbf{x}|/(1 + |\mathbf{x}|)$ [76].

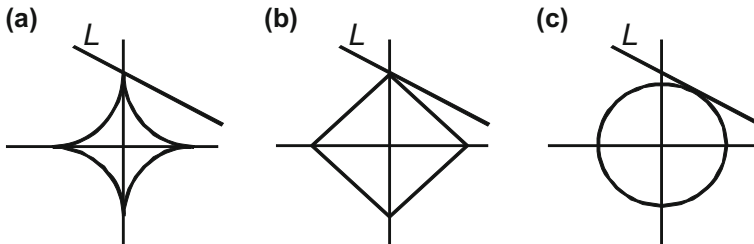


Fig. 2.1 The line L and, **a** the ball $B_{1/2}$, **b** the ball B_1 , **c** the ball B_2

2.2.3 Solving Sparsity Optimization Problems

There are several ways for solving the sparsity optimization problems. The BP problem can be tackled with linear programming. The Lasso problem can be treated with quadratic programming, but there is a better approach. In certain scenarios it would be more appropriate to use first order methods.

Let us enter into specific details.

2.2.3.1 Basis Pursuit

The basic basis pursuit problem,

$$\text{minimize } \|\mathbf{x}\|_1, \text{ subject to } A\mathbf{x} = \mathbf{b}$$

can be expressed as a Linear Programming (LP) problem (see appendix on optimization), as it is detailed in [53]. The main steps are the following:

- First, notice that $\|\mathbf{x}\|_1 = |x_1| + |x_2| + \dots + |x_n|$. One could use a set of auxiliary variables t_1, t_2, \dots, t_n to obtain an equivalent problem:

$$\text{minimize } (t_1 + t_2 + \dots + t_n),$$

$$\text{subject to } |x_1| \leq t_1; |x_2| \leq t_2; \dots; |x_n| \leq t_n \text{ and } A\mathbf{x} = \mathbf{b}$$

- Next, each inequality can be expressed as two inequalities; for example, $|x_1| \leq t_1$ is equivalent to: $x_1 \leq t_1, x_1 \geq -t_1$. Then, the problem can be expressed in the following compact form:

$$\begin{aligned} &\text{minimize } (t_1 + t_2 + \dots + t_n), \\ &\text{subject to:} \\ &I\mathbf{x} - I\mathbf{t} \leq 0 \\ &I\mathbf{x} + I\mathbf{t} \geq 0 \\ &A\mathbf{x} = \mathbf{b} \end{aligned} \tag{2.7}$$

This last formulation corresponds to a LP format.

Another way for obtaining an LP expression is to use a $+/-$ split, so each component x_i is represented as follows:

$$x_i = p_i - n_i \tag{2.8}$$

with:

$$p_i = \begin{cases} x_i & \text{if } x_i > 0 \\ 0 & \text{else} \end{cases} \quad \text{and } n_i = \begin{cases} -x_i & \text{if } x_i < 0 \\ 0 & \text{else} \end{cases}$$

Then: $\|\mathbf{x}\|_1 = \mathbf{p} + \mathbf{n}$

With these changes, the problem now would be:

$$\text{minimize } \mathbf{p} + \mathbf{n}, \text{ subject to } A(\mathbf{p} - \mathbf{n}) = \mathbf{b}; \mathbf{p}, \mathbf{n} \geq 0$$

which is LP.

The reader would find in [53] programs in MATLAB for basis pursuit, using the *linprog()* function.

Of course, other methods, like for instance interior-point methods, can also be applied to solve this problem.

2.2.3.2 Lasso

The Lasso approach can be expressed in two equivalent forms:

- penalized form:

$$\text{minimize } (\|\mathbf{Ax} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_1) \quad (2.9)$$

- constraint form:

$$\text{minimize } \|\mathbf{Ax} - \mathbf{b}\|_2^2, \text{ subject to } \|\mathbf{x}\|_1 < c$$

It sometimes happens that different types of problems lead to similar equations. In the case of Lasso, it appears at least in two contexts: statistics and signal processing.

In statistics, one finds that certain regression problems [156] can be treated with a Lasso approach. For example, the basic regression method (least squares) consists in minimizing $\|\mathbf{Ax} - \mathbf{b}\|_2^2$ (where A and \mathbf{b} are given); however, it has been noticed in practical applications, that this approach may have numerical difficulties or unsatisfactory results (like for instance, overfitting). By adding penalty terms, both regularization and suitable fitting could be obtained.

‘ridge regularization’ substitutes the minimization of $\|\mathbf{Ax} - \mathbf{b}\|_2^2$ with the minimization of $(\|\mathbf{Ax} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_2^2)$. A l_2 penalty is added.

The ridge regularization is obviously related to the Tikhonov regularization. Eventually, the ridge regularization was introduced by a most cited article [101], in 1970; while the work of Tikhonov, which started in the 1940s, was widely known after the publication in 1977 of his first book in English.

One of the good effects of ridge regularization is a certain shrinking of the solution components. According with [166], higher values of λ force these components to be more similar to each other.

In 1996, Tibshirani [176] introduced the method ‘*Least Absolute Shrinkage and Selection Operator*’ (LASSO). A l_1 penalty is added (2.9).

There are analysis of large data and data mining scenarios where it is important, as said before, to assess which variables are more important, more influential. The l_1 penalty has the advantage of promoting sparsity. Contrary to the l_2 penalty, the l_1 penalty leads to solutions with many zero components, and this helps to determine the relevant variables. This is a main reason of Lasso popularity.

As we shall see in the next sections, the l_1 penalty (and the l_1 norm) plays an important role both in compressed sensing and image processing. In other contexts [156], l_1 is used for robust statistics in order to obtain results insensitive to outliers [187], and for maximum likelihood estimates in case of Laplacian residuals.

It is also shown, in an appendix of [187], that the two formulations:

$$\min_x \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 + c_1 \|\mathbf{x}\|_1 \quad \text{and} \quad \min_x \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + c_2 \|\mathbf{x}\|_1$$

are equivalent.

With respect to analytical solutions, recall that the least squares problem in underdetermined systems has the following solution:

$$\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \quad (2.10)$$

The solution for the ridge regularization would be:

$$\mathbf{x} = (\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I})^{-1} \mathbf{A}^T \mathbf{b} \quad (2.11)$$

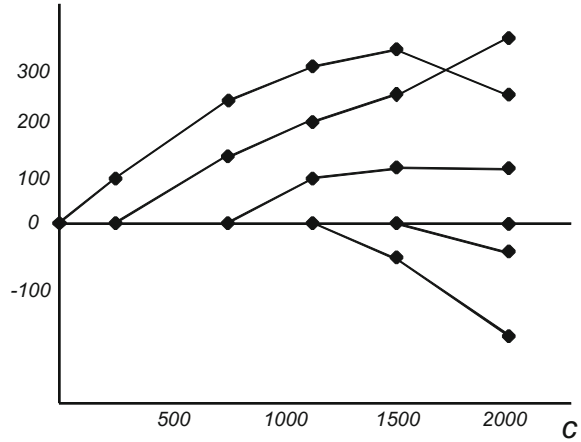
In the case of Lasso, finding the solution is more involved, and many methods have been proposed. Indeed, it can be treated as a Quadratic Programming (QP) problem (see appendix on optimization). In fact, Tibshirani proposed two ways of stating the Lasso as QP [166, 176], much in the line described above for expressing the basis pursuit as LP. Concretely, one of the proposed ways uses a $+/-$ split, and so the Lasso is expressed as follows:

$$\text{minimize } (\|\mathbf{A}(\mathbf{p} - \mathbf{n}) - \mathbf{b}\| + \lambda(\mathbf{p} + \mathbf{n})), \quad \text{subject to } \mathbf{p}, \mathbf{n} \geq 0$$

which is a QP problem.

Nowadays there are better algorithms to solve Lasso problems. A popular one is *LARS* (‘*Least Order Regression*’), which was introduced in 2004 [75]. It is a stepwise iterative algorithm that starts with $c = 0$ (constraint form), and detects the most influential variables as c increases. Figure 2.2 shows an example of the paths of the solutions when c increases: for small values of c there is only one non-zero component of \mathbf{x} , when c reaches a certain value a second non-zero component rises,

Fig. 2.2 An example of the solution paths obtained with LARS



then for larger c a third non-zero component rises, and so on. Notice that components with negative values can also appear. The next paragraphs intend now to introduce LARS in more detail.

Let us follow the scheme of [75] in order to explain LARS. A first point is that the LARS algorithm relates to the ‘Forward Selection’ (FS) algorithm described in [185] (1980). The FS algorithm applies the following steps:

- Set all variables to zero
- Find the column \mathbf{A}_j of matrix \mathbf{A} most correlated with \mathbf{b} . Based on \mathbf{A}_j perform a simple linear regression to obtain x_j . Then, a residue is obtained:

$$\mathbf{r}_j = \mathbf{A}_j x_j - \mathbf{b} \quad (2.12)$$

- Find the column \mathbf{A}_k most correlated to \mathbf{r}_j and perform a linear regression to obtain x_k , obtain a new residue:

$$\mathbf{r}_k = \mathbf{A}_k x_k - \mathbf{r}_j \quad (2.13)$$

- Repeat until a stop criterion is reached (for instance in terms of the statistical characteristics of the residual).

(The correlation of a column \mathbf{A}_i with a residual \mathbf{r}_l can be computed as $|\mathbf{A}_i^T \mathbf{r}_l|$).

Once a final solution \mathbf{x} was built, a model $\boldsymbol{\mu} = \mathbf{A} \mathbf{x}$ is obtained that can be used for several purposes.

Along the process, a series of components, $x_i, x_j, x_k, x_l, \dots$, were selected. At the time of implementing this kind of algorithm it is usual to maintain a set, called *the active set*, which includes the indexes of the selected components:

$$J = \{i, j, k, l, \dots\} = \{j_i\} \quad (2.14)$$

Likewise, one can assemble the selected columns \mathbf{A}_i into a matrix:

$$\mathbf{A}_J = \{\mathbf{A}_i, \mathbf{A}_j, \mathbf{A}_k, \mathbf{A}_l, \dots\} \quad (2.15)$$

These notations will be useful when describing LARS.

The FS algorithm is not in general recommended; in [75] words: it is an aggressive technique that can be too much greedy. However, there is a cautious version, called ‘*Forward Stagewise*’, which can be more acceptable. It also starts with all variables set to zero and with $\mathbf{r} = \mathbf{b}$, and then it repeats many times the following:

-
- Find the \mathbf{A}_j most correlated with \mathbf{r}
 - $\delta = \gamma \operatorname{sign}(\mathbf{A}_j^T \mathbf{r})$
 - $x_j \leftarrow x_j + \delta$
 - $\mathbf{r} \leftarrow \mathbf{r} - \delta \mathbf{A}_j$

where the learning rate γ is small.

In [75] an example was studied with 10 columns of data from 442 diabetes patients. Using forward stagewise, a solution path figure (like Fig. 2.2) was obtained, after 6000 stagewise steps. The article remarks that the cautious nature of the algorithm, with many small steps, implies considerable computational effort. One of the benefits of LARS is that it would require much less steps.

Like the others, the LARS algorithm begins with $\mathbf{x} = 0$, $\mathbf{r} = \mathbf{b}$, and a model $\mu_0 = 0$. It finds the \mathbf{A}_j most correlated with \mathbf{r} , and then it augments the model as follows:

$$\mu_1 = \mu_0 + \gamma_1 \mathbf{A}_j \quad (2.16)$$

(the component x_j is selected, with the value γ_1)

The forward stagewise algorithm would use a small γ_1 ; while the FS would use a large enough γ_1 to make μ_1 equal to the projection of \mathbf{b} into \mathbf{A}_j . LARS takes an intermediate value.

The idea in LARS is to take the largest step possible in the direction of \mathbf{A}_j until some other \mathbf{A}_k has as much correlation with the residual $\mathbf{r}_1 = \mathbf{b} - \mu_1$. The corresponding geometry is that the residual would bisect the angle between \mathbf{A}_j and \mathbf{A}_k (the least angle direction). Let \mathbf{u}_1 be the unit vector along \mathbf{r}_1 , the next step would be:

$$\mu_2 = \mu_1 + \gamma_2 \mathbf{u}_1 \quad (2.17)$$

This step will be the largest possible until a third \mathbf{A}_l is found such as \mathbf{A}_j , \mathbf{A}_k and \mathbf{A}_l had the same correlation with $\mathbf{r}_2 = \mathbf{b} - \mu_2$. The next step will proceed equiangular between the three already selected A columns. And so on.

According with [75] the equiangular direction can be computed as follows:

- Denote:

$$G = A_J^T A_J; \quad Q = (1_J^T G^{-1} 1_J)^{-1/2} \quad (2.18)$$

- Then:

$$\mathbf{u}_J = A_J \boldsymbol{\omega}_J, \text{ where } \boldsymbol{\omega}_J = Q G^{-1} 1_J$$

(the vector 1_J is a vector of 1's)

The equiangular direction \mathbf{u}_J has the following characteristics:

$$A_J^T \mathbf{u}_J = Q 1_J, \text{ and } \|\mathbf{u}_J\|_2^2 = 1$$

In the algorithm steps, the step size is computed as follows:

- Denote:

$$\mathbf{a} = A^T \mathbf{u}_J$$

$$\bar{c} = A^T (\mathbf{b} - \boldsymbol{\mu}_J) \text{ (current correlation)}$$

$$C = \max_i \{|c_i|\} \quad (2.19)$$

- Then:

$$\gamma = \min_i^+ \left\{ \frac{C - c_i}{Q - a_i}, \frac{C + c_i}{Q + a_i} \right\} \quad (2.20)$$

(\min^+ means that the minimum is taken over only positive components).

In order to use LARS for the Lasso problem it is necessary to add a slight modification: if an active component passes through zero, it must be excluded from the active set. It may happen that later on this variable would enter again into the active set.

When that modification is included, the LARS algorithm is able to find all the solutions of the Lasso problem.

A simple, direct implementation of the LARS algorithm is offered in the Program 2.1, which deals with a table of diabetes data. The reader can edit the *lasso_f* flag (see the code) in order to run LARS, or LASSO. The program is based on the information and listings provided by [1], which has a link to the diabetes data (Stanford University). Figure 2.3 shows the solution paths obtained with LARS.

Figure 2.4 shows the solution paths obtained with LASSO, which have noticeable differences compared to the LARS solution paths.

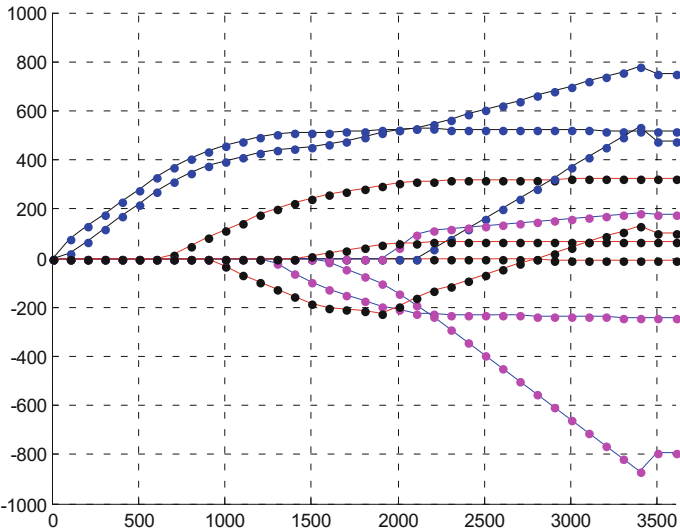


Fig. 2.3 Solution paths using LARS for diabetes set

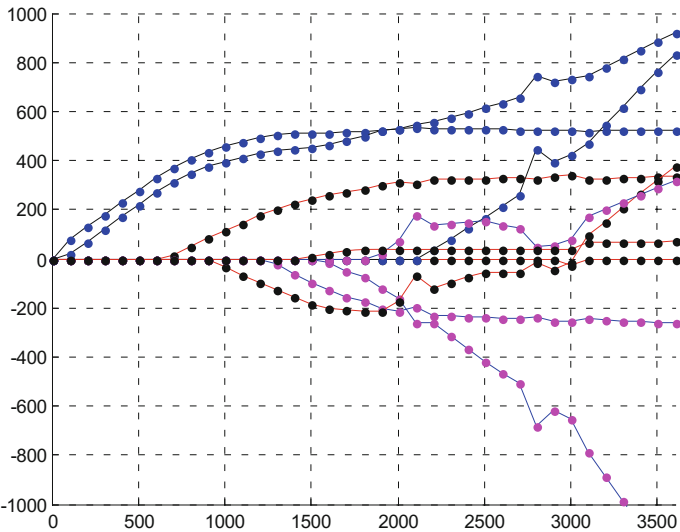


Fig. 2.4 Solution paths using LASSO for diabetes set

Program 2.1 LARS/LASSO example

```

% LARS/LASSO example
% diabetes data
% data matrix (the last column is the response)
data=load('diabetes_table.txt');
% Data preprocessing:
% zero mean, unit L2 norm, response with no bias
[ll,cc]=size(data); A=zeros(ll,cc-1); b=zeros(ll,1);
for j=1:cc-1,
    A(:,j)=data(:,j)-mean(data(:,j)); %mean removal
    aux=sqrt(A(:,j)'*A(:,j)); A(:,j)=A(:,j)/aux; %unit L2-norm
end;
b=data(:,cc)-mean(data(:,cc));
% Initialization of variables
hbeta=[];
maxnav=min(ll,cc-1); %maximum number of active variables
gamma_av=0;
%valid sign flag:
vls=1; %valid
% LARS/LASSO: -----
lasso_f=0; % flag (0- LARS; 1-LASSO) (Edit here)
ni=1; nitmax=3600;
for t=0:100:nitmax,
    %Initialization of variables
    beta=zeros(cc-1,1);
    mu=zeros(ll,1);
    c=[]; maxc=0; maxc_ix=0;
    J=[]; %indices of the active set
    signJ=[];
    %active set variables (av):
    nav=0; %counter of active variables
    gamma_av=0;
    %valid sign flag:
    vls=1; %valid
    inz=1; %initialize flag
    bkloop=0; %for loop breaking
    norml=0;
    oldnorml=0;
    while nav<maxnav,
        c=A'*(b-mu); %current corr
        maxc=max(abs(c));
        %initialization
        if inz==1, [aux, maxc_ix]=max(abs(c)); inz=0; end;
        if vls==1, J=[J, maxc_ix]; nav=nav+1; end; %add index to J
        signJ=sign(c(J));
        complJ=setdiff(1:maxnav,J); %complement of J
        LcomplJ=length(complJ);
        Aj=A(:,J);

```

```

for k=1:length(J),
    Aj(:,k)=signJ(k)*Aj(:,k);
end;
G=Aj'*Aj;
Ij=ones(length(J),1);
iG=inv(G);
Q=1/sqrt(Ij'*iG*Ij);
wj=Q*iG*Ij;
uj=Aj*wj;
a=A'*uj;
if nav==maxnav, gamma_av =maxc/Q; %unusual last stage
else
    gamma=zeros(LcomplJ,2); %min of two terms
    for k=1:LcomplJ,
        n=complJ(k);
        gamma(n,:)=[((maxc-c(n))/(Q-a(n))),...
            ((maxc+c(n))/(Q+a(n)))];
    end;
    %remove complex elements, reset to Inf
    [pi,pj]=find(0~=imag(gamma));
    for nn=1:length(pi), gamma(pi(nn),pj(nn))=Inf; end;
    % find minimum
    gamma(gamma<=0) = Inf;
    mm=min(min(gamma)); gamma_av=mm;
    [maxc_ix,aux]=find(gamma==mm);
end;
%update coeff estimate:
baux(J)=beta(J)+gamma_av*diag(signJ)*wj;
bkloop=0;
Jold=J;
% The LASSO option -----
if lasso_f==1,
    vls=1; %valid sign
    dh=diag(signJ'*wj);
    gamma_c = -beta(J)./dh;
    %remove complex elements, reset to Inf
    [pi,pj]=find(0~=imag(gamma_c));
    for nn=1:length(pi), gamma_c(pi(nn),pj(nn))=Inf; end;
    % find minimum
    gamma_c(gamma_c <= 0) = Inf;
    mm=min(min(gamma_c)); gamma_w=mm;
    [gamma_w_ix,aux]=find(gamma_c==mm);
    %Lasso modification:
    if isnan(gamma_w), gamma_w=Inf; end;
    if gamma_w < gamma_av,
        gamma_av = gamma_w;
        baux(J)=beta(J)+gamma_av*diag(signJ)*wj;
        J(gamma_w_ix)=[]; %delete zero-crossing element
        nav=nav-1;
    end;
end;

```

```

        vls=0;
    end;
end;
% -----
norm1=norm(baux(Jold),1);
if oldnorm1<=t && norm1>=t,
    baux(Jold)=beta(Jold)+Q*(t-oldnorm1)*diag(signJ)*wj;
    bkloop=1; %for loop break
end;
oldnorm1=norm1;
mu=mu+ gamma_av*uj;
beta(Jold)=baux(Jold);
if bkloop==1, break; end %while end
end;
beta(ni,:)=beta';
ni=ni+1;
end;
% display -----
[bm,bn]=size(hbeta);
aux=[0:100:nitmax];
q=0; %color switch
figure(1)
hold on; q=0;
for np=1:bn,
    if q==0,
        plot(aux,hbeta(:,np),'r',aux,hbeta(:,np),'k. ');
        axis([0 nitmax -1000 1000]);
    if lasso_f==0,
        title('LARS, diabetes set');
    else
        title('LASSO, diabetes set');
    end;
end;
end;
if q==1,
    plot(aux,hbeta(:,np),'b',aux,hbeta(:,np),'m. ');
    axis([0 nitmax -1000 1000]); end
if q==2,
    plot(aux,hbeta(:,np),'k',aux,hbeta(:,np),'b. ');
    axis([0 nitmax -1000 1000]); end
    q=q+1; if q==3, q=0; end;
end;
grid;

```

It is interesting to look at a special case: that A is orthonormal and so $A^T A = I$. In this case the ordinary least square (OLS) regression has the following solution, that we will denote as \hat{x} :

$$\hat{x} = A^T \mathbf{b} \quad (2.21)$$

- The solution for the ridge regression would be:

$$\mathbf{x} = \frac{1}{1 + \lambda} \hat{x} \quad (2.22)$$

- And the solution for the Lasso would be:

$$x_i = S_{\lambda/2}(\hat{x}_i), \quad i = 1, 2, \dots, n \quad (2.23)$$

In this last expression the soft-thresholding operator $S_{\lambda/2}$ is used, its values are:

$$S_{\lambda/2}(x) = \begin{cases} x + (\lambda/2), & \text{if } x \leq -\lambda/2 \\ 0, & \text{if } |x| < \lambda/2 \\ x - (\lambda/2), & \text{if } x \geq \lambda/2 \end{cases} \quad (2.24)$$

The action of this operator is expressed in Fig. 2.5. This operator appears in other optimization contexts.

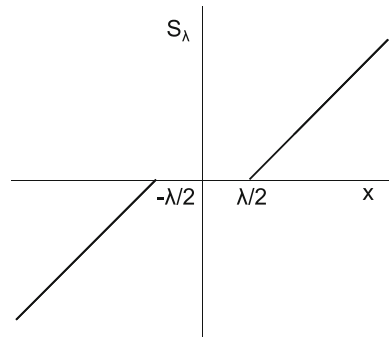
In two most cited papers, [201, 206], the oracle properties of Lasso were studied. These articles include relevant references to previous work. The desired oracle properties are that the right model is identified, and that it has optimal estimation rate. It may happen that, for a certain value of λ (the regularization parameter), a good estimation error was achieved but with an incorrect (inconsistent) model.

One of the sections of [206] shows that the Lasso variable selection could be inconsistent. In coincidence with this, [201] writes that, in general, if an irrelevant \mathbf{A}_k is highly correlated with columns \mathbf{A}_i of the true model, Lasso may not be able to distinguish it from the true model columns. [201] introduces an “*irrepresentable condition*”, which resembles the following constraint:

$$|(A_J^T A_J)^{-1} A_J^T A_I| < 1 \quad (2.25)$$

where J is the active set and I the inactive set.

Fig. 2.5 Soft-thresholding operator



This last expression implies that relevant variables were not too highly correlated with nuisance variables.

The irrepressible condition of [201] is almost necessary and sufficient for Lasso to select the true model. The lesson for experiment designers is to avoid spurious variables that may incur in high correlation with relevant variables.

In order to get better oracle properties, the ‘*adaptive Lasso*’ was introduced in [206]. The original constrained Lasso is transformed to:

$$\text{minimize} \left(\|A\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \sum_{i=1}^n \frac{|x_i|}{|\hat{x}_i|^\nu} \right)$$

(where $\nu > 0$)

Notice that a weighting has been introduced in the penalty term. This weighting is data-dependent. It is shown in [206] that consistency in variable selection can be achieved.

By the way, the reader is referred to [206] for an interesting comparison of the adaptive Lasso and the nonnegative ‘*garotte*’. The *garotte* finds a set of scaling factors c_j for the following problem:

$$\text{minimize} \left(\left\| \mathbf{b} - \sum_{j=1}^n \mathbf{A}_j \hat{x}_j c_j \right\|_2^2 + \lambda \sum_{j=1}^n c_j \right)$$

where $c_j \geq 0$).

In some problems it is known that the variables belong to certain groups, and it is opportune to shrink the members of a group together. The ‘*group Lasso*’ has been introduced in [195]. Suppose the n variables are divided into L groups. Denote A_k the matrix obtained from A by selecting the columns corresponding to the k -th group. Likewise, \mathbf{x}_k is the vector with components being the variables of the k -th group. The group Lasso set the problem as follows:

$$\text{minimize} \left(\|A\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \sum_{k=1}^L \sqrt{n_k} \|\mathbf{x}_k\|_2 \right)$$

where n_k is the number of member of the k -th group.

The penalty term can be viewed as an intermediate between l_1 and l_2 .

A remark from [84] is that the group Lasso does not yield sparsity within a group. A more general penalty was then proposed [84] that promotes sparsity both at the group and individual levels. It is called ‘*sparse group Lasso*’:

$$\text{minimize} \left(\|A\mathbf{x} - \mathbf{b}\|_2^2 + \lambda_1 \sum_{k=1}^L \sqrt{n_k} \|\mathbf{x}_k\|_2 + \lambda_2 \|\mathbf{x}\|_1 \right)$$

In another most cited paper, [207] introduced the ‘*elastic net*’, which is a two-step method. For the first step, the paper introduces the ‘*naïve elastic net*’ as the following problem:

$$\text{minimize } (\|A\mathbf{x} - \mathbf{b}\|_2^2 + \lambda_1 \|\mathbf{x}\|_1 + \lambda_2 \|\mathbf{x}\|_2^2)$$

Denote as \tilde{x} the estimate obtained by the first step. The second step is just the following correction:

$$\mathbf{x} = \frac{\tilde{x}}{\sqrt{1 + \lambda_2}} \quad (2.26)$$

The elastic net can be regarded as compromise between ridge regression and Lasso. Because of the quadratic term, the elastic net penalty is strictly convex.

As it was shown in Fig. 2.2, the solutions of the Lasso problem follow polygonal paths. This fact was observed in [141], in the year 2000. Based on this fact, the paper proposed a homotopy algorithm to solve the problem. It uses an active set, which is updated through the addition and removal of variables. The LARS algorithm, introduced four years later, is an approximation to the homotopy algorithm (see the brief historical account given in [71]). Later on, year 2006, it was shown, [72], that the LARS/homotopy methods, first proposed for statistical model selection, could be useful for sparsity research.

With relative frequency new variants of the LARS and homotopy algorithms appear in the literature, and different aspects of them are investigated, like for instance [123] on the number of linear segments of the solution paths. See [107] for some variants of group Lasso in the context of structured sparsity.

2.2.3.3 First Order Methods

Although most of the sparse approximation problems can be tackled with LP or QP algorithms, there are applications in which, due to the size or the characteristics of the problem, computational difficulties appear [20]. Some of these difficulties are detailed in [81]. Therefore, other solution alternatives have been explored, including the first order methods described in the appendix on optimization.

An example of gradient based method is the ‘*gradient projection sparse representation*’ (GPSR) proposed in [81]. Positive and negative variables are separated, and the problem:

$$\text{minimize } \left(\frac{1}{2} \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1 \right)$$

is rewritten as:

$$\text{minimize } Q(\mathbf{x}) = \left(\frac{1}{2} \|\mathbf{b} - [A, -A][\mathbf{x}_+; \mathbf{x}_-]\|_2^2 + \lambda 1^T (\mathbf{x}_+ + \mathbf{x}_-) \right)$$

with: $\mathbf{x}_+ \geq 0$; $\mathbf{x}_- \geq 0$

Now, denote:

$$\mathbf{z} = [\mathbf{x}_+; \mathbf{x}_-], \quad \mathbf{c} = \lambda \mathbf{1} + [-A^T \mathbf{b}; A^T \mathbf{b}], \quad \text{and:}$$

$$B = \begin{pmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{pmatrix} \quad (2.27)$$

Then:

$$Q(\mathbf{z}) = \frac{1}{2} \mathbf{z}^T B \mathbf{z} + \mathbf{c}^T \mathbf{z} \quad (2.28)$$

and the gradient: $\nabla_{\mathbf{z}} Q(\mathbf{z}) = B \mathbf{z} + \mathbf{c}$

Therefore, a steepest descent algorithm can be enounced:

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \alpha_k \nabla_{\mathbf{z}} Q(\mathbf{z}_k) \quad (2.29)$$

It was shown in [81] that the computation of the matrix B can be done more economically than its size suggests, and that the gradient of Q requires one multiplication each by A and A^T . No inversion of matrices is needed.

There is an important family of methods in connection with the ‘*iterative shrinkage/thresholding*’ (IST) algorithm. Initially, IST was introduced as an EM algorithm for image deconvolution [136]. According with [81], it can be used for Lasso problems and only requires matrix-vector multiplications involving A and A^T . Let us describe the idea; suppose you have the following composite problem:

$$\text{minimize } f(\mathbf{x}) + \lambda g(\mathbf{x})$$

Consider the next iterative approximation of the solution:

$$\begin{aligned} \mathbf{x}_{k+1} &= \arg \min_{\mathbf{x}} \{f(\mathbf{x}_k) + (\mathbf{x} - \mathbf{x}_k)^T \nabla f(\mathbf{x}_k) + \\ &\quad + \frac{1}{2} \|\mathbf{x} - \mathbf{x}_k\|_2^2 \nabla^2 f(\mathbf{x}_k) + \lambda g(\mathbf{x})\} \approx \\ &\approx \arg \min_{\mathbf{x}} \{(\mathbf{x} - \mathbf{x}_k)^T \nabla f(\mathbf{x}_k) + \frac{\alpha_k}{2} \|\mathbf{x} - \mathbf{x}_k\|_2^2 + \lambda g(\mathbf{x})\} \end{aligned} \quad (2.30)$$

(the Hessian $\nabla^2 f(\mathbf{x}_k)$ is approximated by a diagonal matrix $\alpha_k I$)

Denote:

$$\mathbf{u}_k = \mathbf{x}_k - \frac{1}{\alpha_k} \nabla f(\mathbf{x}_k) \quad (2.31)$$

Then, the iterative procedure can be written as:

$$\mathbf{x}_{k+1} = \arg \min_{\mathbf{x}} \left\{ \frac{1}{2} \|\mathbf{x} - \mathbf{u}_k\|_2^2 + \frac{\lambda}{\alpha_k} g(\mathbf{x}) \right\} \quad (2.32)$$

(the \approx was replaced by $=$)

In the case of $g(\mathbf{x})$ being $\|\mathbf{x}\|_1$, which is separable, then there is a component-wise closed-form solution:

$$(\mathbf{x}_{k+1})_i = \arg \min_x \left\{ \frac{1}{2} ((\mathbf{x})_i - (\mathbf{u}_k)_i)^2 + \frac{\lambda}{\alpha_k} |(\mathbf{x})_i| \right\} = S_T((\mathbf{u}_k)_i) \quad (2.33)$$

where we used $(\mathbf{x})_i$ to denote the i -th component of \mathbf{x} . The last, right-hand term includes a soft-thresholding operator $S_T(y)$:

$$S_T(y) = \begin{cases} y + T, & \text{if } y \leq -T \\ 0, & \text{if } |y| < T \\ y - T, & \text{if } y \geq T \end{cases} \quad (2.34)$$

with: $T = \frac{\lambda}{\alpha_k}$

Therefore, a simple iterative shrinkage/thresholding algorithm, named ISTA, has been devised [20]. There are two parameters to specify, λ and α_k . The research has proposed a number of strategies for this specification.

Another, equivalent form of expressing the soft-thresholding operator is:

$$S_T(y) = \text{sgn}(y) \cdot (|y| - T)_+ \quad (2.35)$$

In case the composite problem was of Lasso type:

$$\text{minimize} \left(\frac{1}{2} \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1 \right)$$

then:

$$\nabla f(\mathbf{x}_k) = -A^T(\mathbf{b} - A\mathbf{x}_k)$$

And so, the iterative algorithm would be:

$$\mathbf{x}_{k+1} = S_T \left(\mathbf{x}_k + \frac{1}{\alpha_k} A^T(\mathbf{b} - A\mathbf{x}_k) \right) \quad (2.36)$$

(the operator should be component-wise applied)

The algorithm is clearly related with the '*Landweber algorithm*', introduced in the 1950s to solve ill-posed linear inverse problems and that has the following general expression:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda A^T M(\mathbf{b} - A\mathbf{x}_k) \quad (2.37)$$

(where M is a positive symmetric matrix and λ is a parameter)

The (2.36) algorithm is called the ‘*thresholded Landweber iteration*’ (TLI) in an interesting paper of Daubechies, et al. [63]. This paper proposes an improvement of the TLI algorithm using a projection on l_1 -balls (resulting in a variable thresholding).

Notice that TLI is a particular case of ISTA for the Lasso problem. However, it happens that some papers use the name ISTA to refer just to TLI, so loosing generality.

An illustrative example is the case of a sparse signal that we observe through a filter and that is further contaminated with noise. Program 2.2 attacks this case, applying ISTA for the recovery of the sparse signal. One of the outputs of the program is Fig. 2.6, which shows the original signal, the observed signal (contaminated with noise), and the signal being recovered by ISTA. The program is based on [168].

Another output of Program 2.2 is Fig. 2.7, which shows the evolution of the objective function along the minimization process.

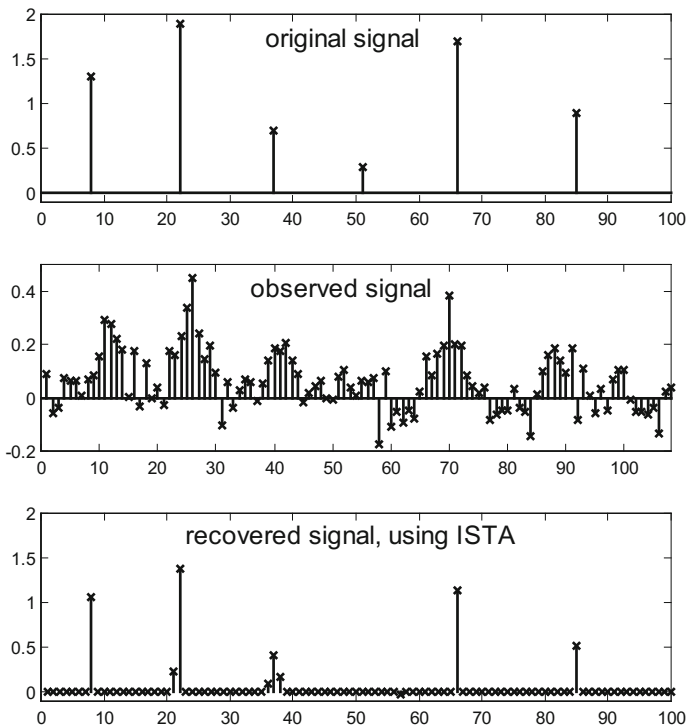
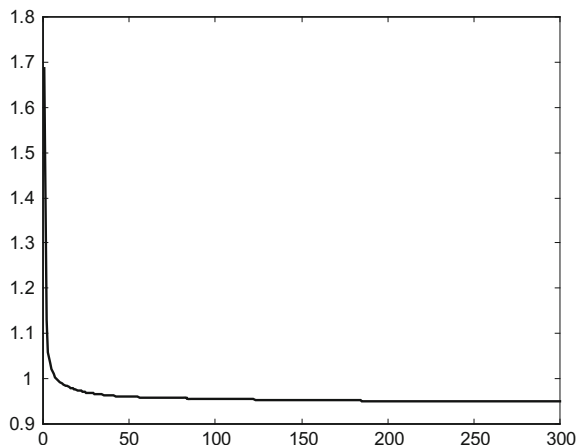


Fig. 2.6 Application of ISTA for a sparse signal recovery example

Fig. 2.7 Evolution of objective function along ISTA iterations



Program 2.2 ISTA example

```
% ISTA example
% 1D sparse signal recovery
% original sparse signal:
x=zeros(1,100);
x(8)=1.3; x(22)=1.9; x(37)=0.7; x(51)=0.3;
x(66)=1.7; x(85)=0.9;
% the signal is observed through a filter
h=[1,2,3,4,5,4,3,2,1]/25; %filter impulse response
y=conv(h,x); L=length(y);
A=convmtx(h',100); %convolution matrix
% gaussian noise is added
y=y+0.06*randn(1,L);
y=y'; %column
% ISTA algorithm
nit=300; %number of iterations
lambda=0.1; %parameter
J=zeros(1,nit); %objective function
z=0*A'*y; %initialize z, the recovered signal
T=lambda/2; %threshold
for k=1:nit,
    q=A*z;
    J(k)=sum(abs(q(:)-y(:)).^2) + lambda*sum(abs(z(:)));
    p=z+(A'*(y-q));
    %soft thresholding (component-wise)
    for n=1:100,
        if abs(p(n)) <= T,
            z(n)=0;
        else
            if p(n)>T,
                z(n)=p(n)-T;
            else
                z(n)=p(n)+T;
            end
        end
    end
end
```

```

else
    z(n)=p(n)+T;
end;
end;
end;
end;
figure(1)
subplot(3,1,1)
[i,j,v]=find(x);
stem(j,v,'kx'); hold on;
axis([0 100 -0.1 2]);
plot([0 100],[0 0],'k');
title('original signal');
subplot(3,1,2)
stem(y,'kx'); hold on;
axis([0 L -0.2 0.5]);
plot([0 L],[0 0],'k');
title('observed signal');
subplot(3,1,3)
stem(z,'kx');
axis([0 100 -0.1 2]);
title('recovered signal, using ISTA')
figure(2)
plot(J,'k')
title('Evolution of objective function')

```

Since the ISTA algorithm involves a gradient descent, it is a good idea to use the Nesterov's accelerated gradient descent (see section on gradient descent in the appendix on optimization). The result is the '*fast iterative shrinkage/thresholding algorithm*' (*FISTA*), which was introduced in 2009 [20].

While ISTA can be expressed as:

$$\mathbf{x}_{k+1} = S_{\lambda/\alpha_k}(\mathbf{x}_k - \frac{1}{\alpha_k} \nabla f(\mathbf{x}_k)) \quad (2.38)$$

FISTA can be expressed as the next three steps:

$$\mathbf{x}_{k+1} = S_{\lambda/L}(\mathbf{y}_k - \frac{1}{L} \nabla f(\mathbf{y}_k)) \quad (2.39)$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2} \quad (2.40)$$

$$\mathbf{y}_{k+1} = \mathbf{x}_k + \left(\frac{t_k - 1}{t_{k+1}} \right) (\mathbf{x}_{k+1} - \mathbf{x}_k) \quad (2.41)$$

There are direct generalizations of the ISTA and FISTA algorithms by replacing the soft-thresholding operator with the proximal operator. This is frequent in the most recent literature. In particular, ISTA is equivalent to the proximal gradient method.

The convergence rate of subgradient methods is $O(1/\sqrt{k})$, proximal gradient methods obtain $O(1/k)$, and accelerated proximal gradient methods further reduce it to $O(1/k^2)$.

Notice that an example of projected subgradient iterations for l_1 minimization was already included in the appendix on optimization. That example was clearly related to a basis pursuit (BP) problem.

Another idea of Nesterov was to substitute non-smooth objective functions by suitable smooth approximations [134]. One application of this idea for our problem, is to consider a smooth approximation of the l_1 norm minimization as follows.

- First, the minimization is written as minimax:

$$\min_x \|\mathbf{x}\|_1 = \min_x \max_u \mathbf{u}_T \mathbf{x} ; \quad -1 \leq u_i \leq 1 \quad (2.42)$$

- Second, the smoothed approximation is introduced:

$$\text{smoothed} \min_x \|\mathbf{x}\|_1 = \min_x \max_u \mathbf{u}_T \mathbf{x} + \frac{\mu}{2} \|\mathbf{u}\|_2^2 \quad (2.43)$$

Based on this approach, a popular algorithm called NESTA was introduced in [21]. A MATLAB implementation is available (see the section on resources). Other relevant algorithms are TwIST [23], SALSA [5], and SpARSA [186].

2.2.3.4 Interior-Point Methods

Since the standard sparsity optimization settings correspond to LP or QP problems, the interior-point methods described in the appendix on optimization can be chosen for solving them. Actually, some of the first algorithms found in the literature on basis pursuit or on ridge regularization adopted this approach [55, 165]. See also the page *l1-magic* on Internet.

Large scale problems may be out of reach, or require excessive computational effort, for conventional interior-point methods. In such situations other approaches, like for instance first-order methods, would be preferred. Responding to this concern, an interior-point method was introduced by [106] that is able to cope with large scale Lasso problems. It is based on a preconditioned conjugate gradient algorithm. This aspect, preconditioning of the CG in relation with sparsity, is becoming important for practical applications [105].

A matrix-free interior-point method has recently been introduced in [83]. The method performs favorably with large scale one-dimensional signals.

2.2.3.5 Alternating Directions

Let us now focus on the ‘*alternating direction method of multipliers*’ (ADMM). According with [192], this approach was not widely employed in the field of image and signal processing until recently, from 2009. It is an iterative method.

Consider the problem:

$$\text{minimize } f(\mathbf{x}) + g(\mathbf{x})$$

where $f(\mathbf{x})$ and $g(\mathbf{x})$ are closed proper convex functions. Both functions can be non-smooth.

The ADMM method can be expressed as follows [145]:

$$\mathbf{x}_{k+1} = \text{Pr } \text{ox}_{\lambda f}(\mathbf{z}_k - \mathbf{u}_k) \quad (2.44)$$

$$\mathbf{z}_{k+1} = \text{Pr } \text{ox}_{\lambda g}(\mathbf{x}_{k+1} + \mathbf{u}_k) \quad (2.45)$$

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{x}_{k+1} - \mathbf{z}_{k+1} \quad (2.46)$$

Typically $g(\mathbf{x})$ encodes the constraints and $\mathbf{z}_k \in \text{dom } g(\mathbf{x})$ (the domain of $g(\mathbf{x})$). On the other hand, $\mathbf{x}_k \in \text{dom } f(\mathbf{x})$. Along the iterations, \mathbf{x}_k and \mathbf{z}_k converge to each other. The ADMM method is helpful when the proximal mappings of $f(\mathbf{x})$ and $g(\mathbf{x})$ are easy to compute but the proximal mapping of $f(\mathbf{x}) + g(\mathbf{x})$ is not so.

Equivalently, the method can be expressed in terms of an ‘*augmented Lagrangian*’, starting from the following problem:

$$\text{minimize } f(\mathbf{x}) + g(\mathbf{z}) \text{ subject to } \mathbf{x} - \mathbf{z} = 0$$

(the problem is expressed in ‘*consensus form*’)

An augmented Lagrangian can be written:

$$L(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{x} - \mathbf{z}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 \quad (2.47)$$

and the ADMM would be:

$$\mathbf{x}_{k+1} = \arg \min_{\mathbf{x}} L(\mathbf{x}, \mathbf{z}_k, \mathbf{y}_k) \quad (2.48)$$

$$\mathbf{z}_{k+1} = \arg \min_{\mathbf{z}} L(\mathbf{x}_{k+1}, \mathbf{z}, \mathbf{y}_k) \quad (2.49)$$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \rho(\mathbf{x}_{k+1} - \mathbf{z}_{k+1}) \quad (2.50)$$

It is shown in [145] that these three equations can be translated to the proximal form introduced above.

The ADMM method has connections with classical methods, like the alternating projections of von Neumann (see [9] for an interesting presentation). An extensive treatment of ADMM, with application to Lasso type problems, can be found in [26]. Also, [94] includes accelerated versions, and application examples concerning the elastic net, image restoration, and compressed sensing.

Some times, in problems of the type found in this chapter, it is convenient to consider the indicator function. Given a set C , its indicator function $i_C(\mathbf{x})$ is 0 if $\mathbf{x} \in C$, and ∞ otherwise. The corresponding proximal is $\text{Prox}_{\lambda f}(\mathbf{x}) = \arg \min_{\mathbf{u}} \|\mathbf{u} - \mathbf{x}\|_2^2 = P_C(\mathbf{x})$, which is the projection over set C .

One of the applications of ADMM treated in [26] is the Basis Pursuit (BP) problem. This problem can be written as:

$$\begin{aligned} & \text{minimize } f(\mathbf{x}) + \|\mathbf{z}\|_1 \\ & \text{subject to } \mathbf{x} - \mathbf{z} = \mathbf{0} \end{aligned}$$

where $f(\mathbf{x})$ is the indicator function of the set $C = \{\mathbf{x} \mid A\mathbf{x} = \mathbf{b}\}$. In this setting, the ADMM iterations are the following:

$$\mathbf{x}_{k+1} = P_C(\mathbf{z}_k - \mathbf{u}_k) \quad (2.51)$$

$$\mathbf{z}_{k+1} = S_{(1/\rho)}(\mathbf{x}_{k+1} + \mathbf{u}_k) \quad (2.52)$$

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{x}_{k+1} - \mathbf{z}_{k+1} \quad (2.53)$$

where $S_{(1/\rho)}$ is the soft thresholding operator, and the projection P_C can be computed as follows:

$$\mathbf{x}_{k+1} = (I - A^T(AA^T)^{-1} - A)(\mathbf{z}_k - \mathbf{u}_k) + A^T(AA^T)^{-1}\mathbf{b} \quad (2.54)$$

Program 2.3 provides an implementation example of ADMM for the BP problem, closely related to the examples given in an ADMM web page (see the section on resources). The program assigns random values to the matrix A and to \mathbf{b} . Then, it finds the sparsest solution to $A\mathbf{x} = \mathbf{b}$.

Of course, each time one executes the Program 2.3 different sparsest solutions are found, since both A and \mathbf{b} are different each time. Figure 2.8 shows one of these sparsest solutions.

Figure 2.9 shows the evolution of $\|\mathbf{x}\|_1$ during a program execution.

Fig. 2.8 A BP sparsest solution

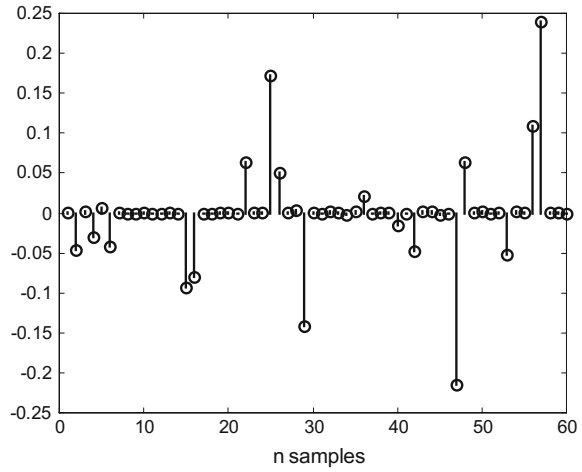
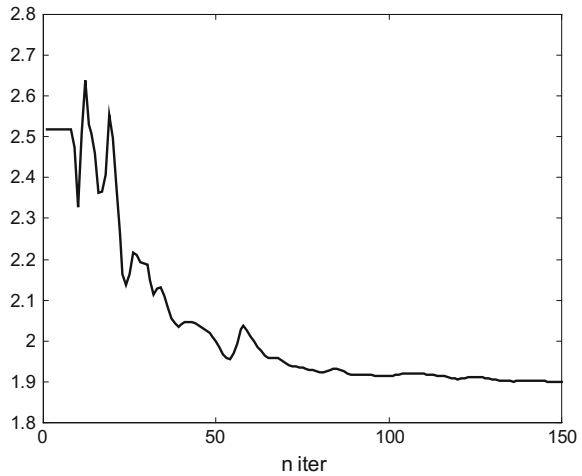


Fig. 2.9 Evolution of objective function $\|x\|_1$ along ADMM iterations



Program 2.3 Example of ADMM for Basis Pursuit

```
% Example of ADMM for Basis Pursuit
n=60; m=20;
A=randn(m,n);
b=rand(m,1);
% ADMM algorithm for finding sparsest solution
niter=150;
x=zeros(n,1); z=zeros(n,1); u=zeros(n,1);
rob=zeros(niter,1);
alpha=1.0; mu=1.0;
uA=inv(A*A');
aux1=eye(n) - (A' * uA * A);
```

```

aux2=A'*uA*b;
for nn=1:niter,
    % x update
    x=(aux1*(z-u))+aux2;
    % z update
    zo=z;
    xe=(alpha*x)+((1-alpha)*zo);
    aux=xe+u; imu=1/mu;
    z=max(0,aux-imu)-max(0,-aux-imu); %shrinkage
    % u update
    u=u+(xe-z);
    % recording
    rob(nn)=norm(x,1);
end
% display
figure(1)
stem(x,'k');
title('the sparsest solution x');
xlabel('n samples');
figure(2)
plot(rob,'k');
title('evolution of norm(x,1)');
xlabel('n iter');

```

2.2.3.6 Douglas-Rachford Splitting

As recognized by the scientific literature, the Douglas-Rachford algorithm is one of the most successful optimization methods, able to tackle nonsmooth objective functions. The method has also been applied to nonconvex problems. An interesting series of applications is cited in [8], including image retrieval, graph coloring, protein studies, matrix completion, and even nonograms.

The optimization problem is, again:

$$\text{minimize } f(\mathbf{x}) + g(\mathbf{x})$$

Using the formulation given in [61], the algorithm is simply:

$$\mathbf{x}_k = \text{Prox}_{\lambda g}(\mathbf{y}_k) \quad (2.55)$$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \lambda_k (\text{Prox}_{\lambda f}(2\mathbf{x}_k - \mathbf{y}_k) - \mathbf{x}_k) \quad (2.56)$$

where $\lambda_k \in [\varepsilon, 2 - \varepsilon]$.

See the Thesis [74] for an extensive treatment of splitting methods. The history of the Douglas-Rachford algorithm and other aspects of interest are concisely included in [61]. There is a web page focusing on Douglas-Rachford (see the section on resources).

An implementation example is provided in the next section for a compressive sensing case.

2.2.3.7 Matching Pursuits

There is a series of methods based on ‘*matching pursuit*’ (MP). The basic idea is simple and familiar. Imagine you have a puzzle with a certain number of pieces. To assemble the puzzle, the usual procedure would be to select pieces that match with certain empty regions of the puzzle. You select and place pieces one after one. This methodology can be applied to many types of problems; actually, some authors refer to it as a *meta-scheme*.

In the next sections we shall see the method being applied to ‘*dictionaries*’. If you place yourself in the Fourier context, it would be natural to combine sinusoidal harmonics. Or, perhaps, it would be better for your problem to combine arcs and line segments to analyze image shapes, etc.

For now, let us focus on the problem considered in this section: to find a sparse vector \mathbf{x} such that:

$$A \mathbf{x} = \mathbf{b} \quad (2.57)$$

Most of the section has been centered on optimization. Matching pursuit is more related with variable selection. Like in the LARS algorithm, one uses residuals and select most correlated columns of A . After setting the initial residual as $\mathbf{r}_0 = \mathbf{b}$ and a counter $k = 0$, the basic MP would be as follows:

-
- Find the \mathbf{A}_j most correlated with \mathbf{r}_k
 - Update the active set J with the selected j .
 - $x_j \leftarrow (\mathbf{A}_j^T \mathbf{r}_k)$
 - $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - \mathbf{A}_j x_j$
 - $k \leftarrow k + 1$

(repeat until stop criterion)

This basic algorithm suffers from slow convergence. The ‘*orthogonal matching pursuit*’ (OMP) has better properties. Due to the orthogonalization, once a column is selected, it is never selected again in the next iterations. The OMP algorithm starts as the MP algorithm, and then:

- Find the \mathbf{A}_j most correlated with \mathbf{r}_k
- Update the active set J with the selected j .

- Update the matrix A_J
- $\mathbf{x} \leftarrow (A_J^{-1}\mathbf{b})$ (solve the least squares regression)
- $\mathbf{r}_{k+1} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}$
- $k \leftarrow k + 1$

(repeat until stop criterion)

Once the suitable columns of A were selected, a matrix B can be built with these columns. Now, there is a theorem that establishes that OMP will recover the optimum representation if:

$$\max_{A_i} \|B^+ A_i\|_1 < 1 \quad (2.58)$$

where A_i are the columns of A not in B , and B^+ is the pseudo-inverse of B , [27].

Other variants of this approach are ‘*stagewise orthogonal matching pursuit*’ (*StOMP*), ‘*regularized orthogonal matching pursuit*’ (*ROMP*), and ‘*compressive sampling matching pursuit*’ (*CoSaMP*). A concise description of them is included in [27].

An example of OMP application is given below. The case $A\mathbf{x} = \mathbf{b}$ is considered. Both A and \mathbf{b} are created with random values. Using OMP, a sparse \mathbf{x} is found. Figure 2.10 shows the result.

A record of the residual values along iteration steps was done. Figure 2.11 plots the evolution of the residual until a certain specified low value is reached. Both Figs. 2.10 and 2.11 have been generated with the Program 2.4, which includes a MATLAB implementation of OMP. Note that *pinv()* has been used for the least square regression.

Fig. 2.10 Sparse solution obtained with OMP

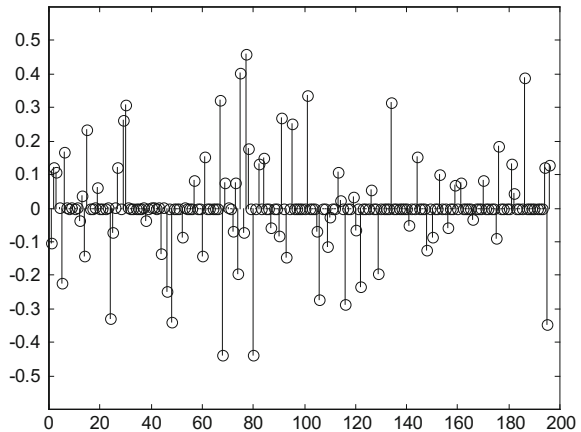
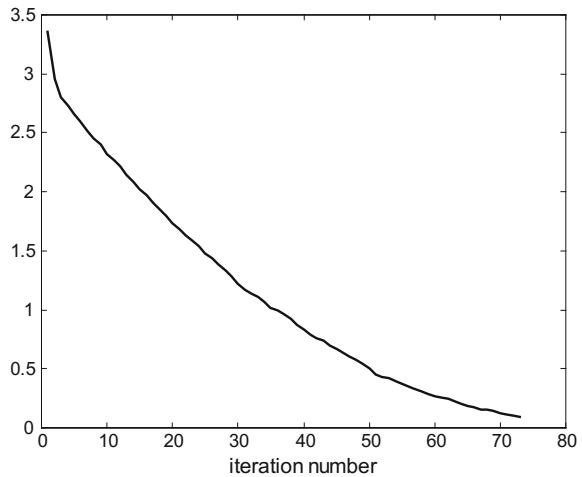


Fig. 2.11 Evolution of the norm of the residual**Program 2.4** Example of OMP

```
% Example of OMP
clear all;
% random A and b
N=100;
A=rand(N,2*N);
b=rand(N,1);
M=1; %measure of the residual
hM=zeros(200,1);
% OMP algorithm
Aj=[];
R=b; %residual
n=0; %counter
while M>0.1, %(edit this line)
    [v,ix]=max(abs(A'*R));
    Aj(:,ix)=A(:,ix);
    A(:,ix)=0;
    %x=Aj\b;
    x=pinv(Aj)*b;
    R=b-Aj*x;
    M=norm(R);
    n=n+1; hM(n)=M; %keep a record of residuals
end;
%display
figure(1)
stem(x,'k');
title('sparse solution');
axis([0 200 -0.6 0.6]);
figure(2)
plot(hM(1:n),'k');
```

```
title('evolution of the residual norm');
xlabel('iteration number');
```

2.3 Compressed Sensing

The compressed sensing (CS) theory claims that it is possible to recover certain signals from fewer samples than the Nyquist rate required by Shannon. This is based on the observation that many natural signals, such sound or images, can be well approximated with a sparse representation in some domain. For example, most of the energy of typical images is preserved with less than 4 % of the wavelet coefficients [153, 175].

The term ‘compressed sensing’ was coined by Donoho [68] in 2006. Other important contributions came, the same year, in [34, 41]. Since then, the theory has advanced and nowadays there are books on the mathematics of CS, like [62, 82]; however, there are still open questions, as we shall see.

Instead of acquiring with a sensor large amounts of data and then compress it, the new paradigm proposes to directly obtain compressed data. The recovery of the signal would be done via an optimization process.

These ideas are supported by the scenario that has appeared several times in this book, having the form of a system of linear equations $A \mathbf{x} = \mathbf{b}$. In particular, our interest focuses on undetermined systems.

One of the peculiar aspects of CS is that it is convenient to select a random matrix to be used as matrix A .

2.3.1 Statement of the Approach

The idea of CS is based on the following fact: if you have a signal \mathbf{x} (n samples) and a $m \times n$ matrix A , with $m < n$, you can write the following equation:

$$A \mathbf{x} = \mathbf{b} \tag{2.59}$$

This is an undetermined system of linear equations: more unknowns than equations. However, if the matrix A is appropriate and the signal is sparse, then it is possible to recover \mathbf{x} from \mathbf{b} .

Exploiting this fact, the practical use of compressed sensing can be to apply a linear operator A to a sparse signal \mathbf{x} ; in other words, compute $\mathbf{b} = A \mathbf{x}$. Then, you just store \mathbf{b} instead of \mathbf{x} . This \mathbf{b} has less numbers (is denser) than \mathbf{x} .

The application of the linear operator could be done in real time, in your sensor. From the point of view of CS, the application of the operator A to the signal \mathbf{x} is regarded as a sampling process. Therefore A is called the *sampling matrix* (or the *sensing matrix*). A $m \times n$ matrix A takes m samples from \mathbf{x} . In the case that A was a random matrix, CS would tell that there is a random sampling of \mathbf{x} via A .

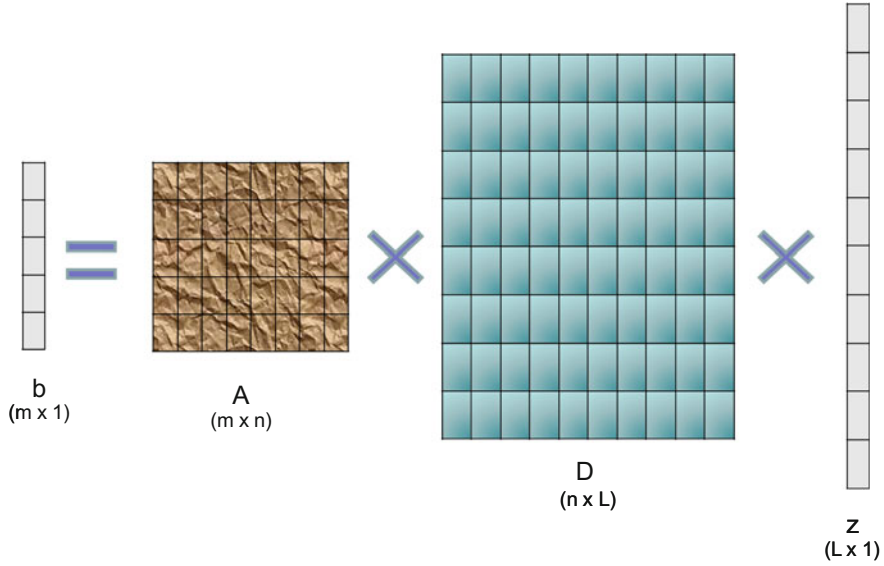


Fig. 2.12 The CS scheme

If you have a signal \mathbf{z} that can be expressed in a sparse format, like for instance $\mathbf{x} = \mathbf{D} \mathbf{z}$, then you can use $\mathbf{b} = \mathbf{A} \mathbf{x} = \mathbf{A} \mathbf{D} \mathbf{z}$ for adhering to a compressed sensing. A typical example would be a signal \mathbf{z} that can be expressed with a few Fourier harmonics. In general, \mathbf{D} would represent the use of a dictionary; and the operation $\mathbf{D} \mathbf{z}$ could be regarded as a ‘*sparsification*’ of the signal \mathbf{z} . The diagram shown in Fig. 2.12 illustrates the approach.

An analysis of these ideas under the view of information flow, would say that you are exploiting the low-rate of information being transmitted by \mathbf{z} ; so, in fact, you can concentrate the same information in \mathbf{b} .

In relation with these points, it must be said that (strictly) sparse signals are rarely encountered in real applications. What is more probable is to find signals with a small number of elements significantly non-zero (the rest could also be non-zero). Such signals are considered as ‘*compressible*’. One way of formalizing this concept is to look for an exponential decay of values; the signal components are rearranged in a decreasing order of magnitude $|x_{(1)}| \geq |x_{(2)}| \geq \dots \geq |x_{(n)}|$ and then one checks that:

$$|x_{(k)}| \leq R k^{1/p} \quad (2.60)$$

In general, compressible signals are well approximated by s -sparse signals \mathbf{x}_s :

$$\|\mathbf{x} - \mathbf{x}_s\|_1 \leq C_p R s^{1-1/p} \quad (2.61)$$

$$\|\mathbf{x} - \mathbf{x}_s\|_2 \leq D_p R s^{1/2-1/p} \quad (2.62)$$

with: $C_p = (1/p - 1)^{-1}$ and $D_p = (2/p - 1)^{-1/2}$ (see [133]).

2.3.2 Compression and Recovery. The Matrix A

In order to apply the CS approach, one has to guarantee that the compressed signal can be recovered. This is the question that we want to consider now.

A first and important point is that, while in regression problems you inherit a given matrix A formed from experimental data, in the case of compressed sensing you are the responsible of designing the matrix A . In consequence, it is convenient to have some guidance for building A .

In relation to this aspect—the design of A —an important step in the early development of CS theory was to establish the ‘*restricted isometry property*’ (RIP). Let us introduce with some care this property.

Recall that a vector \mathbf{x} is s -sparse if at most s of its entries are non-zero. A first, natural question would be: how many samples are necessary to acquire s -sparse signals? The answer is that m should be $m \geq 2s$ (note that the dimensions of A are $m \times n$). As explained in [133], the sampling matrix must not map two different sparse signals to the same set of measurement samples. Hence, each collection of $2s$ columns from A must be nonsingular.

In this line of thinking, [35] considered that the geometry of sparse signals should be preserved under the action of the sampling matrix. The ‘ s -th restricted isometry constant δ'_s ’ was defined as the smallest number δ_s such that:

$$(1 - \delta_s) \|\mathbf{x}\|_2^2 \leq \|A \mathbf{x}\|_2^2 \leq (1 + \delta_s) \|\mathbf{x}\|_2^2 \quad (2.63)$$

When $\delta_s < 1$, the expression above implies that each collection of s columns of A is non-singular, so $(s/2)$ -sparse signals can be acquired. In case $\delta_s < 1$, the action of A would nearly maintain the l_2 distance between each pair of signals (the term *isometry* refers to keep the distance).

It is said that A satisfies the *restricted isometry property* (RIP) if it has an associated isometry constant with value $\delta_s < 1$.

Now, let us see if we will be able to recover \mathbf{x} from \mathbf{b} . One of the alternatives considered in CS is to state and solve a BP problem:

$$\text{minimize } \|\mathbf{x}\|_1, \text{ subject to } A \mathbf{x} = \mathbf{b}$$

According with [37], having got the BP solution \mathbf{x}^* , this solution recovers \mathbf{x} exactly if the signal is sufficiently sparse and the matrix A has the RIP property. Moreover, assume that $\delta_{2s} < \sqrt{2} - 1$, then (theorem):

$$\|\mathbf{x}^* - \mathbf{x}\|_1 \leq C_0 \|\mathbf{x} - \mathbf{x}_s\|_1 \quad (2.64)$$

and:

$$\|\mathbf{x}^* - \mathbf{x}\|_2 \leq C_0 s^{1/2} \|\mathbf{x} - \mathbf{x}_s\|_1 \quad (2.65)$$

where \mathbf{x}_s is the vector \mathbf{x} with all but the s -largest entries set to zero. See [37] for the constant C_0 , which is rather small. In case that \mathbf{x} was s -sparse, the recovery is exact.

Two consequences of the theorem are that:

- if $\delta_{2s} < 1$ the l_0 problem has a unique s -sparse solution
- if $\delta_{2s} < \sqrt{2} - 1$ the solution of the l_1 problem is the same as the l_0 problem solution

Part of the research is trying to find larger values of δ_s still guaranteeing the recovery. For instance, recently (year 2013), a value of 0.5746 (instead of 0.4142) has been established in [202]. This paper includes a table, with key references, of the previous results from other specialists.

Normally, one wants a matrix A with a small δ_s . It has been found that many types of random matrices have very good δ_s . Often a value $\delta_s \leq 0.1$ can be obtained. On the contrary, it is difficult for deterministic matrices to have a good δ_s .

Random matrices can be obtained in several ways. Here is a short selection:

- **Gaussian matrices:** the entries of A are independent normal variables, with zero mean and $1/m$ variance; that is: $a_{i,j} \in N(0, 1/m)$
- **Bernoulli matrices:** the entries of A are:

$$a_{i,j} = \begin{cases} +1/\sqrt{m}, & \text{with probability } 1/2 \\ -1/\sqrt{m}, & \text{with probability } 1/2 \end{cases} \quad (2.66)$$

- **Partial Fourier matrices:** the entries of A are random samples of a Fourier matrix

See [42] and [133] for more alternatives, and details. A contemporary research topic is to deterministically generate matrices with good RIP.

Supposing that the signal recovery would be done with BP, it is important to consider also the ‘null space property’ (NSP). Let us introduce this property.

Recall that the null space $\text{Nul}(A)$ of the matrix A is the set of all solutions of $A\mathbf{x} = 0$.

Consider the set of indexes $L = \{1, 2, \dots, n\}$ and choose for example a subset $S \subset L$ with $|S| = s$. Then, any vector \mathbf{x} can be written as the sum $\mathbf{x}_S + \mathbf{x}_{\bar{S}}$, where \bar{S} is the complement of S , the non-zero elements of \mathbf{x}_S are in S , and the non-zero elements of $\mathbf{x}_{\bar{S}}$ are in \bar{S} . For example, suppose that $\mathbf{x} = \{4, 3, 2, 1, 5, 6\}$, a possible decomposition could be: $\mathbf{x}_S = \{0, 0, 2, 1, 5, 0\}$ and $\mathbf{x}_{\bar{S}} = \{4, 3, 0, 0, 0, 6\}$.

The matrix A satisfies the null space condition of order s if for any non-zero vector $\mathbf{x} \in \text{Nul}(A)$ and any index subset S with $|S| = s$, one has:

$$\|\mathbf{x}_S\|_1 < \|\mathbf{x}_{\bar{S}}\|_1 \quad (2.67)$$

Intuitively, the NSP implies that non-zero vectors in the null space cannot be too sparse. The problem we want to avoid is having a sparse vector with $A\mathbf{x} = 0$, which would clearly interfere with the recovery of other vectors.

When dealing with exactly sparse vectors, the spark characterizes when recovery is possible. However, when dealing with approximately sparse signals (for instance when there is noise), a more restrictive condition about the vectors in $\text{Nul}(A)$ is needed, and this is why the NSP [64]. In relation with RIP, it can be shown that RIP implies NSP; that is: RIP is stronger than NSP.

A theorem establishes that BP will obtain exact recovery iff A satisfies the NSP [3, 157].

For the interested reader it could be opportune to consult [179] about the relationship between the irrepresentable condition, already mentioned in the Lasso context, and the RIP. The irrepresentable condition is more restrictive.

Nowadays, experience with some applications is showing that the RIP condition—which is a sufficient condition is too stringent. The theory is advancing, and new, weaker conditions are being discovered or re-discovered [4].

As an example of sparse signal recovery, suppose that one has a sparse 1D signal, then one uses a sensing matrix A to obtain samples (a vector \mathbf{b}), and then applies the Douglas-Rachford algorithm to recover the original signal. This example is borrowed from a contribution of G. Peyre to the MATLAB Central file exchange site (Toolbox of Sparse Optimization).

Program 2.5 handles this example and, at the same time, provides an example of implementation of the Douglas-Rachford algorithm.

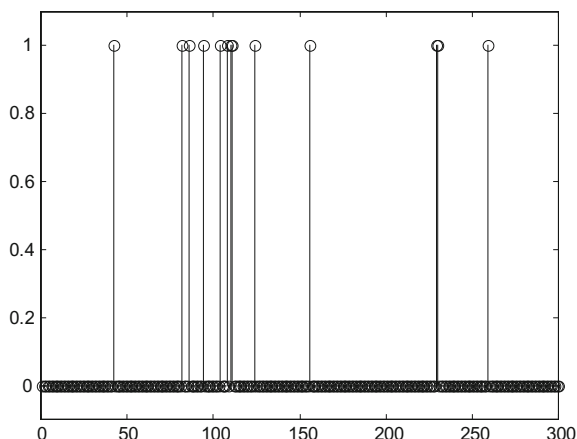
Figure 2.13 shows the original signal to be sampled.

The signal recovery can be considered as the following problem:

$$\text{minimize } f(\mathbf{x}) + g(\mathbf{x})$$

where $f(\mathbf{x})$ is the indicator function and $g(\mathbf{x})$ is $\|\mathbf{x}\|_1$.

Fig. 2.13 An sparse signal being measured



The Douglas-Rachford algorithm can be written in this case as follows:

$$\mathbf{x}_k = \text{Prox}_{\lambda f}(\mathbf{y}_k) = \mathbf{y}_k + A^T (A A^T)^{-1} (\mathbf{b} - A \mathbf{y}_k) \quad (2.68)$$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \lambda_k (\text{Prox}_{\lambda g}(2\mathbf{x}_k - \mathbf{y}_k) - \mathbf{x}_k) \quad (2.69)$$

where $\text{Prox}_{\lambda g}(\cdot)$ is soft thresholding.

(notice that $f(\mathbf{x})$ and $g(\mathbf{x})$ have been exchanged with respect to the description of the algorithm given in the previous section; both are equivalent).

Figure 2.14 shows the recovered signal.

The evolution of the algorithm can be followed in several ways. For example, Fig. 2.15 shows the evolution of $\|\mathbf{x}\|_1$ along the iterations.

Fig. 2.14 Recovered signal

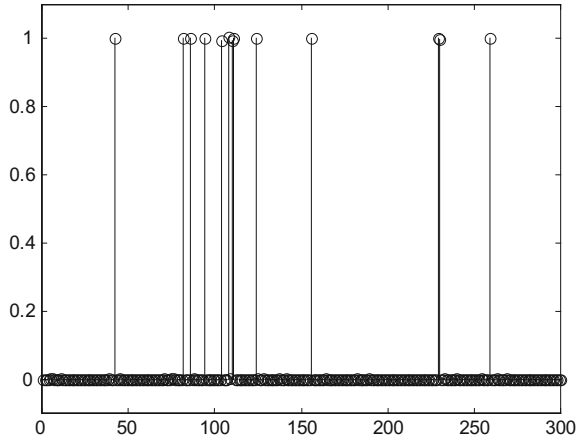
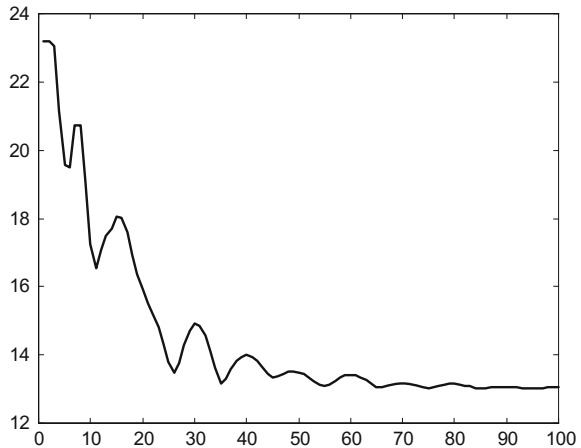


Fig. 2.15 Evolution of $\|\mathbf{x}\|_1$ along iterations



Program 2.5 Example of Douglas-Rachford

```

% Example of Douglas-Rachford
% for compressed sensing of 1D signal
clear all;
N=300; %problem dimension
q=N/4; %number of measurements
A=randn(q,N)/sqrt(q); %random Gaussian sensing matrix
% A s-sparse signal, with s non-zero values (=1)
s=13;
x0=zeros(N,1);
aux=randperm(N);
x0(aux(1:s))=1;
% display
% the signal to be measured
figure(1)
stem(x0,'k');
axis([0 N -0.1 1.1]);
title('the original signal')
% perform random measurements
b=A*x0;
% Algorithm for recovering the measured signal
niter=100;
lambda=1.0; ; gamma=1.0;
x=zeros(N,1);
y=zeros(N,1);
rnx=zeros(niter,1);
uA=A'*inv(A*A');
for nn=1:niter,
    % x update
    x=y+(uA*(b-(A*y))); %proxF(indicator function)
    %proxG (soft thresholding):
    aux=(2*x)-y;
    S=max(0,1-gamma./max(0,abs(aux))).*aux; %soft th.
    % y update
    y=y+(lambda*(S-x));
    % recording
    rnx(nn)=norm(x,1);
end
% display
% recovered signal
figure(2)
stem(x,'k');
axis([0 N -0.1 1.1]);
title('recovered signal')
% evolution of ||x||1
figure(3)
plot(rnx,'k');
title('evolution of signal norm-1')
axis([0 100 12 24]);

```

2.3.3 Incoherence and Sensing

Some of the CS authors have proposed a kind of uncertainty principle, in the vein of the time-frequency duality [42, 76]. It is illustrative to have a quick look to this aspect.

Two important examples of orthogonal matrices are the identity matrix I and the Fourier matrix F . Both matrices correspond to two orthobases, one allows for time-domain representation of signals, and the other for frequency-domain representation. More in general, given two orthobases Ψ and Φ , the signal \mathbf{b} can be represented as follows:

$$\mathbf{b} = \Psi \alpha = \Phi \beta \quad (2.70)$$

Suppose that the matrix A was the concatenation of two orthogonal matrices Ψ and Φ . A particular example could be $A = [I, F]$; a sparse approximation of the signal \mathbf{b} would be a superposition of spikes and sinusoids.

The ‘mutual-coherence’ $\mu(A)$ is defined as follows [76]:

$$\mu(A) = \text{proximity}(\Psi, \Phi) = \max_{i,j} |\Psi_i^T \cdot \Phi_j| \quad (2.71)$$

where Ψ_i and Φ_j are columns. It can be shown that $(1/\sqrt{n}) \leq \mu(A) \leq 1$. For the case $A = [I, F]$, the mutual-coherence is $\mu(A) = (1/\sqrt{n})$.

If the two orthogonal matrices were not normalized, the mutual coherence would be expressed as follows:

$$\mu(A) = \max_{i,j} \frac{|\Psi_i^T \cdot \Phi_j|}{\|\Psi_i\|_2 \|\Phi_j\|_2} \quad (2.72)$$

The interesting fact is that, according with a theorem [76], one has:

$$\|\alpha\|_0 + \|\beta\|_0 \geq \frac{2}{\mu(A)} \quad (2.73)$$

Therefore, if the mutual-coherence of two bases is small, then α and β cannot both be very sparse. This can be regarded as an uncertainty principle. In particular, a signal cannot be sparsely represented both in time and frequency.

There is a theorem in [42] establishing the following: suppose the signal \mathbf{x} is s -sparse and take m samples of it. Then if:

$$m \geq C \mu^2(A) \cdot s \cdot \log n \quad (2.74)$$

for some $C > 0$, then BP will exactly recover from $\mathbf{b} = A \mathbf{x}$ the signal \mathbf{x} with very high probability.

In consequence, the smaller the coherence, the fewer samples are needed [42]. Then, in general, incoherent sampling would be recommended.

The case $A = [I, F]$ has maximal incoherence. The coherence of *noiselets* [60] and Haar wavelets is $(\sqrt{2}/\sqrt{n})$. The coherence of random matrices and a fixed basis would be about $(\sqrt{2 \log n}/\sqrt{n})$ (see [42]).

See [108] for details on the relations between spark, NSP, mutual coherence, and RIP.

2.3.4 Stable and Robust Recovery

Usually, signals are not exactly sparse but compressible. Given a compressible signal \mathbf{x} , it can be approximated by a sparse signal \mathbf{x}_s being the vector \mathbf{x} with all but the s -largest entries set to zero, with the following error:

$$\sigma = \|\mathbf{x} - \mathbf{x}_s\|_1 \quad (2.75)$$

In addition, real life measurement is always contaminated with noise \mathbf{e} . Then:

$$\mathbf{b} = A \mathbf{x} + \mathbf{e} \quad (2.76)$$

It is assumed that $\|\mathbf{e}\|_2 < \eta$.

Returning to basis pursuit, it was found that in the presence of noisy or imperfect data, it is better to state the problem as:

$$\text{minimize } \|\mathbf{x}\|_1, \text{ subject to } \|A \mathbf{x} - \mathbf{b}\|_2 \leq \eta$$

This problem is called ‘*basis pursuit denoise*’ (BPDN), and it was proposed in the famous paper of [55]. Clearly, the problem looks like a Lasso problem. In fact, BPDN became so popular that the literature sometimes use the term BPDN to refer to Lasso.

Assuming that $\delta_{2s} < \sqrt{2} - 1$, one of the theorems presented in [37] establishes that the recovery obtained by BPDN solution obeys:

$$\|\mathbf{x}^* - \mathbf{x}\|_2 \leq C_0 s^{1/2} \sigma + C_1 \eta \quad (2.77)$$

for constants $C_0, C_1 > 0$. A small C_0 would mean that the recovery is *stable* with respect to inexact sparsity; and a small C_1 would mean that the recovery is *robust* with respect to noise. It is established in the theorem that both constants are rather small. For example, with $\delta_{2s} = 1/4$ the values of the constants would be $C_0 \leq 5.5$ and $C_1 \leq 6$, [42]. In the noiseless case, the solution could be found with BP, and will have the same value of C_0 .

Some literature is considering alternatives to BPDN. One of these alternatives is the ‘*Dantzig selector*’, which was proposed by [36] for cases where the number of

variables is much larger than the number of observations. The problem is formulated as follows:

$$\text{minimize } \|\mathbf{x}\|_1, \text{ subject to } \|A^T(\mathbf{b} - A\mathbf{x})\|_\infty \leq c$$

(the norm $\|\cdot\|_\infty$ is just the maximum absolute value).

2.3.5 Phase Transitions

As a preliminary step for introducing phase transitions, define two parameters:

- The undersampling ratio: $\delta = m/n$, $\delta \in [0, 1]$
- The oversampling ratio: $\rho = s/m$, $\rho \in [0, 1]$

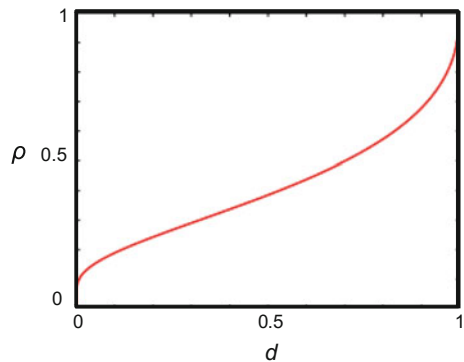
Imagine you take pairs (δ, ρ) and represent with colour pixels, on the plane ρ vs δ the difficulty of signal recovery using l_1 norm. Smaller δ and larger ρ would mean more recovery difficulty.

Based on counting faces of polytopes, Donoho and Tanner found what they called ‘phase transitions’ associated to l_1 recovery. Figure 2.13 shows a phase transition curve similar to the result of that study using a Gaussian matrix A . It can be obtained in the Monte-Carlo style with many (δ, ρ) trials; the curve represented in the figure corresponds to the points where the recovery success probability crosses 50 %. The epigraph of the curve is a region of improbable solvability (Fig. 2.16).

The curve represented in the figure corresponds to the noiseless recovery case. A kind of universality of phase transitions was observed [67], so the gaussianity of A can be considerably relaxed. See [70] for a extensive treatment that includes phase transitions.

Phase transitions provide a way to compare recovery methods that has been adopted by many research publications. An important aspect is what happens in the noisy case; according with [205] several performance regions could be recognized.

Fig. 2.16 A phase transition curve



2.3.6 *Some Applications*

Evidently, the potential of CS is suitable for applications in need of compression, like modern digital communication. In addition, there are applications in which something similar to sampling with random matrices occur, due to the characteristics of the sensorial system—for instance a network of sensors—or because the random nature of the process under measurement—for instance traffic-.

First examples of CS applications are being published. The intention in this part of the CS section is to show the variety and interest of what is going on in this field.

With respect to communication networks, [102] shows with some detail, and many references, that CS has a range of applications from the physical layer through the application layer, reaching in many cases performance gains of the order of ten times. In [100] a user's guide to CS for communication is offered, its first half being a good concise compendium of CS theory and algorithms. The second half of this article is devoted to CS applications for wireless channel estimation, wireless sensor networks, network tomography, cognitive radio, array signal processing, etc. Opportune references were given for each application. More specific publications are [14] on estimating sparse multipath wireless channels, and [22] on sparse channel estimation in underwater communications.

Two CS based dimensionality data reduction algorithms were introduced in [88], using for experiments a database of handwritten digits. The application of CS for speech and audio signal was considered in [57] via sparse decompositions using dictionaries of windowed complex sinusoids.

A compressed sensing based fingerprint identification for wireless systems has been introduced in [200]. This is an example of future applications for security.

A block-by-block CS was proposed by [85] for sensing of natural images. This approach has been continued by [131] and others for pictures and video.

Due to constraints of sensing devices the classes of measurement matrices are limited. Also, some kinds of signals exhibit certain structured features. So it is convenient to study algorithms that go beyond the random measurement paradigm. This subject is treated in [73] and other related papers.

There are remote control cases where the communication channel is rate-limited and so it is interesting to investigate the use of CS. This problem is discussed in [132].

A brief review of CS applied to Radar was given by [80]. It included a short realistic discussion of the practical interest of CS in this context. In [183] a CS method for SAR imaging was introduced. It outperforms the conventional SAR algorithm. There are now a significant number of papers proposing CS techniques for several types of Radar and involved issues.

There is an interesting long report on CS application in Defense sensor systems [126] that covers the single-pixel camera, SAR, Doppler Radars, sparse installations of coastal HF radars, etc.

The structural health monitoring is becoming more and more important in engineering. For example, in a detailed contribution [103] describes the application of CS for real-time accelerometer data concerning the Tianjin Yonghe cable-stayed bridge (512 m. total length). An extensive academic study of CS application in this context,

focusing on vibrations, is [87]. In large constructions, the surveillance of the structure should be done with networked sensors, as discussed in [124].

A natural application field for CS is networks of sensors. The issues to be solved are described in [99], and a complete design of a large-scale wireless sensor network was presented in [119]. A centralized iterative hard thresholding algorithm was proposed in [147] for distributed CS in static or time-varying networks. The detection of events is treated by [43, 127]. The use of CS for target counting and localization in sensor networks is presented in [197].

Medical applications are abundant. For instance, [146] describes a fast compressed sensing-based cone-beam computed tomography (CBCT) reconstruction method. A number of publications, like the most cited [120], describe the application of CS for magnetic resonance imaging. An interesting contribution, concerning dynamic MRI, is [104]. The web page of J. Huang contains links to several research activities related to CS and MRI.

With respect to CS in medical ultrasound, there is an extensive academic treatment in [158]; a short review of applications is made in [112]; a description of results is given in [154].

Based on CS, [Ak] proposed a method for human activity sensing using mobile phone accelerometers.

A CS approach to urban traffic sensing with probe vehicles was introduced in [111]. This is a topic of increasing interest.

A representative publication on CS for environmental monitoring is [191], in which the methodology is demonstrated for Ozone and surface air temperature real data. An interesting CS application for ocean monitoring is described in [118].

2.4 Image Processing

Images can be treated with the Fourier transform or wavelets with a number of purposes, like compression, filtering, denoising, deblurring, etc. This has been already treated in different chapters of this book.

However, processing innovations do not stop. Several new ways of decomposing an image into components have been proposed, opening exciting possibilities. Many publications appeared, mainly in two directions: one focuses on edges and the rest, and the other on a kind of spatial dictionaries.

The objective of this section is to introduce with some detail these new approaches, indicating a number of references that can be useful for the reader to dig into the proposed methods.

2.4.1 *Texture + Cartoon*

In a famous work published in 2001 [128], Meyer proposed the decomposition of images into texture and cartoon parts. Different methods can be used for isolating the edges, while other can focus on texture modeling. This approach has prompted an active research on the three involved aspects: edges, textures, and decomposition.

The idea of the decomposition can be simply expressed as a decomposition of the image f into cartoon u and texture v , as follows:

$$f = u + v \quad (2.78)$$

There are authors that prefer other terms, like geometry or structure, for the u term. The v term could include constant color regions, textures that could be modeled with oscillatory functions, and other. Textiles can usually be modeled with oscillatory functions.

Of course, it would be always possible to extract a u component from f , and then obtain v as $f - u$. However, it is more compliant with the purpose of sparse representation to try a model based approximation of v .

Notice that image denoising considers that there is an original image p and a noise w , so one has a noisy image $q = p + w$. The target of the denoising effort is to extract p . Compared with the texture + cartoon decomposition, there are similarities that could be more or less strong depending on the particular problem to be tackled. This should be taken into account when looking at the literature.

It has been pointed out that a decomposition $f = u + v$ seems to be analogous to the high-pass and low-pass decompositions described in previous chapters. However, this will not work as we want: both cartoon and texture contain high frequencies, so a linear filtering cannot separate u and v . Other alternatives should be explored, like for instance variational approaches.

In general, variational approaches try to obtain u as a function that minimizes a certain criterion, which is expressed as an image model.

Looking at the recent history of variational decompositions, as summarized in [29], an important proposal was done in 1989 by Mumford and Shah [130]. It is worthwhile to consider in some detail this contribution.

2.4.1.1 Mumford-Shah, and Image Segmentation

It may well happen that in a certain photograph one wants to separate objects of interest. For instance, cells (recall the example about thresholding in Chap. 1 1), roads, faces, etc. This is the type of applications addressed by image segmentation. It leads typically to the detection of edges, and so it has many things in common with $f = u + v$ decomposition.

Suppose a rectangular picture with multiple objects O_1, O_2, \dots, O_n . There would be a set of regions $\Omega_1, \Omega_2, \dots, \Omega_n$ in the image corresponding to these objects. Denote as Γ the set of smooth arcs that make up boundaries for the regions Ω_i . In total one has: $\Omega = \Omega_1 \cup \Omega_2 \cup \dots \cup \Omega_n \cup \Gamma$. Our main goal is to capture the boundaries while the texture does not vary much inside each object. The image f could in this case be approximated by a piecewise-smooth function u .

Information is related to changes. With a certain analogy, one could speak of energies in the sense of information content. Using the Sobolev's H^1 norm, the energy of a region Ω_i would be:

$$E(\Omega_i) = \int_{\Omega_i} |\nabla u(\bar{x})|^2 d\mathbf{x} \quad (2.79)$$

(the function u must belong to the space H^1 of functions whose derivative is square-integrable).

It is known that functions belonging to H^1 cannot present discontinuities across lines, such as boundaries. Therefore, u alone cannot model the boundaries.

It is opportune to consider the energy of the boundaries, as follows:

$$E(\Gamma) = \text{Length}(\Gamma) \quad (2.80)$$

(more formally, this would be the Hausdorff measure of Γ)

In many applications, there is noise and blurring to be taken into account. A ‘fidelity term’ can be used for this purpose:

$$E(\text{error}) = \int_{\Omega} (f(\mathbf{x}) - u(\mathbf{x}))^2 d\mathbf{x} \quad (2.81)$$

In a most cited, seminal article [130], Mumford and Shah approximated the image f by a piecewise-smooth function u and a set Γ that solves the following problem:

$$\arg \min_{u, \Gamma} \left\{ \mu \text{Length}(\Gamma) + \int_{\Omega - \Gamma} |\nabla u(\mathbf{x})|^2 d\mathbf{x} + \lambda \int_{\Omega} (f(\mathbf{x}) - u(\mathbf{x}))^2 d\mathbf{x} \right\} \quad (2.82)$$

Clearly, it is an energy minimization. It can be shown that removing any of the three terms gives a trivial, not suitable solution.

The observed result of the minimization is usually a simplified version of the original image, similar to a cartoon. It is not a perfect tool; for instance, shadows, reflections, gross textures, may cause difficulties.

Mainly because the term with Γ , it is not easy to solve the minimization problem. Many methods and approximations have been proposed, see [33, 66, 152] and references therein. An Octave implementation is included in [178]. There is a popular related segmentation method based on the Chan-Vese model; this model is described in [89] with an implementation in C available from the web.

An example of image segmentation using the Chan-Vese method will be given below. In preparation for the example, let us include a brief summary of this method. An important simplification is that $u(\mathbf{x})$ can take only two values: c_{in} for \mathbf{x} inside Γ , and c_{out} for \mathbf{x} outside Γ . These constants can be regarded as average gray values:

$$c_{in} = \frac{\int_{in \Gamma} f d\mathbf{x}}{\int_{in \Gamma} d\mathbf{x}} \quad c_{out} = \frac{\int_{out \Gamma} f d\mathbf{x}}{\int_{out \Gamma} d\mathbf{x}} \quad (2.83)$$

Instead of a direct manipulation of Γ , it is represented as zero-crossings of a ‘level set’ function φ (this idea was introduced in [143], a heavily cited article). This function φ is positive inside Γ and negative outside.

The energy to be minimized would be written as follows:

$$E = \mu \int_{\Omega} \delta(\varphi(\mathbf{x})) |\nabla \varphi(\mathbf{x})| d\mathbf{x} + \lambda_{in} \int_{in\Gamma} (f(\mathbf{x}) - c_{in})^2 d\mathbf{x} + \lambda_{out} \int_{out\Gamma} (f(\mathbf{x}) - c_{out})^2 d\mathbf{x} \quad (2.84)$$

The first integral in the previous equation corresponds to the length of Γ . In some cases it would be convenient to add a term with the area enclosed by Γ , to control its size, but it has been not considered in our example.

A semi-implicit gradient descent could be applied for the minimization. This is done with an iterative evolution of:

$$Q = \frac{\varphi_{xx} \varphi_y^2 - 2 \varphi_x \varphi_y \varphi_{xy} + \varphi_{yy} \varphi_x^2}{(\varphi_x^2 + \varphi_y^2)^{3/2}}$$

$$\frac{\partial \varphi}{\partial t} = \delta(\varphi) \cdot [Q - \lambda_{in}(f - c_{in})^2 + \lambda_{out}(f - c_{out})^2] \quad (2.85)$$

(where sub-indexes represent partial derivatives).

The $\delta(\varphi)$ can be approximated with:

$$\delta(\varphi) = \frac{\varepsilon}{\pi(\varepsilon^2 + t^2)} \quad (2.86)$$

In each iteration, the values of c_{in} and c_{out} were updated.

Now, let us introduce the example. Figure 2.17 shows the original image. After running Program 2.6, which is a slightly modified version of code from Fields Institute (see the section on resources), an interesting image segmentation was obtained. The results are shown in Fig. 2.18.

Fig. 2.17 Original image



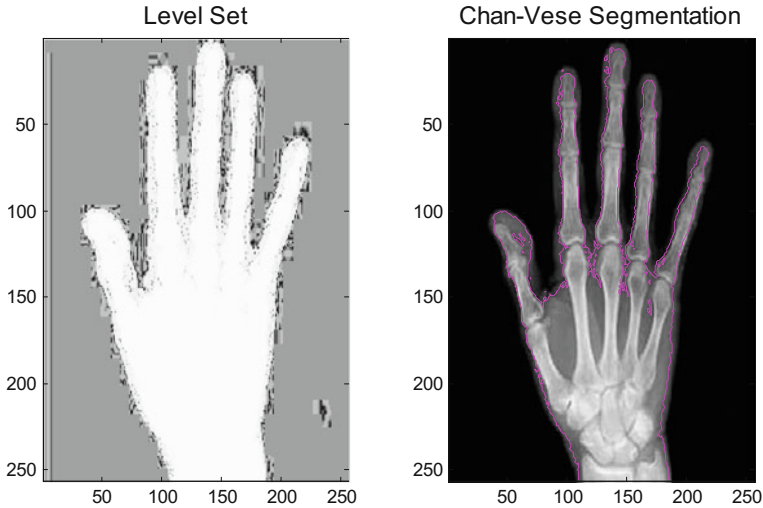


Fig. 2.18 (right) Chan-Vese segmentation, (left) level set

Program 2.6 Example of Chan-Vese algorithm

```
% Example of Chan-Vese algorithm
%(active contours segmentation, using level sets)
I=imread('hand1.jpg'); %read b&w image
I = double(I);
[m,n] = size(I);
lambda=0.1; %parameter
%Set initial box=1, with border=-1
Phi=ones(m,n);
Phi(1,1:n)=-1; Phi(m,1:n)=-1;
Phi(1:m,1)=-1; Phi(1:m,n)=-1;
%Prepare loop
disp('working...');
a = 0.01; %to avoid division by zero.
ep1 = 0.1; %small value
T = 10;
dt = 0.2;
for t = 0:dt:T
    ax2=2*Phi;
    P1=Phi(:, [2:n,n]); P2=Phi([2:m,m], :);
    P3=Phi(:, [1,1:n-1]); P4=Phi([1,1:m-1], :);
    %partial derivatives (approx.)
    Phi_x = (P1 - P3)/2;
    Phi_y = (P2 - P4)/2;
    Phi_xx = P1 - ax2 + P3;
    Phi_yy = P2 - ax2 + P4;
    Q1=Phi([2:m,m], [2:n,n]); Q2=Phi([1,1:m-1], [1,1:n-1]);
```

```

Q3=Phi([1,1:m-1],[2:n,n]); Q4=Phi([2:m,m],[1,1:n-1]);
Phi_xy = (Q1 + Q2 - Q3 - Q4)/4;
%TV term
Num = (Phi_xx.*Phi_y.^2) - (2*Phi_x.*Phi_y.*Phi_xy) +
(Phi_yy.*Phi_x.^2);
Den = (Phi_x.^2 + Phi_y.^2).^(3/2) + a;
%Compute averages
c_in = sum([Phi>0].*I)/(a+sum([Phi>0]));
c_out = sum([Phi<0].*I)/(a+sum([Phi<0]));
%Update
aux=( Num./Den - lambda*(I-c_in).^2 + lambda*(I-c_out).^2);
Phi = Phi + dt*ep1./(pi*(ep1^2+Phi.^2)).*aux;
end;
% display of results
figure(1)
imagesc(I);
title('Original image');
colormap gray;
figure(2)
subplot(121);
imagesc(Phi);
title('Level Set');
subplot(122);
imagesc(I); hold on;
title('Chan-Vese Segmentation');
contour(Phi,[0,0],'m');
colormap gray;

```

2.4.1.2 Total Variation, and BV Functions

Further steps in the direction suggested by the Mumford-Shah model were taken by considering a total variation (TV) term. Already, a brief introduction to TV has been done in the previous book, in the context of image restoring. It is now opportune to include a more extended consideration.

There are excellent publications, with a mathematical formal orientation like [47, 164], that use test functions ϕ belonging to the set $C_0^1(\Omega, \mathbb{R}^2)$ of continuously differentiable vector functions of compact support contained in Ω , and such $\|\phi\|_\infty \leq 1$. These functions are employed for the definition of TV as follows:

$$TV(u) = \sup \left\{ \int_{\Omega} u \operatorname{div} \phi \, d\mathbf{x} , \forall \mathbf{x} \in \Omega \right\} \quad (2.87)$$

Given a differentiable function u defined on a bounded open set Ω , its total variation is:

$$TV(u) = \int_{\Omega} |\nabla u(\mathbf{x})| d\mathbf{x} \quad (2.88)$$

If the TV of the differentiable function u is $TV(u) < +\infty$, then $u \in BV$, where BV is the space of *bounded variation functions*.

The set of bounded variation functions is a Banach space with the norm:

$$\|u\|_{BV} = \|u\|_{L^1} + TV(u) \quad (2.89)$$

Bounded variation functions can have sharp edges. Actually, the norm $\|\cdot\|_{BV}$ takes into account the number of edges.

If a function $u \in BV$ belongs also to the smaller Sobolev space, then the norm is just $TV(u)$. See [59] for an interesting study of BV functions, including wavelets.

The BV functions play an important role in the cartoon + texture decomposition.

2.4.1.3 The Rudin, Osher and Fatemi (ROF) Model

In 1992 Rudin, Osher and Fatemi [163] proposed to apply TV for image denoising, in a variational framework with the following expression:

$$\inf_u \left\{ \int_{\Omega} |\nabla u(\mathbf{x})| d\mathbf{x} + \lambda \int_{\Omega} (f(\mathbf{x}) - u(\mathbf{x}))^2 d\mathbf{x} \right\} \quad (2.90)$$

where $u \in BV$.

Evidently, the ROF model (enclosed in braces) is composed of a TV term and a fidelity term. This model has been cited in more than six thousand papers.

The TV term removes noise or small details, while preserving edges. The authors of the ROF model give in [163] an algorithm for computing the adequate value of λ if the noise level was known. In other cases this value has to be chosen, considering that it determines in some sense the smallest image feature to be kept.

From the point of view of optimization, the good news is that the ROF denoising problem is convex, so the solution exists in BV and is unique [164]. A detailed study of image recovery via TV minimization is [47]. The field of TV in imaging, including algorithms, is reviewed by [45].

There are a number of observed problems when adhering to the ROF approach, as described in [48]: loss of contrast, loss of geometry, staircasing, and loss of texture. Part of the recent developments cited in [48] are oriented to solve these problems.

The ROF variational method can be adapted to different types of noise, as introduced in [90]. This paper is plenty of practical numerical and analytical details, and contains a link to one implementation in C code.

Next section includes an example of ROF-TV image denoising accompanied with a MATLAB program.

2.4.1.4 Meyer's Approach

Meyer suggested to replace the l_2 norm in the fidelity term with a weaker norm more adequate for modeling textures or oscillatory patterns. Hence, he proposed the following minimization problem:

$$\inf_u \left\{ \int_{\Omega} |\nabla u(\mathbf{x})| d\mathbf{x} + \lambda \|f(\mathbf{x}) - u(\mathbf{x})\|_* \right\} \quad (2.91)$$

where $u \in BV$.

Continuing with his approach, Meyer defined the space G , which is the Banach space of all generalized functions v that can be written as $v = \text{div}(\mathbf{g})$, where $\mathbf{g} = (g_1, g_2)$ and $g_1, g_2 \in l_{\infty}(\Omega)$. The space G is endowed with the G -norm, which is defined as the lower bound of all $l_{\infty}(\Omega)$ norms of the functions $|\mathbf{g}|$, with the infimum being computed over all decompositions of v .

The space G is the dual of the closed subspace BV of BV . When applying the G -norm in the minimization problem, the second term is $\lambda \|f(\mathbf{x}) - u(\mathbf{x})\|_G$. Meyer also defined two more spaces: E and F , see [128]. The spaces are related as follows: $BV \subset G \subset E \subset F$.

G -functions may have large oscillations and nevertheless small norms, which is suitable for the minimization to preserve textures.

Meyer did not propose any numerical procedure for the decomposition. A first algorithmic contribution to this aim was made in [180] (the Vese-Osher model), which soon was followed by other proposals, like [10, 193]. In [29] a simple conversion of a linear filter pair into a nonlinear filter pair was proposed, obtaining a fast separation of cartoon and texture.

2.4.1.5 TV- l_1 Approach

It was suggested in [135] to replace the l_2 term in the ROF model by a l_1 term. This article, year 2004, contains interesting references from the 90s about the fidelity term and how to avoid outliers. According with this approach, the functional to minimize is:

$$\inf_u \left\{ \int_{\Omega} |\nabla u(\mathbf{x})| d\mathbf{x} + \lambda \int_{|f(\mathbf{x}) - u(\mathbf{x})|} d\mathbf{x} \right\} \quad (2.92)$$

As shown in [135], the l_1 norm is well suited to remove salt and pepper noise. Further analysis by [49, 193], shows that the model enjoys interesting properties of morphological invariance and texture extraction by scale, so geometrical features are better preserved. A first, fast algorithm for solving the optimization problem was presented in [12].

All three models, Meyer, Vese-Osher, and TV- l_1 , were compared by [194] using a uniform computing approach. The three models were solved as second-order cone programs (SCOP). The comparison refers to 1D signals and 2D images. Also, [194] contains a detailed history, with references, of alternatives for solving the three models.

See [109] for a practical treatment of the TV-/1 decomposition, including various examples and a link to software from a web site.

2.4.1.6 Other Approaches for the Texture Term

The Report [11] proposed the following generalization:

$$\inf_u \left\{ \int_{\Omega} |\nabla u(\mathbf{x})| \, d\mathbf{x} + \lambda \|f(\mathbf{x}) - u(\mathbf{x})\|_H^2 \right\} \quad (2.93)$$

where H is some Hilbert space. This generalization can include a number of different models, and in particular the ROF model. One of the contributions of [12] is a Hilbert space of Gabor wavelets.

From time ago there was interest on texture modeling and analysis for different purposes. A brief review of invariant texture analysis methods is offered in [198]. With the advent of the cartoon-texture approach, more research has been devoted to texture models favoring better image decompositions; see [125] for a modern perspective involving a decomposition of the functional into three terms: the fidelity term, a cartoon term, and a texture term.

2.4.1.7 Other Approaches for the Cartoon Term

As said before, it has been observed that the TV term induces image staircasing effects. A remedy for this problem was introduced in [47], by including higher order derivatives in the energy.

Several versions of the cartoon-texture decompositions were proposed in [50], by combining the improved cartoon term of [47] and three alternatives for the texture term (the third alternative considers texture + noise). This article is particularly interesting in several ways: discussion, formulae, and experimental results.

See [110] for an interesting work on second order TV, and [121] for combining TV and a fourth-order partial derivative filter.

2.4.1.8 Some Related Aspects

Let us briefly collect a number of references that deal with important aspects of the methods already introduced.

- A correlation tool was introduced in [12] to properly select the parameter λ .
- Relevant analysis results concerning BV functions are presented in [2, 58]. By the way, in [96] it was provocatively questioned if natural images are BV.
- The decomposition into cartoon + texture + noise ($u + v + w$) was introduced by [92].
- Almost every cited paper includes a review of the decomposition topic. In addition, the reader could consult [29, 93] for more extensive reviews.

2.4.2 Patches

Science has surprising connections between seemingly disparate fields. For instance, it happens that the term “*sparse coding*” also belongs, from the 90s, to Neuroscience research.

As it will become clear soon, this biological aspect deserves some attention now. A convenient guide is offered by the review done in [171], in special connection with the work of Olshausen. Many references in [171] quote observations and conjectures made by Barlow in the 60s.

2.4.2.1 A Bit of Neural Processing

Images are captured by the retina, transmitted to the LGN, and then to the area V1 of the visual cortex; subsequent areas are V2, V4, MT, and MST. It has been reported that nobody has been able to reconstruct the input image from the recordings of neurons in V1. Cells in the visual cortex were classified as simple, complex, and hyper-complex cells. Simple and complex cells are sensitive to specific stimuli orientations.

One of the Barlow’s hypotheses is that the role of early sensory neurons is to remove statistical redundancy of the input. Hence, it is not strange that the first section of [171] was devoted to PCA and ICA. This also corresponds to an important direction of the research, which assumes that sensory neurons are specially adapted to the statistical properties of those signals that occur more frequently. So, it is important to investigate the relationship between natural signals and neural processing. In general, natural images are not Gaussian and there are significant spatial correlations.

Another observation of Barlow was that neurons at later stages of processing are generally less active than those at earlier stages. It seems that we may model what happens in visual areas using multiple stages of *efficient coding*. This is, indeed, an ambitious objective. Most initial steps of the research have focused on retina and the first visual cortex areas.

Concerning the retina, it has been shown that the single-cell physiology and contrast sensitivity functions are consistent with the product of a whitening filter and a Wiener filter for noise removal and adaptation to mean luminance level (see [171] and references therein). Because non-Gaussianity, a whitened natural image still has lines, edges, contours, etc.

It seems that there is efficient coding at the retina level. The next question is if we have this in V1.

In a famous letter to *Nature* in 1996, [138] proposed to represent an image as a linear superposition of 2D basis functions, which are *image patches extracted from natural scenes*. The representation has a conventional form:

$$I(x, y) = \sum_i a_i \phi_i(x, y) \quad (2.94)$$

Fig. 2.19 A patch dictionary

The set of basis functions emerged after training on many (in the order of 10^5) image patches randomly extracted from natural scenes. The training tried to maximize the sparsity of the representation, searching for components that are both sparse and statistically independent (in the ICA sense). Actually, [138] shows an example of 192 basis functions, which are 16×16 pixel patches, extracted from 512×512 natural images. These functions resemble the spatial receptive field properties of simple cells, they are spatially localized, oriented, and band-pass.

Figure 2.19 shows an example of a patch dictionary, with 256 patches of 8×8 pixel each.

In 1997, [139] made the hypothesis that V1 employs sparse coding with an over-complete basis set.

The principle behind this kind of coding is that it tries to represent each image in terms of a small set of functions, chosen from an overcomplete set. Only a few neurons need to be active and expend energy. Actually, it has been estimated that at any given moment only 1/50th of the cortical neurons could afford to be active, due to energy constraints (see [140] and references therein).

It is being found that sparse coding is also employed by other senses and other neural functions [140]. Part of current research is considering space-time statistics, using natural environment movies as inputs [31, 171].

2.4.2.2 Dictionaries for Sparse Representation

Back to our signal processing atmosphere, the lessons learned from neural processing are summarized in [162] in the context of a history of transform design that includes Fourier, wavelets, etc. This summary put in contrast analytic versus trained dictionaries. Analytic dictionaries are linked to harmonic analysis, use pre-defined classes of functions, and are usually too simplistic for the nature complexity. Machine learning

assumes, instead, that the structure of complex phenomena can be more accurately extracted directly from the data.

Of course, if one adheres to the use of learned dictionaries, then a training method should be devised. In his review, [161] identifies five available methods. Perhaps the most popular of them is K-SVD, which will be described in more detail below. The other methods are: generalized PCA, union of orthobases, the method of optimal directions (MOD), and parametric training methods. References are provided for all methods. The advantage of parametric training is that structured dictionaries were obtained.

Given a set of training column data vectors, $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N$, the problem is to find a $(m \times K)$ dictionary D , with $K \ll N$, such every \mathbf{y}_i is a sparse combination of elements of D . Note that the data vectors \mathbf{y}_i could contain 1D signals or vectorized image patches.

Figure 2.20 shows a diagram corresponding to the problem. The representation matrix X should be sparse (with sparsity s).

In mathematical terms, the problem to solve is:

$$\min_{D, X} \sum_i \|\mathbf{y}_i - D \mathbf{x}_i\|^2, \text{ subject to } \|\mathbf{x}_i\|_0 \leq s$$

Notice that the optimization is over both D and X .

The problem could be relaxed from l_0 to l_1 norm, and be stated in Lagrangian form adding two terms.

Some authors prefer to use a simpler expression for the term $\sum_i \|\mathbf{y}_i - D \mathbf{x}_i\|^2$, as follows:

$$\|Y - DX\|_F^2 \quad (2.95)$$

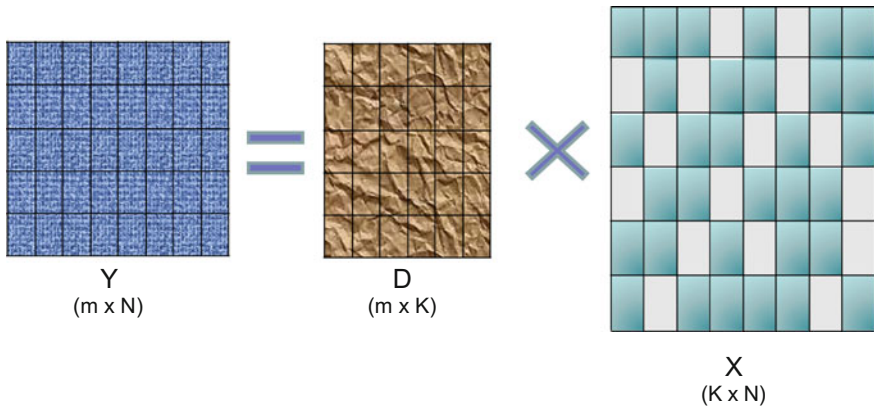


Fig. 2.20 The dictionary problem

There is a general iterative scheme that can be used for solving the problem [44]. Each iteration includes two steps:

- Sparse coding, for finding a minimizing X (for a fixed D)
- Dictionary update, finding a minimizing D (for a fixed X)

Usually this scheme would lead to a local-minimum.

The sparse coding part could be done with methods already introduced in this chapter, like for instance orthogonal matching pursuit (OMP).

The dictionary update could be done by directly solving the least square problem, using the Moore-Penrose pseudo-inverse. This is the alternative chosen by the MOD method; but it tends to lead the iteration towards the local minimum nearest to the initial guess. Other alternative could be to use a gradient descent on D .

2.4.2.3 The K-SVD Training Method

The K-SVD algorithm is described in detail in [6], which is a most cited article. It can be considered as a generalization of the K-means clustering algorithm, already seen in the chapter on data classification.

As MOD and other methods, K-SVD iterates two steps: sparse coding and dictionary update. Sparse coding can be done by any suitable method; K-SVD focuses on the dictionary update. For this update, assume that both D and X are fixed and select one column \mathbf{d}_k of D ; then, select the k -th row of X , which will be denoted as \mathbf{x}_T^k . Now, following [6], let us derive a convenient expression:

$$\begin{aligned} \|Y - DX\|_F^2 &= \left\| Y - \sum_j \mathbf{d}_j \mathbf{x}_T^j \right\|_F^2 = \left\| \left(Y - \sum_{j \neq k} \mathbf{d}_j \mathbf{x}_T^j \right) - \mathbf{d}_k \mathbf{x}_T^k \right\|_F^2 = \\ &= \|E_k - \mathbf{d}_k \mathbf{x}_T^k\|_F^2 \end{aligned} \quad (2.96)$$

where E_k is an error matrix.

Denote as ω_k the group of indices pointing to vectors \mathbf{y}_i that use \mathbf{d}_k (those where $\mathbf{x}_T^k(i)$ is nonzero). The group has a certain number L of indices.

Also, denote as Ω_k a matrix with ones on the $(\omega_k(i), i)$ entries, and zeros elsewhere. This matrix will be used for shrinking purposes. For instance, the result of $\mathbf{x}_R^k = \mathbf{x}_T^k \Omega_k$ is a row vector \mathbf{x}_R^k of length L , which is obtained from \mathbf{x}_T^k by discarding the zero entries. Similarly, $E_k^R = E_k \Omega_k$ is the set of error columns corresponding to examples that use \mathbf{d}_k .

Then, for the dictionary update one has to minimize:

$$\|E_k \Omega_k - \mathbf{d}_k \mathbf{x}_T^k \Omega_k\|_F^2 = \|E_k^R - \mathbf{d}_k \mathbf{x}_R^k\|_F^2 \quad (2.97)$$

Here, one can use SVD (singular value decomposition) for getting the desired solution. The matrix E_k^R is decomposed as $E_k^R = U \Delta V^T$. The solution for \mathbf{d}_k is the first column of U , and for the vector \mathbf{x}_R^k , the first column of V multiplied by the singular value $\Delta(1, 1)$.

Therefore, the algorithm takes the following steps:

-
- Set an initial dictionary with l_2 normalized columns
 - $j = 1$
 - (*) repeat until convergence
 - *Sparse coding step*
 - *Dictionary update step*: for each column $k = 1, 2, \dots$ in D
 - Find ω_k , build Ω_k
 - Compute E_k , and E_k^R
 - Decompose E_k^R using SVD
 - Update D with the column \tilde{d}_k and X with the vector \mathbf{x}_R^k
 - $j = j + 1$, go to (*)
-

See [52] for a longer explanation of the K-SVD method.

It is convenient to note that part of the specialized literature use the term *atom* to refer to columns, \mathbf{d}_k , of the dictionary D .

A main contribution of K-SVD is that the dictionary update does not require matrix inversion; the update is made atom-by-atom, sequentially, in an efficient manner. Another difference with other algorithms is that during dictionary update, the content of X is also modified. The algorithm reduces, or at worst maintains, the error of the representation at each iteration.

A more efficient implementation of K-SVD was introduced in [161], with links to MATLAB toolboxes.

Next three figures correspond to an example of image denoising using K-SVD. Figure 2.21 depicts the original image and the same image contaminated with Gaussian noise. Figure 2.22 shows the patch dictionary obtained with the K-SVD method. Then, using this dictionary the image is denoised, as described in [78], and the result is shown in Fig. 2.23. These figures have been obtained with a program based on the software available from the Elad's web site. Although this program is a simplified version, it is relatively long and so it has been included in the appendix of long programs. The program is not only long, but also relatively slow: it would probably take around 5 min. The reader is invited to add more K-SVD iterations, or to specify less error (the constant C at the beginning of the program), or to increase the number of blocks.

The program uses two times a sparse codification function that has also been included in that appendix. This function uses sparse matrices.

2.4.2.4 Further Developments

The methods already described, like MOD or K-SVD, learn a dictionary D to sparsely represent the patches of an image, rather than the whole image itself [189]. In an important article, [78] a proposal was made in the context of image denoising.

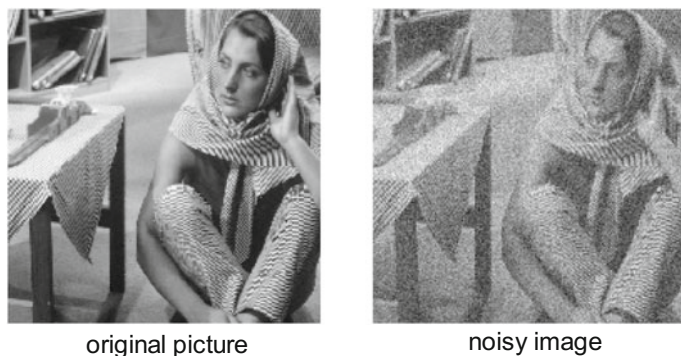
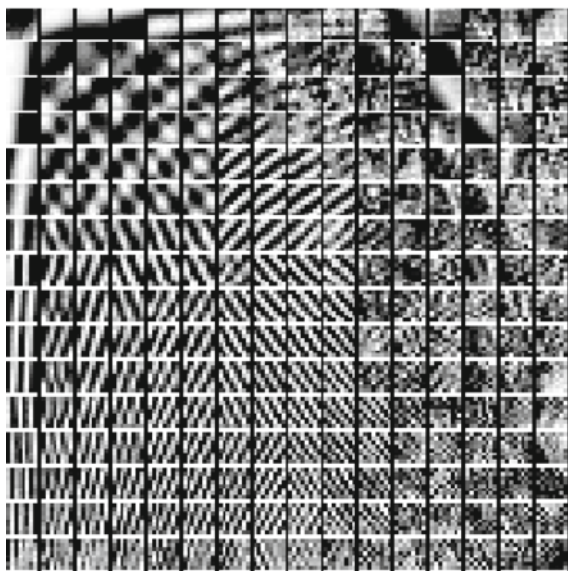


Fig. 2.21 Original picture, and image with added Gaussian noise

Fig. 2.22 Patch dictionary obtained with K-SVD



Overlapping patches were used. The idea was to denoise each patch via sparse coding, and then estimate the total image as the average of the patches together with the observed noisy image. Actually, [78] adopted a Bayesian perspective, by defining a global image prior that forces sparsity over all patches. Further elaboration in this line was presented in [77, 79].

In the case of denoising and other applications (impainting, deblurring, etc.) it is natural to consider a Bayesian treatment. If I see a noisy or corrupted image I would say: this image is not clean. So, I expected something cleaner. In consequence, I have a kind of *model* (a prior) of what an image should be.

See [189] for a fast method for whole image recovery using patch-dictionary. It has an associated web page with MATLAB code.

Fig. 2.23 Denoised image

Responding to the current consumer market needs, it is important to deal with colour images. This was the target of [122], which extends the approach of [78] to colour images. The proposed scheme was also able to properly handle non-homogeneous noise; this is valuable in cases of missing data, such in image demosaicing or inpainting (colloquially: filling holes).

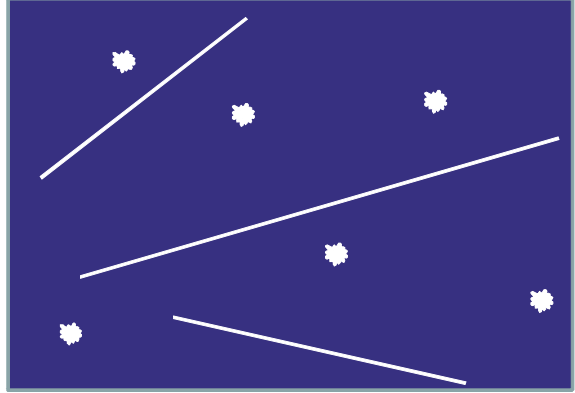
According with [155], the use of patch dictionaries has become very popular, showing good performances. The research is trying to exploit interrelations between patches, using for instance clustering into disjoint sets to treat them adequately, etc. ([155] includes a short review of this issue).

In his interesting discussion on K-SVD, [16] remarks that a good quality dictionary for sparse coding should have a small mutual coherence constant μ . However, the μ of dictionaries obtained by K-SVD (which are highly redundant) is usually not small. Another problem is that in very redundant dictionaries, some atoms might be highly similar to others, or play an insignificant role. These arguments are used by [16] to promote, as better approach, orthogonal dictionaries with small μ . This paper also includes a convenient review.

2.4.3 Morphological Components

Consider the example depicted in Fig. 2.24, which is a synthetic image that may be similar to an astronomical picture (stars and filaments).

The stars could be well represented using wavelets, while for the lines ridgelets are more suitable. Then, it seems appropriate to use two dictionaries, wavelets and ridgelets, for representing the image. A decomposition of the image into two components would be possible, one component with the stars and the other with the lines.

Fig. 2.24 A synthetic image

Although simplistic, this example illustrates well the idea of *morphological component analysis* (MCA), [172, 173]. As summarized in [24], the MCA method relies on an iterative thresholding algorithm, with a threshold that decreases linearly along iterations. Let us describe the algorithm, based on [24].

The case considered is a signal consisting of a sum of K signals \mathbf{y}_i , having different morphologies. A dictionary of bases $\{\Phi_1, \Phi_2, \dots, \Phi_K\}$ is assumed to exist. Signal \mathbf{y}_1 is sparse in Φ_1 ; signal \mathbf{y}_2 is sparse in Φ_2 ; and so on. Denote $\alpha_i = \Phi_i^T \mathbf{y}_i$.

Suppose, for simplicity, that one has only two signals ($K = 2$), so $\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2$ (the results can be easily generalized to more components). It is proposed, in order to estimate the components of \mathbf{y} , to solve the following minimization:

$$\min_{\mathbf{y}_1, \mathbf{y}_2} (\|\Phi_1^T \mathbf{y}_1\|_1 + \|\Phi_2^T \mathbf{y}_2\|_1), \text{ subject to } \|\mathbf{y} - \mathbf{y}_1 - \mathbf{y}_2\|_2 \leq \sigma$$

where σ is the noise standard deviation. Continuing with a simplified view, assume for now that $\sigma = 0$.

The first step of the MCA algorithm sets the number of iterations I_{\max} , the minimum threshold λ_{\min} , initial estimated values of \mathbf{y}_1 and \mathbf{y}_2 , and the thresholds $\lambda_1^{(1)}$ and $\lambda_2^{(1)}$.

Then, iterations begin:

While the two thresholds are higher than a lower bound λ_{\min} , do:

- $k = 2$
- Compute residuals, for $j = 1, 2$, using current estimates $\tilde{\mathbf{y}}_i^{(k-1)}$ of the components:

$$\mathbf{r}_j^{(k)} = \mathbf{y} - \tilde{\mathbf{y}}_{i \neq j}^{(k-1)} \quad (2.98)$$

- Estimate, for $j = 1, 2$, current coefficients by hard thresholding:

$$\tilde{\alpha}_j^{(k)} = \delta_T (\Phi_j^T r_j^{(k)}) ; T = \lambda_j^{(k)} \quad (2.99)$$

- Get, for $j = 1, 2$, new estimates $\tilde{y}_j^{(k)}$:

$$\tilde{y}_j^{(k)} = \Phi_j \tilde{\alpha}_j^{(k)} \quad (2.100)$$

- Decrease the thresholds

$$\lambda_j^{(k+1)} = \lambda_j^{(1)} - k \frac{\lambda_j^{(1)} - \lambda_{\min}}{I_{\max} - 1} \quad (2.101)$$

If there is no noise, λ_{\min} should be set to zero. On the other hand, when there is noise, λ_{\min} should be set to a few times σ (see [24]).

The MCA algorithm provides a good components separation when the Φ_i , the members of the dictionary, are mutually incoherent enough. In the examples provided by [173], textures are treated with DCT (discrete cosine transform) since DCT is suitable for the representation of natural periodicity (in case of non-homogeneous textures, local DCT could be used). As said before, lines are well represented with ridgelets. In addition, the curvelet transform represents well edges in images.

One of the contributions of [24] with respect to the original MCA, is a method called ‘*mean-of-max*’ (MOM) for the decrease of thresholds. Denote:

$$m_j = \|\Phi_j^T r^{(k)}\| ,$$

with: $r^{(k)} = \mathbf{y} - \mathbf{y}_1^{(k-1)} - \mathbf{y}_2^{(k-1)}$

Then, thresholds should be chosen as:

$$\lambda_j^{(k)} = \frac{1}{2} (m_1 + m_2) \quad (2.102)$$

It is also shown in [24] that MCA/MOM is clearly faster than BP (basis pursuit), being at least as efficient as BP in achieving sparse decompositions in redundant dictionaries.

See the book [174] for an extensive treatment of sparse representation and processing, including chapters on wavelets, ridgelets, curvelets, etc. In particular, the chapter 18 of that book focuses on MCA, using the MCALab package (in MATLAB) by the same authors. Several interesting application examples were presented.

A simple 1D example of MCA processing is the case of a signal composed of two sine signals with close frequencies, and three spikes at random positions. The target of MCA is to separate the signal morphological components. We chose this example and prepared a simplified version of MCALab for this case. Only two dictionaries were considered, one based on discrete cosine transform (DCT), and the other for

Dirac pulses. The simplified version is the Program 2.7. In this program, the initial value of σ was estimated using the finest details obtained by Daubechies wavelet.

During the execution of the program, the selection process is visualized with an animated figure. Figure 2.25 shows an example. When the program stops, another figure is shown with the original signal and its components (Fig. 2.26). Then, the user can see how good was the MCA work, by comparing the original and the separation result.

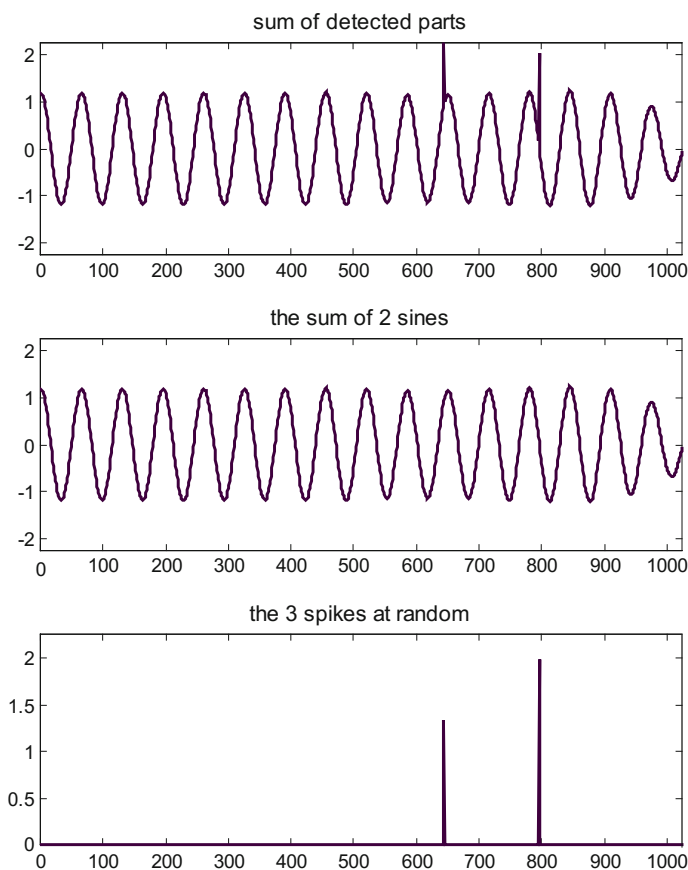


Fig. 2.25 Example of figure during MCA process

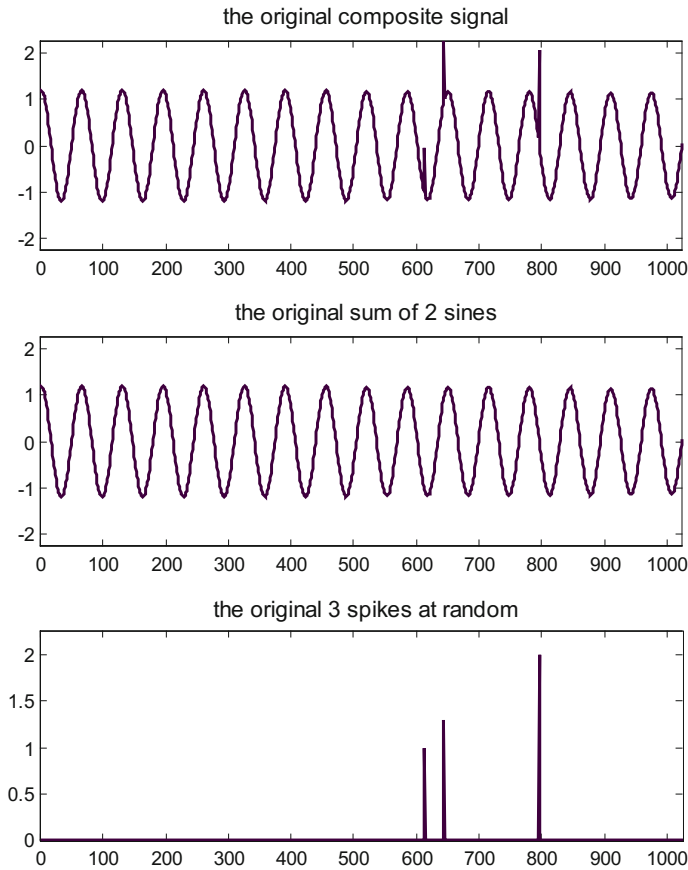


Fig. 2.26 The original composite signal and its components

Program 2.7 MCA example

```
% MCA example
% The signal is a mix of 2 sines and 3 random spikes
% signal synthesis -----
N=1024; %length
% sines have almost equal frequencies:
A=0.6; %sine amplitude
itv=1;
t=0:itv:N-itv;
s1=A*cos(pi*0.0307*t);
s2=A*cos(pi*0.0309*t);
ys=s1+s2; %sum of sines
% 3 spikes at random positions:
k=3;
v=[randn(k,1); zeros(N-k,1)]; p=randperm(N);
yk=v(p); pos=find(yk);
miny=min(yk); maxy=max(yk);
yk(pos) = (yk(pos)-miny)/(maxy-miny)+1; %normalize
```

```

% the composite signal
y = ys + yk';
signal=y(:); %column format
maxp=max(signal); %to set vertical limit in figures
nfg=0; %figure interval counter
% MCA parameters
qq=4; %to specify how fine is the approx
itermax = 200;
ndict=2; %number of dictionaries
%-----
%Initial variables
part= zeros(N,ndict);
%Initial sigma-----
[Wc,Wl]=wavedec(signal,10,'db4'); %wavelet toolbox
L=sum(Wl(1:10)); %to select details
Dv=-Wc(L+1:end);
% MAD method
aux=Dv(find(~isnan(Dv)));
sigma = median(abs(aux - median(aux)))./0.6745;
%Initial coeffs (using two dictionaries: DCT and Dirac)
coeff={};
% DCT coeffs:
nv=N*qq; fq=(1:(nv-1))';
Ko=[N; 0.5*(N+sin(2*pi*fq/qq)./(2*sin(pi*fq/nv)))];
Ko=Ko.^0.5;
x=zeros(4*nv,1); L=2*N;
x(2:2:L)=signal(:);
z=fft(x);
coeff{1}= [struct('coeff',[]) struct('coeff',real(z(1:nv))./Ko)];
% Dirac coeffs:
coeff{2}= [struct('coeff',[]) struct('coeff', signal(:))];
%Initial threshold (minimum of maximal coeffs in each dictionary)
for nn=1:ndict
    aux = []; cfs = coeff{nn};
    sinx = length(cfs);
    for j = 2:sinx
        aux = [aux;cfs(j).coeff(:)];
    end
    buf(nn)=max(abs(aux(:)));
end
buf=flipud(sort(buf(:),1))';
deltamax=buf(2);
delta=deltamax;
lambda=delta/(itermax-1); %Linear decrease: slope
%Start the algorithm-----
for iter=0:itermax-1
    %residual computation
    residual=signal-sum(part,2);
    % DCT part-----
    Ra=part(:,1)+residual;

```

```

%analysis:
x=zeros(4*nv,1); L=2*N;
x(2:2:L)=Ra(:); z=fft(x);
Ca= [struct('coeff',[]) struct('coeff',real(z(1:nv))./Ko)];
%thresholding (not the low frequency components):
cf = Ca; ay=cf(2).coeff;
cf(2).coeff=ay.*(abs(ay)>delta);
Ca = cf;
%synthesis:
c=Ca(2).coeff; lc=length(c); M=lc/qq;
fu=(1:(lc-1))';
Ku=[M; 0.5*(M+sin(2*pi*fu/qq)./(2*sin(pi*fu/lc)))];
Ku=Ku.^0.5;
c=C./Ku;
x=zeros(4*lc,1); L=2*M;
x(1:lc)=c; z=fft(x);
y=real(z(2:2:L));
part(:,1)=y(:)/qq; %output
% Dirac part-----
Ra=part(:,2)+residual;
%analysis:
Ca= [struct('coeff',[]) struct('coeff', Ra(:))];
%thresholding (not the low frequency components):
cf = Ca; ay=cf(2).coeff;
cf(2).coeff=ay.*(abs(ay)>delta);
Ca = cf;
%synthesis:
part(:,2)=Ca(2).coeff(:); %output
% Update parameters-----
delta=delta-lambda; %linear decrease
% Display along the process
nfg=nfg+1;
if nfg==4,
    nfg=0; %restart counter
    figure(1)
    subplot(3,1,1)
    plot(sum(part(1:N,:),2));axis tight;drawnow;
    title('sum of detected parts')
    axis([0 N -maxp maxp]);
    subplot(3,1,2)
    plot(part(1:N,1));axis tight;drawnow;
    title('the sum of 2 sines')
    axis([0 N -maxp maxp]);
    subplot(3,1,3)
    plot(part(1:N,2));axis tight;drawnow;
    title('the 3 spikes at random')
    axis([0 N 0 maxp]);
end
end

```



```

part = part(1:N,:);
% Final display-----
%Original signals
figure(2)
subplot(3,1,1)
plot(signal);axis tight;
title('the original composite signal')
axis([0 N -maxp maxp]);
subplot(3,1,2)
plot(ys);axis tight;
title('the original sum of 2 sines')
axis([0 N -maxp maxp]);
subplot(3,1,3)
plot(yk);axis tight;
title('the original 3 spikes at random')
axis([0 N 0 maxp]);

```

2.5 An Additional Repertory of Applicable Concepts and Tools

Some of the topics to be treated in this chapter require new concepts and tools. For instance, the functions provided by MATLAB for sparse calculus, and concepts and solution techniques related to certain optimization or signal processing problems. In particular, it has been found that the Bregman algorithms are very suitable for optimal compressed sensing and other applications.

2.5.1 Sparse Representation in MATLAB

There are two matrix storage modes in MATLAB. *Full* storage is the default and stores the value of each element. *Sparse* storage –that should be explicitly invoked– stores only the values of nonzero elements.

The function `sparse()` can be used to create a sparse matrix. For example, if you write:

$$S = \text{sparse}([], [], [], 1000, 1000);$$

An empty sparse matrix S is created. Then, you can specify some nonzero elements, for instance:

$$S(5, 3) = 15; S(312, 1) = 120.5; S(25, 25) = 1;$$

The function `nnz(B)` returns the number of nonzero elements of the matrix B , which can be in full or in sparse format. The function `find(B)` returns all (i, j) indices of nonzero elements. The function `nonzeros(B)` returns all the nonzero elements.

As with full matrices, with sparse matrices you also can use the expression:

$$x = A \setminus b$$

In many applications is more convenient the use of sparse matrices than full matrices. For instance, in case of having a $10,000 \times 10,000$ matrix with 20,000 nonzero elements. Solving $Ax = b$ will be much faster with A sparse matrix than with A full matrix.

- A full matrix B can be converted to sparse with $S = \text{sparse}(B)$. A sparse matrix S can be converted to full with $B = \text{full}(S)$.
- The function `spdiags()` can be used to create sparse banded matrices, which have a few nonzero diagonals.
- The identity matrix is created with `speye()`. Given a sparse matrix S , the function `spones()` replaces the nonzero elements with ones.
- A simple visualization of the nonzero elements localization in the matrix S can be obtained with `spy(S)`.

Many more details of the use of sparse matrices in MATLAB can be found in [91].

As a first, simple example, the Program 2.8 creates a tri-banded sparse matrix, that you can list using `full(A)` without semicolon, and then applies `spy()` to display (Fig. 2.27) the matrix structure (non-zero entries).

Program 2.8 Create a tri-banded sparse matrix

```
%Create a tri-banded sparse matrix
%
A=spdiags([2 1 0; 5 4 3; 8 7 6; 11 10 9; 0 13 12],-1:1,5,5);
full(A); %list the matrix in usual format
figure(1)
spy(A); %visualize the matrix structure (non-zero entries)
%
title('spy diagram');
```

Fig. 2.27 Visualization of banded matrix using `spy()`

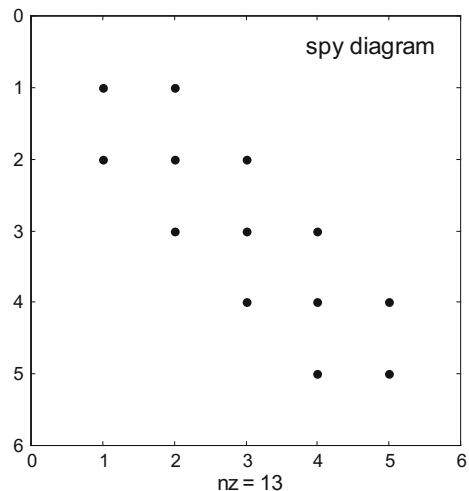
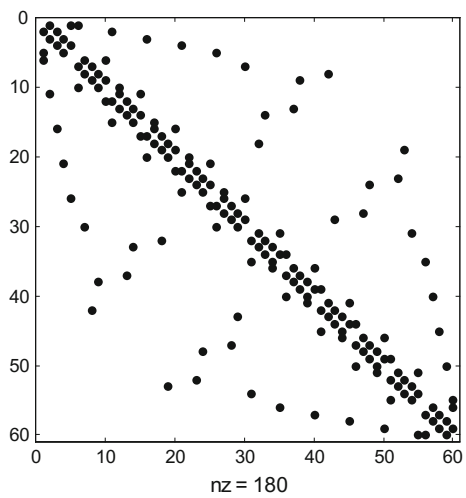


Fig. 2.28 Visualization of the Bucky ball matrix structure



Since it is a popular example, MATLAB includes data for the Bucky ball, which is composed of 60 points distributed as in a soccer ball. Each point has three neighbours. The Bucky ball models the geodesic dome made popular by Buckminster Fuller, and also the C60 molecule (a carbon molecule with 60 atoms). The Bucky ball adjacency matrix is a 60×60 symmetric matrix.

Figure 2.28 shows the `spy()` visualization of the Bucky ball matrix.

From the data given by MATLAB it is also possible to get coordinates, and to plot with `gplot()` the Bucky ball. Both Figs. 2.28 and 2.29 were generated with the Program 2.9.

Fig. 2.29 Visualization of the bucky ball graph

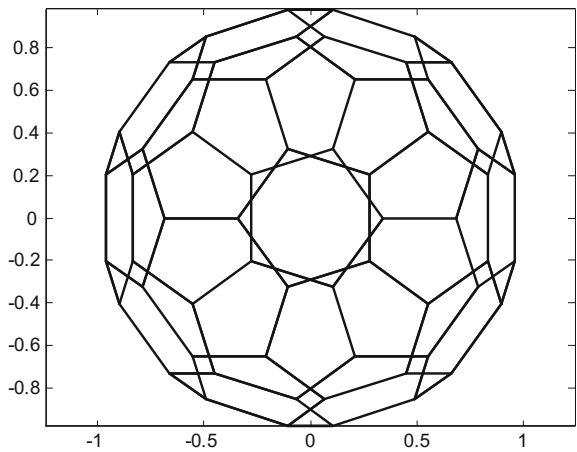
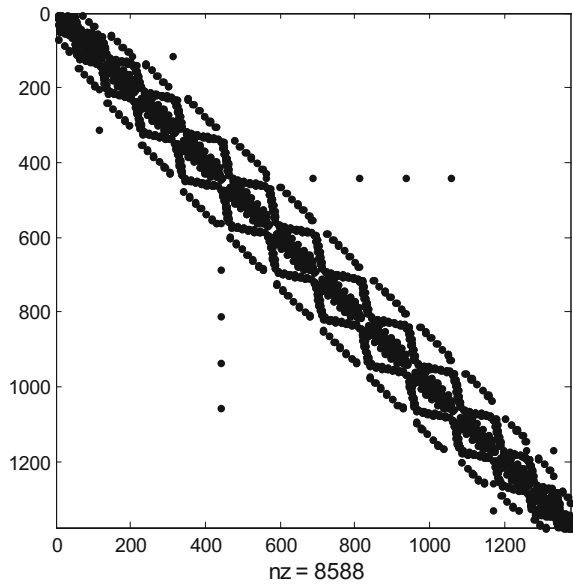


Fig. 2.30 Visualization of HB/nnc1374 matrix using `spy()`



Program 2.9 Load and visualize the bucky matrix

```
%Load and visualize the bucky matrix
%
A=bucky; %rapid loading of the 60x60 bucky matrix
figure(1)
spy(A, 'k');
title('the bucky ball matrix');
%
[A,v]=bucky; %load the bucky matrix and the coordinates
figure(2)
gplot(A,v, 'k');
axis equal;
title('the bucky ball graph');
```

Among the sources of data on Internet, there are two important collections of sparse matrices, one in the ‘Matrix Market’ and the other in the University of Florida Sparse Matrix Collection. From this second source, we chose the matrix HB/nnc1374 as an example. It is a matrix with 1374×1374 entries, and only 8588 non-zero entries. Notice that earlier versions of MATLAB cannot handle such a large matrix.

Figure 2.30 displays the `spy()` diagram of this matrix. The figure has been generated with the Program 2.10, which shows how to extract the matrix from the data structure downloaded from the Internet repository.

Program 2.10 Load and visualize HB nnc1374

```
%Load and visualize HB nnc1374
%(note: earlier MATLAB versions cannot handle the large matrix)
%
S=load('nnc1374.mat'); %file (structure) loading
fieldnames(S) %to see structure field names
getfield(S, 'Problem') %to see inside the field 'Problem'
M=getfield(S, 'Problem', 'A'); %get the sparse matrix from 'A'
figure(1)
spy(M, 'k');
title('the HB nnc1374 matrix');
```

2.5.2 Diffusion in 2D

Let us begin with a kind of strange example. Suppose you want to paint your inclined roof. You could come to the top edge and just drop the paint, letting the colour flow down. After some evolution time, you could expect some uniformity. Of course, this is not a recommended method, but it gives you the flavour of what will be introduced next for denoising, inpainting and other applications. Notice that the colour flow will be mostly unidirectional as governed by the slope.

Another example would be the following: you take a thin aluminium plate, put a flame below and near the centre during some time, and observe how the heat flows from the centre toward the borders of the plate. The heat diffusion will be omnidirectional.

As will be seen next, the heat diffusion can be related with 2D Gaussian filtering, considering the image intensity as analogous to energy. The image filtering would take some time (some iterations), to let the diffusion evolve.

In the case of denoising, one wants to eliminate the noise using diffusion, but, at the same time, one wants to preserve edges. Is this possible? This also will be treated next.

Since diffusion directions would become important, the proper mathematical tool should be partial differential equations (PDE). Let us advance some important basic equations concerning diffusion and heat.

Denote $\phi(x, y, t)$ the density of the diffusing substance at a given position and time. The diffusion equation is:

$$\frac{\partial \phi}{\partial t} = \nabla [D(\phi, x, y, t) \cdot \nabla \phi] \quad (2.103)$$

where $D()$ is the *diffusion coefficient* (or diffusivity). This coefficient could be a scalar, a scalar function of coordinates (non-homogeneous diffusion), or a tensor (which could correspond to anisotropic diffusion). In 2D this tensor is a symmetric positive definite matrix. The equation becomes non-linear if the coefficient depends on ϕ .

The heat equation is obtained with a $D()$ being a constant:

$$\frac{\partial \phi}{\partial t} = D \cdot \nabla^2 \phi \quad (2.104)$$

In order to discretize the equation for computing the heat diffusion on a grid, one could approximate the second derivatives as follows:

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi(x_{i+1}, y_j) - 2\phi(x_i, y_j) + \phi(x_{i-1}, y_j)}{(dx)^2} \quad (2.105)$$

$$\frac{\partial^2 \phi}{\partial y^2} = \frac{\phi(x_i, y_{j+1}) - 2\phi(x_i, y_j) + \phi(x_i, y_{j-1})}{(dy)^2} \quad (2.106)$$

These expressions are to be introduced in: $\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}$

For the first derivative, one could use a first-order approximation:

$$\frac{\partial \phi}{\partial t} = \frac{\phi(t + dt) - \phi(t)}{dt} \quad (2.107)$$

A simple example has been devised, in which a central region of a plate is heated, and then the heat diffusion takes place along time. The computation of the diffusion on a grid has been done with the Program 2.11. For a simpler notation, the heat has been denoted as u . Figure 2.31 shows a sequence of plots, from left to right and top to bottom, corresponding to the process evolution. The program execution may take several minutes.

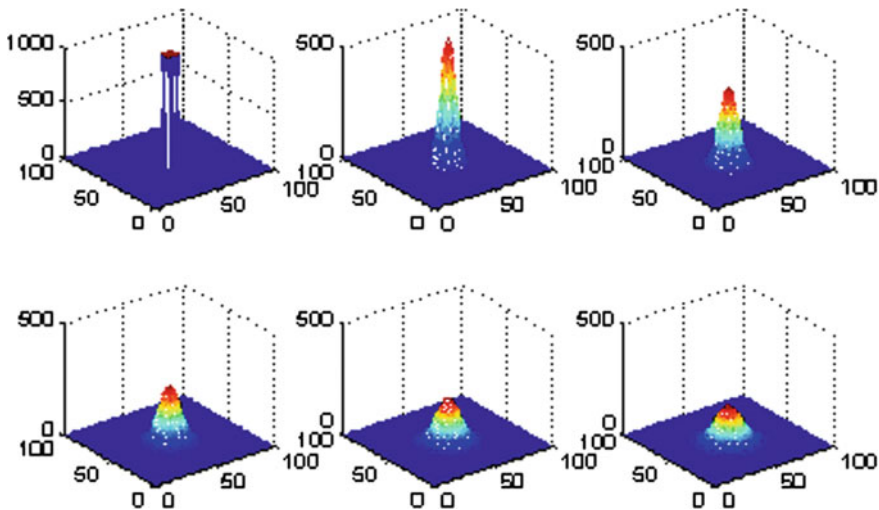


Fig. 2.31 Visualization of heat diffusion example

Program 2.11 Example of heat diffusion

```

%Example of heat diffusion
%
%Initialization of variables
D=0.25; %diffusion constant
dx=1; dy=dx; %we will use a regular grid, with dy=dx
dt=1;
x=0:dx:99; y=0:dy:99; %the grid
lx=length(x); ly=length(y);
u=zeros(lx,ly); un=u;
u(46:54,46:54)=1000; %central region is heated
%first display
figure(1)
subplot(2,3,1);
mesh(u);
axis([0 100 0 100 0 1000]);
title('heat diffusion');
for nn=2:6,
    for T=1:dt:30, %time
        for i=2:ly-1,
            for j=2:lx-1,
                aux=u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)-(4*u(i,j));
                un(i,j)=u(i,j)+((D*dt*aux)/dx^2);
            end;
        end;
        u=un;
    end
    subplot(2,3,nn); %the other plots
    mesh(un);
    axis([0 100 0 100 0 500]);
end;

```

When using the approximations already described, it is important to keep:

$$\frac{2 D \cdot dt}{\min((dx)^2, (dy)^2)} \leq 1 \quad (2.108)$$

otherwise, the numerical scheme becomes unstable (as you may want to check).

2.5.2.1 Gaussian Diffusion

In the case of heat diffusion with homogeneous Neumann boundary conditions and $D = 1$, one has the following problem:

$$\frac{\partial \phi}{\partial t} = \nabla^2 \phi \quad (2.109)$$

$$\phi(x, y, 0) = \phi_0 \quad (2.110)$$

$$\frac{\partial \phi}{\partial \mathbf{n}} = 0, \quad (x, y) \in b(\Omega) \quad (2.111)$$

where $b(\Omega)$ is the boundary of the region of interest Ω (usually a rectangle in the case of a picture), and \mathbf{n} is the normal to this boundary.

It has been established that the solution of this problem is given by the convolution of ϕ_0 and the Gaussian function with $\sigma = \sqrt{2t}$. In the Fourier domain, the solution is:

$$\Phi(\omega) = \exp\left(\frac{-|\omega|^2}{2/\sigma^2}\right) \Phi_0(\omega) \quad (2.112)$$

Therefore, the Gaussian diffusion is equivalent to a special low-pass filter. In the case of a picture, the effect would be image blurring. This blurring does not respect edges, so the structure is lost.

Figure 2.32 shows the effect of Gaussian diffusion on a picture. On top, the original picture; in the middle, the image after 10 s of diffusion; in the bottom, the image

Fig. 2.32 Effect of Gaussian diffusion, original on top



after another 10 s of diffusion. The processing has been done with the Program 2.12, which is similar to the previous program for heat diffusion.

Program 2.12 Example of picture diffusion blurring

```
%Example of picture diffusion blurring
% (like heat diffusion)
%
%Initialization of variables
D=0.25; %diffusion constant
dx=1; %we will use a regular grid, with dy=dx
dt=1;
x=0:dx:399; y=0:dy:249; %the grid (image size)
lx=length(x); ly=length(y);
P=imread('spencer.jpg'); %read image
u=double(P);
un=u;
%first display
figure(1)
subplot(3,1,1);
imshow(uint8(u));
title('Gaussian diffusion');
for nn=2:3,
    for T=1:dt:10, %time
        for i=2:ly-1,
            for j=2:lx-1,
                aux=u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)-(4*u(i,j));
                un(i,j)=u(i,j)+((D*dt*aux)/dx^2);
            end;
        end;
        u=un;
    end
    subplot(3,1,nn); %the other plots
    imshow(uint8(un));
end;
```

Although it may then easily become unstable, the diffusion process could be reversed in order to sharpen (or deblur) the image. Figure 2.33 shows an example, with the original picture on top and the sharpened image below. Notice in Program 2.13 that the diffusion constant and the diffusion time have been decreased.

Program 2.13 Example of picture anti-diffusion sharpening

```
%Example of picture anti-diffusion sharpening
% (reverse heat diffusion)
%
%Initialization of variables
D=0.05; %diffusion constant
dx=1; %we will use a regular grid, with dy=dx
dt=1;
```

```

x=0:dx:399; y=0:dy:249; %the grid (image size)
lx=length(x); ly=length(y);
P=imread('spencer.jpg'); %read image
u=double(P);
un=u;
%first display
figure(1)
subplot(2,1,1);
imshow(uint8(u));
title('Gaussian anti-diffusion');
for T=1:dt:5, %time
    for i=2:ly-1,
        for j=2:lx-1,
            aux=u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)-(4*u(i,j));
            un(i,j)=u(i,j)-((D*dt*aux)/dx^2);
        end;
    end;
    u=un;
end
subplot(2,1,2); %the other plot
imshow(uint8(un));

```

Fig. 2.33 Effect of Gaussian anti-diffusion, original on top



2.5.2.2 The Perona-Malik Diffusion

In 1987 Perona and Malik introduced a celebrated model [149], see also [150], that has been cited by more than eight thousand papers. The target was to protect, and even improve, the edges while smoothing more homogeneous regions of the picture.

The diffusion coefficient $D()$ will depend on the local image gradient. For small gradients, corresponding to homogeneous regions, large values of $D()$ are allowed, promoting stronger smoothing. On the other hand, for large gradients, corresponding to edges, smaller values of $D()$ are used to slow down the diffusion or even force the diffusion to go backwards.

Using now the notation commonly employed for images ($u()$ instead of $\phi()$), the Perona-Malik formulation of the diffusion becomes:

$$\frac{\partial u}{\partial t} = \nabla [D(|\nabla u|) \nabla u] \quad (2.113)$$

$$u(x, y, 0) = u_0 \quad (2.114)$$

$$\frac{\partial u}{\partial \mathbf{n}} = 0, \quad (x, y) \in b(\Omega) \quad (2.115)$$

Two different choices of $D(s)$ where suggested:

$$D(s) = \frac{1}{1 + s^2/\lambda^2} \quad (2.116)$$

$$D(s) = \exp(-s^2/\lambda^2) \quad (2.117)$$

(substitute s with ∇ where appropriate)

The *flux function* is defined as $\varphi(\nabla u) = [D(|\nabla u|) \nabla u]$. The derivative of this function tells you if there is forward diffusion, or backward diffusion. If you selected (2.116) for $D(s)$, then:

$$\varphi'(\nabla u) \begin{cases} < 0 \text{ if } |s| > \lambda \\ > 0 \text{ if } |s| < \lambda \end{cases} \quad (2.118)$$

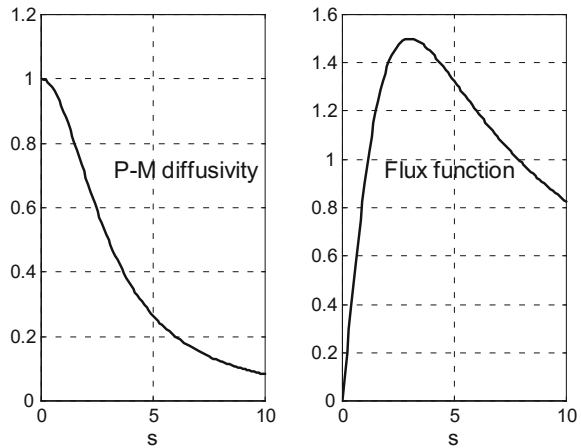
Therefore, near edges (large gradient) there is backward diffusion that will enhance these edges.

In the simple one-dimensional diffusion case, one has:

$$\frac{\partial u}{\partial t} = \varphi'(\nabla u) \Delta u \quad (2.119)$$

Figure 2.34 shows the one-dimensional diffusion coefficient and the flux function for $D(s)$ given by (2.116), with $\lambda = 3$.

Fig. 2.34 The diffusion coefficient and the corresponding flux function



Program 2.14 Perona-Malik diffusivity function and flux function

```
%Perona-Malik diffusivity function and flux function
% (first alternative)
%
lambda=3;
s=0:0.1:10; %variable values
g=1./(1+(s.^2/lambda^2));
f=s.*g;
figure(1)
subplot(1,2,1);
plot(s,g,'k');
xlabel('s'); grid;
title('P-M diffusivity');
axis([0 10 0 1.2]);
subplot(1,2,2);
plot(s,f,'k');
xlabel('s'); grid;
title('Flux function');
axis([0 10 0 1.6]);
```

An extensive research has been devoted to the proposal contained in the Perona-Malik paper. See [184] for a detailed treatment of the topic with abundant references. Also in [184] an anisotropic diffusion was introduced using a tensorial diffusion coefficient.

The Perona-Malik method can be implemented in several discretization ways. Program 2.15 represents an example of implementation that has fast execution time. Figure 2.35 shows the result of the method for the denoising of a picture having salt & pepper noise (this noise has been added using *imnoise()*). The noise has been softened while keeping image features.



Fig. 2.35 Denoising of image with salt & pepper noise, using P-M method

Program 2.15 Example of picture denoising using Perona-Malik diffusion

%Example of picture denoising using Perona-Malik diffusion

```

lambda=160; %constant for P-M diffusivity
D=0.1; %general diffusion constant
P=imread('face1.jpg'); %read image
A=imnoise(P,'salt&pepper',0.01); %add salt&pepper noise
u=double(A); un=u;
[ly,lx]=size(u);
for nn=1:8,
    %zero padding around image
    udif=zeros(ly+2,lx+2);
    udif(2:ly+1,2:lx+1)=u;
    %differences: north, south, east, west
    difN=udif(1:ly,2:lx+1)-u;
    difS=udif(3:ly+2,2:lx+1)-u;
    difE=udif(2:ly+1,3:lx+2)-u;
    difW=udif(2:ly+1,1:lx)-u;
    %Diffusivities
    DN=1./(1+(difN/lambda).^2);
    DS=1./(1+(difS/lambda).^2);
    DE=1./(1+(difE/lambda).^2);
    DW=1./(1+(difW/lambda).^2);
    %diffusion
    un=u+D*(DN.*difN + DS.*difS + DE.*difE + DW.*difW);
    u=un;
end;
%display:
figure(1)
subplot(1,2,1)
imshow(A);
xlabel('original');

```

```
title('Salt&pepper denoising');
subplot(1,2,2)
imshow(uint8(un));
xlabel('denoised image');
```

2.5.3 Bregman-Related Algorithms

A convenient iterative method for finding extrema of convex functions was proposed by Bregman in 1967, [28]. Later on, in 2005, it was shown by Osher et al. [142] that this method was very appropriate for image processing (in particular for total variation applications).

The method is based on the Bregman divergence, and it can be employed in its basic iterative version, or as split Bregman iteration [95].

2.5.3.1 Bregman Divergence

Aspects concerning divergence, distances, similarity, etc. have already been considered in the chapter on data analysis and classification. They are crucial for important applications, like for instance face recognition (and distinction between faces).

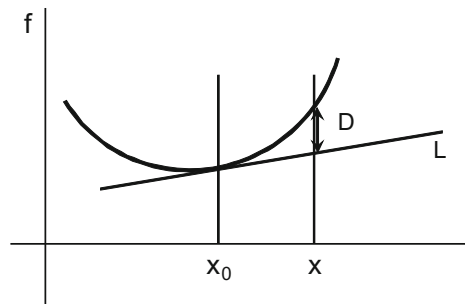
An expression of the Bregman divergence between two points \mathbf{x} and \mathbf{x}_0 would be the following:

$$D(\mathbf{x}, \mathbf{x}_0) = f(\mathbf{x}) - f(\mathbf{x}_0) - \nabla f(\mathbf{x}_0)^T \cdot (\mathbf{x} - \mathbf{x}_0) \quad (2.120)$$

This concept was introduced by Bregman for differentiable convex functions, and was given the name ‘*Bregman distance*’ by Censor and Lent [46] in 1981.

Figure 2.36 gives a graphical interpretation of this distance D , sitting above the tangent L .

Fig. 2.36 Example of Bregman distance



Notice that a first-order Taylor expansion to approximate $f(\mathbf{x})$, would be $f_a(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T (\mathbf{x} - \mathbf{x}_0)$, and therefore the distance D is the difference: $D(\mathbf{x} - \mathbf{x}_0) = f(\mathbf{x}) - f_a(\mathbf{x})$.

Distances are non-negative functions. A metric distance has symmetry, identity and triangle inequality properties. Bregman distances are not necessarily symmetric, nor supporting triangle inequality.

Particular cases of Bregman distance are the Euclidean distance, with the choice $f(\mathbf{x}) = \|\mathbf{x}\|^2$, the Kullback-Leibler divergence, with $f(\mathbf{x}) = \sum -x_i \log x_i$, and the Itakura-Saito distance, with $f(\mathbf{x}) = \sum -\log x_i$. The Mahalanobis distance is also a particular case.

Other equivalent expression of the Bregman distance is the following:

$$D(\mathbf{x}, \mathbf{x}_0) = f(\mathbf{x}) - f(\mathbf{x}_0) - \langle \nabla f(\mathbf{x}_0), \mathbf{x} - \mathbf{x}_0 \rangle \quad (2.121)$$

Recently, some important authors, [30, 95], used the following expression:

$$D(\mathbf{x}, \mathbf{x}_0) = f(\mathbf{x}) - f(\mathbf{x}_0) - \langle p, \mathbf{x} - \mathbf{x}_0 \rangle \quad (2.122)$$

where p is the subgradient at \mathbf{x}_0 .

There are interesting connections of Bregman divergences with clustering, exponential functions, and information theory [15]; or in particular with Voronoi diagrams [25]. It is also worthwhile to mention [7] for a more complete view on the concept and applications of divergence.

2.5.3.2 Bregman Iteration

Consider the following constrained minimization problem:

$$\min_u J(\mathbf{u}), \text{ subject to } H(\mathbf{u}) = 0$$

where J and H are defined in \Re^n and convex. The associated unconstrained problem is:

$$\min_u J(\mathbf{u}) + \lambda H(\mathbf{u}) \quad (2.123)$$

The Bregman distance of J between \mathbf{u} and \mathbf{v} would be:

$$D(\mathbf{u}, \mathbf{v}) = J(\mathbf{u}) - J(\mathbf{v}) - \langle p, \mathbf{u} - \mathbf{v} \rangle \quad (2.124)$$

The *Bregman iteration* for solving the minimization problem is:

$$\mathbf{u}^{(k+1)} = \arg \min_u D^{(k)}(\mathbf{u}, \mathbf{u}^{(k)}) + \lambda H(\mathbf{u}) \quad (2.125)$$

Suppose that H is differentiable. In this case, the Bregman iteration would be:

$$\mathbf{u}^{(k+1)} = \arg \min_{\mathbf{u}} D^{(k)}(\mathbf{u}, \mathbf{u}^{(k)}) + \lambda H(\mathbf{u}) \quad (2.126)$$

$$p^{(k+1)} = p^{(k)} - \lambda \nabla H(\mathbf{u}^{(k+1)}) \quad (2.127)$$

For the particular problem of TV based image denoising [142], one could take:

$$H(\mathbf{u}, \mathbf{f}) = \frac{1}{2} \|A \mathbf{u} - \mathbf{f}\|_2^2 \quad (2.128)$$

then, the iteration can be expressed in the following simplified form:

$$\mathbf{u}^{(k+1)} = \arg \min_{\mathbf{u}} J(\mathbf{u}) + \lambda H(\mathbf{u}, \mathbf{f}^{(k)}) \quad (2.129)$$

$$\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} + (\mathbf{f} - A \mathbf{u}^{(k+1)}) \quad (2.130)$$

It has been verified that these iterations converge very quickly.

2.5.3.3 Linearized Bregman Iteration

In appropriate cases, it would be possible to use the following approximation:

$$H(\mathbf{u}) \approx H(\mathbf{u}^{(k)}) + \nabla H(\mathbf{u}^{(k)}) \cdot (\mathbf{u} - \mathbf{u}^{(k)}) \quad (2.131)$$

For a better approximation a penalty quadratic term could be added, and then:

$$\mathbf{u}^{(k+1)} = \arg \min_{\mathbf{u}} J(\mathbf{u}) + \langle \lambda H(\mathbf{u}^{(k)}) - p^{(k)}, \mathbf{u} \rangle + \frac{1}{2\delta} \|\mathbf{u} - \mathbf{u}^{(k)}\|_2^2 \quad (2.132)$$

2.5.3.4 Split Bregman Iteration

A characteristic aspect of split Bregman method is that it could separate the typical l_1 and l_2 portions of important image processing approaches. For instance, suppose that the minimization problem is:

$$\min_{\mathbf{u}} \|\mathbf{d}\|_1 + H(\mathbf{u}) + \frac{\lambda}{2} \|\mathbf{d} - \phi(\mathbf{u})\|_2^2 \quad (2.133)$$

The split Bregman iteration for solving this problem is:

$$\mathbf{u}^{(k+1)} = \arg \min_{\mathbf{u}} H(\mathbf{u}) + \frac{\lambda}{2} \|\mathbf{d}^{(k)} - \phi(\mathbf{u}) - \mathbf{b}^{(k)}\|_2^2 \quad (2.134)$$

$$\mathbf{d}^{(k+1)} = \arg \min_{\mathbf{d}} \|\mathbf{d}\|_1 + \frac{\lambda}{2} \|\mathbf{d} - \phi(\mathbf{u}^{(k+1)}) - \mathbf{b}^{(k)}\|_2^2 \quad (2.135)$$

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + (\phi(\mathbf{u}^{(k+1)}) - \mathbf{d}^{(k+1)}) \quad (2.136)$$

As you can see, the method involves two optimization steps and a simple update of \mathbf{b} . The first optimization step is differentiable and so it can be solved by a variety of methods, like Gauss-Seidel or conjugate gradient, or even in the Fourier domain. The second optimization step can be computed by shrinkage, that is:

$$\mathbf{d}_j^{(k+1)} = \text{shrink}(\phi(\mathbf{u})_j + \mathbf{b}_j^{(k)}, 1/\lambda) \quad (2.137)$$

where the shrinkage would be:

$$\text{shrink}(\mathbf{x}, \lambda) = \frac{\mathbf{x}}{|\mathbf{x}|} * \max(|\mathbf{x}| - \lambda, 0) \quad (2.138)$$

2.5.3.5 Using Split Bregman for ROF-TV Image Denoising

It has been shown in [95] how to apply the split-Bregman method for total variation (TV) image denoising, based on the ROF model. The method is simple and efficient. It handles two-dimensional variables.

There are two denoising formulations: the anisotropic problem, and the isotropic problem. In the *anisotropic* problem, one has to solve:

$$\min_{\mathbf{u}} |\nabla_x \mathbf{u}| + |\nabla_y \mathbf{u}| + \frac{\mu}{2} \|\mathbf{u} - f\|_2^2 \quad (2.139)$$

Let us replace $\nabla_x \mathbf{u}$ by d_x and $\nabla_y \mathbf{u}$ by d_y . In order to strengthen the constraints, two penalty terms were added:

$$\min_{\mathbf{u}} |d_x| + |d_y| + \frac{\mu}{2} \|\mathbf{u} - f\|_2^2 + \frac{\lambda}{2} \|d_x - \nabla_x \mathbf{u}\| + \frac{\lambda}{2} \|d_y - \nabla_y \mathbf{u}\| \quad (2.140)$$

Coming now to the split Bregman algorithm, the first optimization step would be:

$$\mathbf{u}^{(k+1)} = \arg \min_{\mathbf{u}} \frac{\mu}{2} \|\mathbf{u} - f\|_2^2 + \frac{\lambda}{2} \|d_x^{(k)} - \nabla_x \mathbf{u} - b_x^{(k)}\| + \frac{\lambda}{2} \|d_y - \nabla_y \mathbf{u} - b_y^{(k)}\| \quad (2.141)$$

Because the system is strictly diagonal, it is recommended in [95] to get the solution with the Gauss-Seidel method:

$$u_{i,j}^{(k+1)} = G_{i,j}^{(k)} = \frac{\mu}{\mu+4\lambda} f_{i,j} + \frac{\lambda}{\mu+4\lambda} \{ u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)} + d_{x,i-1,j}^{(k)} - d_{x,i,j}^{(k)} + d_{y,i,j-1}^{(k)} - d_{y,i,j}^{(k)} - b_{x,i-1,j}^{(k)} + b_{x,i,j}^{(k)} - b_{y,i,j-1}^{(k)} + b_{y,i,j}^{(k)} \}$$

Then, the complete split-Bregman algorithm, to be iterated, is:

$$u^{(k+1)} = G^{(k)} \quad (2.142)$$

$$d_x^{(k+1)} = \text{shrink}(\nabla_x u^{(k+1)} + b_x^{(k)}, 1/\lambda) \quad (2.143)$$

$$d_y^{(k+1)} = \text{shrink}(\nabla_y u^{(k+1)} + b_y^{(k)}, 1/\lambda) \quad (2.144)$$

$$b_x^{(k+1)} = b_x^{(k)} + (\nabla_x u^{(k+1)} - d_x^{k+1}) \quad (2.145)$$

$$b_y^{(k+1)} = b_y^{(k)} + (\nabla_y u^{(k+1)} - d_y^{k+1}) \quad (2.146)$$

In the case of *isotropic* denoising, the minimization problem is:

$$\min_u \sum \sqrt{(\nabla_x u)_i^2 + (\nabla_y u)_i^2} + \frac{\mu}{2} \|u - f\|_2^2 \quad (2.147)$$

Like before, the problem is transformed to:

$$\min_u \sum_{i,j} \sqrt{d_{x,i,j}^2 + d_{y,i,j}^2} + \frac{\mu}{2} \|u - f\|_2^2 + \frac{\lambda}{2} \|d_x - \nabla_x u\| + \frac{\lambda}{2} \|d_y - \nabla_y u\| \quad (2.148)$$

Due to the coupling between d_x and d_y , the second optimization step is decomposed into two shrinkages. Hence, the split-Bregman algorithm, to be iterated, is the following:

$$u^{(k+1)} = G^{(k)} \quad (2.149)$$

$$d_x^{(k+1)} = \max(s^k - 1/\lambda, 0) \frac{\nabla_x u^{(k)} + b_x^{(k)}}{s^k} \quad (2.150)$$

$$d_y^{(k+1)} = \max(s^k - 1/\lambda, 0) \frac{\nabla_y u^{(k)} + b_y^{(k)}}{s^k} \quad (2.151)$$

$$b_x^{(k+1)} = b_x^{(k)} + (\nabla_x u^{(k+1)} - d_x^{k+1}) \quad (2.152)$$

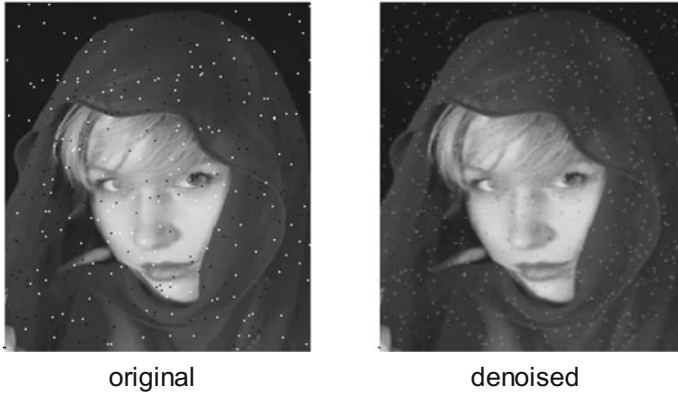


Fig. 2.37 ROF total variation denoising using split Bregman

$$b_y^{(k+1)} = b_y^{(k)} + (\nabla_y u^{(k+1)} - d_y^{k+1}) \quad (2.153)$$

where:

$$s^k = \sqrt{|\nabla_x u^{(k)} + b_x^{(k)}|^2 + |\nabla_y u^{(k)} + b_y^{(k)}|^2} \quad (2.154)$$

According with [30], the shrinking steps could be approximated as follows:

$$d_x^{(k+1)} = \frac{s^k \lambda (\nabla_x u^{(k)} + b_x^{(k)})}{s^k \lambda + 1} \quad (2.155)$$

$$d_y^{(k+1)} = \frac{s^k \lambda (\nabla_y u^{(k)} + b_y^{(k)})}{s^k \lambda + 1} \quad (2.156)$$

Figure 2.37 shows an example of ROF-TV anisotropic denoising. It is the same example as before, with salt and pepper noise. The program is based on the implementation in [30]. It has been included in the appendix on long programs. Actually, the program is not complicated, but the discretization implies many loops; surely the algorithm could be implemented in more compact form, using matrices.

2.6 Matrix Completion and Related Problems

A matrix could be regarded as a set of signal samples. Therefore, it can be treated from the compressed sensing point of view. For instance, it may correspond to the measurements taken at a given time by a network of spatially distributed sensors, in which some sensors could fail.

The consideration of matrices has originated a fruitful new research area, which is very active and expansive. This section introduces some fundamental aspects of this area.

2.6.1 Matrix Completion

In a most cited article, [40], Candés and Recht introduced a topic of considerable interest: the recovery of a data matrix from a sampling of its entries. A number m of entries are chosen uniformly at random from a matrix M , and the question is whether it is possible to entirely recover the matrix M from these m entries.

The topic was naturally introduced by extension of compressed sensing, in which a sparse signal is recovered from some samples taken at random. In the case of matrices, if the matrix M has low rank or approximately low rank, then accurate and even exact recovery from random sampling is possible by nuclear norm minimization [40].

By the way, it is now convenient to quote a certain set of norms, denoted as *Schatten- p norms*, for $p = 1, 2$ or ∞ . In particular:

- Spectral norm:

$$\text{The largest singular value: } \|X\|_S = \max(\sigma_i(X)) = \|\sigma(X)\|_\infty$$

- Nuclear norm:

$$\text{The sum of singular values: } \|X\|_* = \sum_{i=1}^N \sigma_i(X) = \|\sigma(X)\|_1$$

- Frobenius norm:

$$\|X\|_F = \left(\sum_{i=1}^N \sum_{j=1}^M X_{ij}^2 \right)^{1/2} = \left(\sum_{i=1}^t \sigma_i^2(X) \right)^{1/2} = \|\sigma(X)\|_2$$

Using a simple example of a $n \times n$ matrix M that has all entries equal to zero except for the first row, it is noted in [40] that this matrix cannot be recovered from a subset of its entries. There are also more pathological cases as well. In general, we need the singular vectors of M to be spread across all coordinates. If this happens, the recovery could be done by solving a rank minimization problem.

$$\text{minimize } \text{rank}(X), \text{ subject to } X_{ij} = M_{ij}, (i, j) \in \Omega$$

where X is the recovered matrix.

Like in the case of compressed sensing, where a $l1$ norm is used instead of a $l0$ norm for easiest treatment, it is preferred to consider the minimization of the nuclear norm:

$$\text{minimize } \|X\|_*, \text{ subject to } X_{ij} = M_{ij}, (i, j) \in \Omega$$

It is shown in [40] that most matrices can be recovered provided that the number of samples m obeys:

$$m \geq C n^{6/5} r \log n \quad (2.157)$$

where C is some positive constant, and r is the rank.

The nuclear norm minimization problem can be solved in many ways. A special mention should be done of the singular value thresholding algorithm introduced in [32]. This algorithm is a de facto reference for comparison with other algorithms (see the web page on low-rank matrix completion cited in the section on resources).

Actually, the proximal operator corresponding to the minimization of $\|X\|_*$ is the singular value shrinkage operator that shrinks towards zero the singular values of X , by subtracting a certain threshold value [145].

An example of matrix sampling and recovery is given below. The Douglas-Rachford splitting is used [151], based on two proximal operators: one corresponds to the indicator function, and the other to the nuclear norm. In the case of the indicator function, the proximal operator is a projection that accumulates repeated samples of the same matrix entry. The algorithm is implemented in the Program 2.16, which generates two figures.

Figure 2.38 shows the evolution of the nuclear norm as the iterations go on. It converges in less than 100 iterations.

In order to show how accurate is the recovery, a simple plot has been devised. One takes a row of the original matrix and represents its entry values as ‘x’ points. Then, one plots on top the recovered values for this row, as a series of segments. Figure 2.39 shows the result for the 10th row. It can be seen that the recovery is fairly precise.

Fig. 2.38 Evolution of nuclear norm

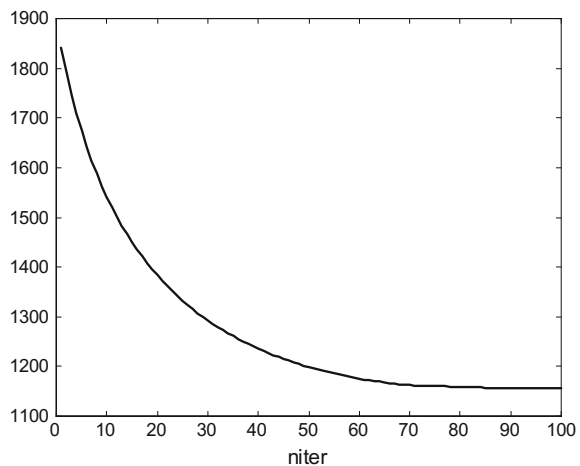
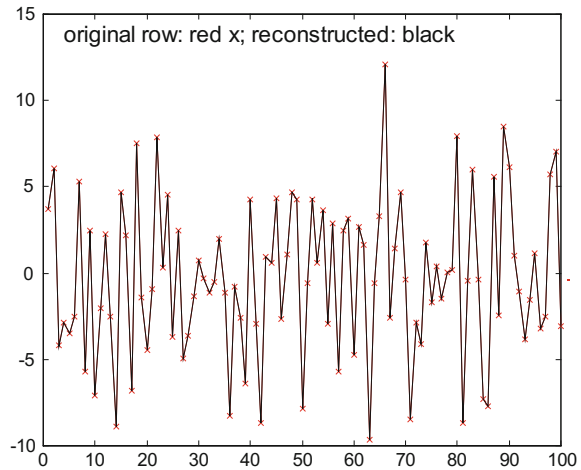


Fig. 2.39 A test of reconstruction quality



Program 2.16 Example of Matrix Completion

```
% Example of Matrix Completion
% (using Douglas-Rachford)
clear all;
disp('working...')
% a random matrix of rank r
n=100;
r=12; %rank
A=randn(n,r)*randn(r,n); % Original random matrix
disp('nuclear norm of original matrix:');
[u,d,v]=svd(A);
nuc_N = sum(diag(d))
% a subset of A is measured by
% sampling at random a number p of entries of A
p=round(n*log(n)*r); % see theory
aux=randperm(n*n);
ix=aux(1:p)'; %extract p integer random numbers
% b is a column vector:
b=A(ix); %retain a subset of entries of A
% start the algorithm -----
X=zeros(n,n);
Y=zeros(n,n);
L=n*n;
niter=100;
lambda=1; gamma=2;
rnx=zeros(niter,1);
for nn=1:niter,
    % X update
    % R(b-M(Y)) term
    O=zeros(L,1);
```

```

% accumulating repeated entries
for j=1:p;
    ox=ix(j);
    O(ox)=O(ox)+(b(j)-Y(ox));
end;
Q=reshape(O,[n n]);
% proxF (indicator function)
X=Y+Q;
% Y update
%proxG (soft thresholding of singular values):
P=(2*X)-Y;
[U,D,V]=svd(P);
for j=1:n,
    aux=D(j,j);
    if abs(aux)<=gamma,
        D(j,j)=0;
    else
        if aux>gamma, D(j,j)=aux-gamma; end
        if aux<-gamma, D(j,j)=aux+gamma; end;
    end;
end;
S=U*D*V'; % result of thresholding
Y=Y+ (lambda*(S-X));
% recording
[u,d,v]=svd(X);
rn timer= sum(diag(d)); %nuclear norm
end
%display -----
% evolution of nuclear norm
figure(1)
plot(rnx,'k');
title('evolution of nuclear norm')
xlabel('niter');
% see a matrix row: original and reconstructed
figure(2)
Nrow=10; %(edit this number)
plot(A(Nrow,:), 'r-x'); hold on;
plot(X(Nrow,:), 'k');
title('original row: red x; reconstructed: black');
% a measure of error
er=A(Nrow,:)-X(Nrow,:);
E=sum(er.^2) %to be printed

```

The issues related to matrix completion have attracted a lot of research activity. For instance, what assumptions are needed to guarantee the matrix recovery? A recent article on this aspect is [98], which includes important references. See [69] for another perspective that connects phase transitions and matrix denoising. An extensive work on fundamental limits and efficient algorithms is [137]. It happens that the SVD

decomposition applied in the first proposed algorithms can imply excessive computational effort for large matrices, and so many improvements or alternatives have been explored [129, 182]. Some authors have proposed the use of other norms, instead of the nuclear norm (see [117] and references therein).

2.6.2 Decomposition of Matrices

It was said in [39] that in real world applications the measured entries would be corrupted by noise (perhaps outliers, [190]). This observation originated a new type of problem, in which one has to find a decomposition of the observed matrix M into a low-rank matrix L and a sparse matrix S . The problem was recognized as a robust PCA analysis in [38] (a most cited paper), being stated as follows:

$$\text{minimize } (\|L\|_* + \lambda \|S\|_1), \text{ subject to } M = L + S$$

Again, the new problem lead to a broad range of research efforts, which have found many interesting applications.

Let us build a simple example by adding a low-rank random matrix and a sparse matrix (just a diagonal matrix). Figure 2.40 shows images corresponding to these matrices.

The result of adding the previous two matrices is shown in Fig. 2.41.

Now, the problem is to recover L and S from M . One of the methods that can be used is an adaptation of the Douglas-Rachford algorithm for this case. According with [86], it can be formulated as follows:

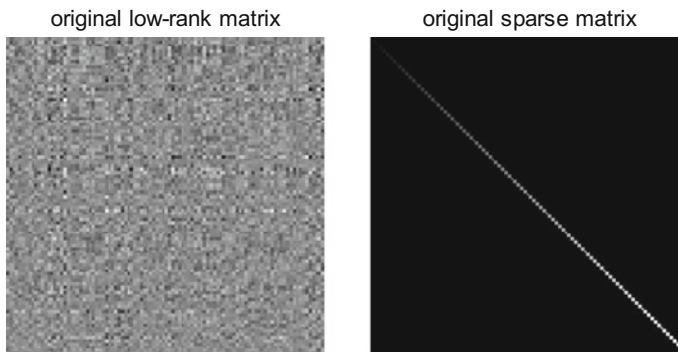
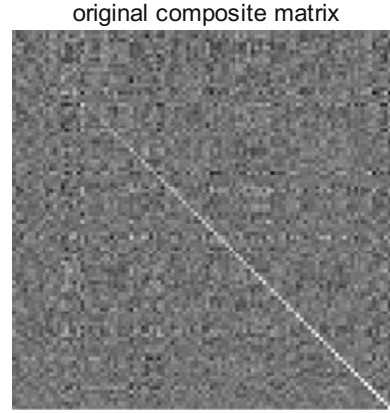


Fig. 2.40 A low-rank matrix and a sparse matrix

Fig. 2.41 The observed matrix $M = L + S$



repeat:

$$Le = (M + L^{(k)} - S^{(k)})/2 \quad (2.158)$$

$$Se = (M - L^{(k)} + S^{(k)})/2 \quad (2.159)$$

$$L^{(k+1)} = L^{(k)} + t_k(\text{shrink}(2Le - L^{(k)}) - Le) \quad (2.160)$$

$$S^{(k+1)} = S^{(k)} + t_k(\text{soft_threshold}(2Se - S^{(k)}) - Se) \quad (2.161)$$

until convergence.

The shrinkage corresponds to the proximity operator for the nuclear norm; and the soft-thresholding corresponds to the proximity operator for the l_1 norm.

An implementation of this algorithm is provided by Program 2.17. The result is satisfactory as shown in Fig. 2.42.

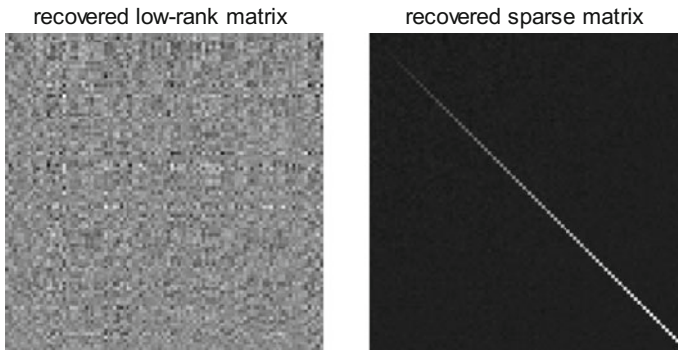


Fig. 2.42 The recovered matrices L and S

Program 2.17 Decomposition into low-rank (L) and sparse (S) matrices

```

% decomposition into low-rank (L) and sparse (S) matrices
% Douglas-Rachford
% a random matrix of rank r
n=100;
r=8; %rank
L0=randn(n,r)*randn(r,n); % a low-rank matrix
% a sparse (diagonal) matrix
nn=1:100;
S0=diag(0.2*nn,0);
% composite original matrix
M=L0+S0;
% parameter settings
lambda=1;
tk=1;
Th=3; %Threshold
nnL=30; % number of loops
rx=zeros(nnL,1);
L=zeros(n,n); S=zeros(n,n);
% start the algorithm -----
for nn=1:nnL,
    Le=0.5*(M+L-S); Se=0.5*(M-L+S);
    % shrinking-----
    aux1=(2*Le)-L;
    [U D V]=svd(aux1);
    for j=1:n,
        D(j,j)=max(D(j,j)-Th,0);
    end;
    aux=U*D*V';
    L=L+(tk*(aux-Le));
    % soft_threshold-----
    aux1=(2*Se)-S;
    aux=sign(aux1).*max(0,abs(aux1)-lambda);
    S=S+(tk*(aux-Se));
    [u,d,v]=svd(L);
    rx(nn)=sum(diag(d)); %nuclear norm, record
end;
% display -----
figure(1)
subplot(1,2,1)
imshow(L0,[]);
title('original low-rank matrix')
subplot(1,2,2)
imshow(S0,[]);
title('original sparse matrix')
figure(2)
imshow(M,[]);
title('original composite matrix');

```

```

figure(3)
subplot(1,2,1)
imshow(L, []);
title('recovered low-rank matrix')
subplot(1,2,2)
imshow(S, []);
title('recovered sparse matrix')

```

A number of methods for matrix decomposition has been proposed. Soon after the publication of [39], the ‘*principal component pursuit*’ (PCP) was introduced. In [204] a study of PCP was presented, with mentions to robust PCA and a reference to [38] as preprint. In general, the preferred methods are based on alternating minimization schemes, which are reviewed in the introduction of [170]. One of the factors that promote the popularity of certain methods is the public availability of code [115, 116, 196]. Theoretical aspects on conditions for the recovery of matrices are treated in [51].

Some illustrative application examples are [148] for alignment of images, [199] for low-rank image textures, [54] for face recognition based on robust PCA, [113] on cognitive radio networks, [203] for medical image analysis, [13] on target tracking (for example a TV camera focusing on a basketball player during the game), [65] on computer vision, [56] on genotype imputation, or [181] for movie colorization.

2.7 Experiments

This section includes three experiments, the first is an example of 1D signal denoising, while the other two examples are applications of matrix completion and decomposition to images.

2.7.1 Signal Denoising Based on Total Variation (TV)

Let us consider an example of signal denoising based on TV, using an algorithm proposed in [169]. Given a signal $x(n)$ composed of N samples, its total variation would be:

$$TV(x) = \sum_{i=1}^N (x(n) - x(n-1)) \quad (2.162)$$

An equivalent expression is the following:

$$TV(x) = \|Dx\|_1 \quad (2.163)$$

with:

$$D = \begin{bmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & & -1 & 1 \end{bmatrix} \quad (2.164)$$

Suppose a noise is added to the signal, and then:

$$y = x + n \quad (2.165)$$

In order to denoise the signal y , the following criterion can be defined:

$$J(x) = \|y - x\|_2^2 + \lambda \|Dx\|_1 \quad (2.166)$$

and a solution x must be found for the minimization of $J(x)$.

The minimization algorithm proposed in [169] is the following:

$$x_{k+1} = y - D^T z_k \quad (2.167)$$

$$z_{k+1} = \text{clip}(z_k + \frac{1}{\alpha} Dx_{k+1}; \lambda/2) \quad (2.168)$$

with $z_0 = 0$ and $\alpha \geq \max \text{eig}(D D^T)$.

The algorithm uses a clipping function, which is defined as follows:

$$\text{clip}(y; T) = \begin{cases} y, & |y| < T \\ T \text{ sign}(y), & |y| > T \end{cases} \quad (2.169)$$

An implementation of this algorithm is provided by the Program 2.18, which is a modification of the code given in [169]. The case considered is a signal with some abrupt changes. The original signal, and the same signal with added noise, are shown in Fig. 2.43.

Figure 2.44 shows the evolution of the minimization of $J(x)$, and the denoised signal as it was obtained at the end of the iterations. Note that the abrupt changes of the signal have been preserved.

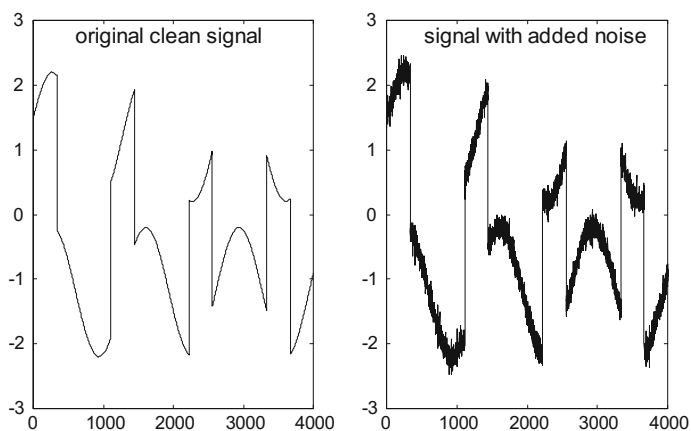


Fig. 2.43 Signal to be denoised

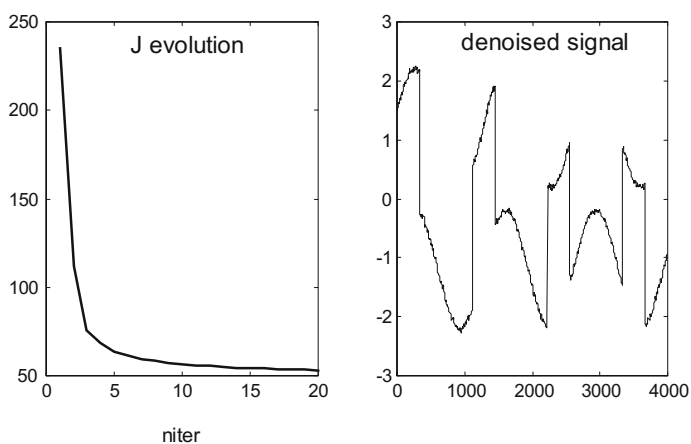


Fig. 2.44 TV denoising: (left) convergence, (right) denoised signal

Program 2.18 Example of TV denoising

```
% Example of TV denoising
clear all;
% build a test signal
t=0:0.1:400;
N=length(t);
u= sin(0.3+(0.015*pi*t));
v= 1.2*square(0.018*pi*t,30);
a=u+v; %signal with no noise
x=a+(0.1*randn(1,N)); %signal with noise
% Prepare for computations
niter=20;
```

```

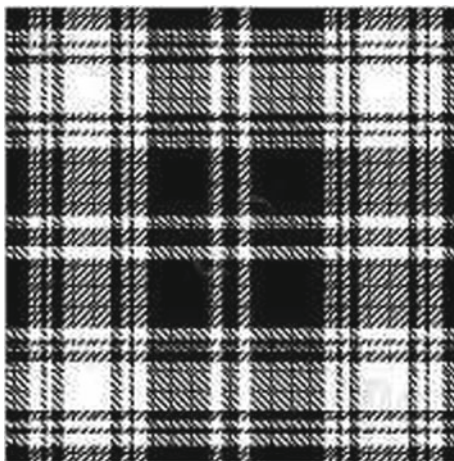
z=zeros(1,N-1);
J=zeros(1,niter);
alpha=3;
lambda=0.5; th=lambda/2;
%start the algorithm -----
for nn=1:niter,
    aux=[-z(1) -diff(z) z(end)];
    y=x-aux;
    aux1=sum(abs(y-x).^2);
    aux2=sum(abs(diff(y)));
    J(nn)=aux1+(lambda*aux2);
    z=z+((1/alpha)*diff(y));
    z=max(min(z,th),-th);
end
% display -----
figure(1)
subplot(1,2,1)
plot(a,'k')
axis([0 4000 -3 3]);
title('original clean signal')
subplot(1,2,2)
plot(x,'k')
axis([0 4000 -3 3]);
title('signal with added noise')
figure(2)
subplot(1,2,1)
plot(J,'k')
xlabel('niter')
title('J evolution')
subplot(1,2,2)
plot(y,'k')
axis([0 4000 -3 3]);
title('denoised signal')

```

Another way of computing the total variation denoising of unidimensional signals was proposed in [167].

2.7.2 Picture Reconstruction Based on Matrix Completion

A direct example of matrix completion is the case of taking at random some pixels of a picture, and see if it is possible to recover the picture from these samples.

Fig. 2.45 Original image

An image with evident redundancies has been chosen, Fig. 2.45. It could be expected that this image has approximately low rank. Anyway, a certain value of p was tried.

The sampling and recovery experiment was done with the Program 2.19, which is just an adaptation of the Program 2.16.

Figure 2.46 shows the evolution of the nuclear norm along iterations.

And Fig. 2.47 shows the good result of the image recovery.

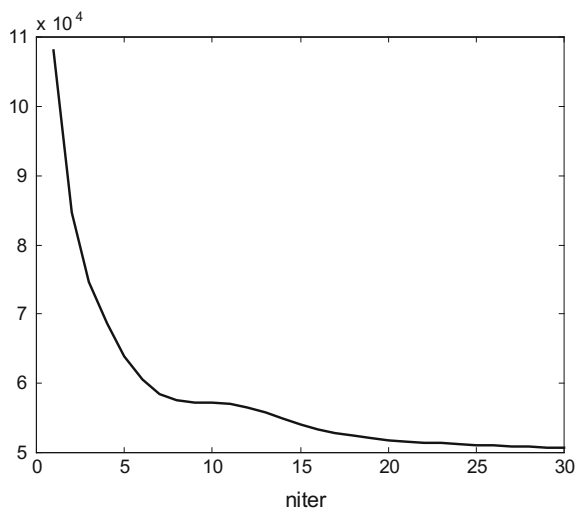
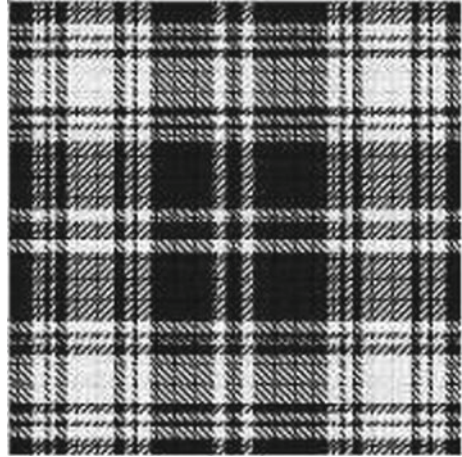
Fig. 2.46 Evolution of nuclear norm

Fig. 2.47 Image reconstruction by matrix completion



Program 2.19 Example of Picture Completion

```
% Example of Picture Completion
% (using Douglas-Rachford)
clear all;
disp('working...')
% original picture
figu=imread('tartan.jpg'); %read picture
F=double(figu);
A=F(1:200,1:200); %crop;
aux=mean(mean(A));
A=A-aux;
n=200;
disp('nuclear norm of original matrix:');
[u,d,v]=svd(A);
nuc_N = sum(diag(d))
% a subset of A is measured by
% sampling at random a number p of entries of A
p=10000;
aux=randperm(n*n);
ix=aux(1:p)'; %extract p integer random numbers
% b is a column vector:
b=A(ix); %retain a subset of entries of A
% start the algorithm-----
X=zeros(n,n);
Y=zeros(n,n);
L=n*n;
niter=30;
lambda=1; gamma=1000; %(notice the value of gamma)
rnz=zeros(niter,1);
for nn=1:niter,
```



```

% X update
% R(b-M(Y)) term
O=zeros(L,1);
% accumulating repeated entries
for j=1:p;
    ox=ix(j);
    O(ox)=O(ox)+(b(j)-Y(ox));
end;
Q=reshape(O,[n n]);
% proxF (indicator function)
X=Y+Q;
% Y update
%proxG (soft thresholding of singular values):
P=(2*X)-Y;
[U,D,V]=svd(P);
for j=1:n,
    aux=D(j,j);
    if abs(aux)<=gamma,
        D(j,j)=0;
    else
        if aux>gamma, D(j,j)=aux-gamma; end
        if aux<-gamma, D(j,j)=aux+gamma; end;
    end;
end;
S=U*D*V'; % result of thresholding
Y=Y+ (lambda*(S-X));
% recording
[u,d,v]=svd(X);
rn timer=0; %nuclear norm
end
%display -----
% evolution of nuclear norm
figure(1)
plot(timer,'k');
title('evolution of nuclear norm')
xlabel('niter');
figure(2)
imshow(A,[]);
title('original picture')
figure(3)
imshow(X,[]);
title('reconstructed picture')

```

2.7.3 Text Removal

Suppose you have a photograph and someone has written some text on it. It would be good to remove that text. Given this situation, matrix decomposition could be helpful, as far as the picture has low rank, and the text corresponds to a sparse matrix of pixels.

In order to explore this application, a synthetic problem has been fabricated: some text has been added to a wall of bricks. It is a crude simulation of a graffiti. Figure 2.48 shows the original image.

Since we wanted to explore alternating minimization schemes, a fast scheme proposed by [159]. In this paper, the matrix decomposition problem is treated as:

$$\text{minimize } \left(\frac{1}{2} \|L + S - M\|_F + \lambda \|S\|_1 \right), \text{ subject to } \text{rank}(L) = t$$

The problem is solved with the following alternating minimization:

$$\begin{aligned} L^{(k+1)} &= \arg \min_L (\|L + S^{(k)} - M\|_F), \text{ subject to } \text{rank}(L) = t \\ S^{(k+1)} &= \arg \min_S (\|L^{(k+1)} + S - M\|_F + \lambda \|S\|_1) \end{aligned} \quad (2.170)$$

These two sub-problems are solved as follows:

- The first sub-problem is solved with a partial SVD of $S^{(k)} - M$. Only t components of the SVD are selected (corresponding to the t largest singular values). This is done with the *lansvd()* routine included in the PROPACK library [114], which is commonly found in the public available codes.
- The second sub-problem is a element-wise shrinkage of $M - L^{(k)}$.

Fig. 2.48 Simulated graffiti image



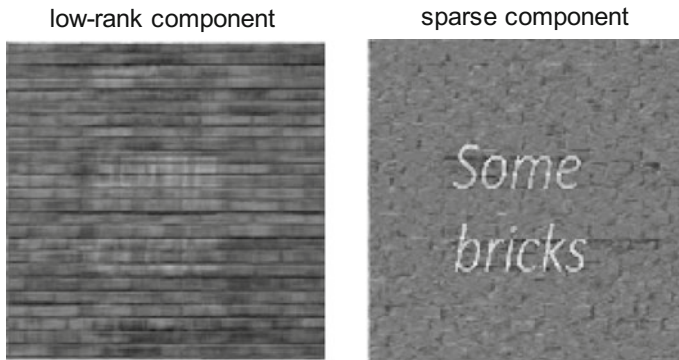


Fig. 2.49 Image decomposition into low-rank and sparse matrices

Program 2.20 presents a simple implementation of the algorithm. Figure 2.49 shows the result, which is not perfect but gives an idea of the approach. Of course, the image on the left hand would require better inpainting, and the image on the right hand would thank a better background extraction.

Program 2.20 Decomposition into low-rank (L) and sparse (S) matrices

```
% decomposition into low-rank (L) and sparse (S) matrices
% Alternating Minimization
% load image
figu=imread('wall.jpg'); %read picture
F=double(figu);
n=380;
M=F(1:n,1:n); %crop;
aux=mean(mean(M));
M=M-aux;
% parameter settings
lambda=0.5;
lambF=1.01;
Th=0.01; %Threshold
rank0=1; %intial rank guess
irk=1; %for rank increments
nnL=50; % number of loops
rank=rank0; %current rank
% start the algorithm -----
%
[UL SL VL] = lansvd(M, rank, 'L'); %partial SVD
L1=UL*SL*VL'; %initial low-rank approximation
aux=M-L1;
S1= sign(aux).*max(0,abs(aux)-lambda); %shrinkage
for nn=2:nnL,
    if irk==1,
        lambda = lambda * lambF; % lambda is modified in each iteration
        rank = rank + irk; % rank is increased " " "
```

```

end;
[UL SL VL] = lansvd(M-S1, rank, 'L'); %partial SVD
L1=UL*SL*VL'; %current low-rank approximation
aux=M-L1;
S1= sign(aux).*max(0,abs(aux)-lambda); %shrinkage
% change rank increment when appropriate
vv=diag(SL);
rho=vv(end)/sum(vv(1:end-1));
if rho<Th,
    irk=0;
else
    irk=1;
end;
end;
% display -----
figure(1)
imshow(M, []);
title('original picture')
figure(2)
subplot(1,2,1)
imshow(L1, []);
title('low-rank component')
subplot(1,2,2)
imshow(S1, []);
title('sparse component')

```

Background extraction or subtraction is a subject of considerable interest. The web site of T. Bouwmans offer several surveys of this topic. Fast techniques are needed in the case of video, [144, 188]. One application example is video surveillance [19, 97].

2.8 Resources

The file exchange web site of Mathworks has several MATLAB programs of interest. Also, some of the methods introduced in this chapter are implemented in some routines of the MATLAB Statistics Toolbox.

2.8.1 MATLAB

2.8.1.1 Toolboxes

- l1-MAGIC:
<http://users.ece.gatech.edu/~justin/l1magic/>
- SparseLab (Stanford University):
<http://sparselab.stanford.edu/>

- SpaSM Toolbox (includes LARS):
<http://www.imm.dtu.dk/projects/spasm/>
- SPAMS (C++ interfaced to MATLAB):
<http://spams-devel.gforge.inria.fr/doc-R/html/index.html>
- NESTA (Stanford University):
<http://statweb.stanford.edu/~candes/nesta/>
- Toolbox Sparse Optimization (G. Peyre):
<http://www.mathworks.com/matlabcentral/fileexchange/16204-toolbox-sparse-optimization>
- Model-based Compressive Sensing Toolbox (Rice University):
<http://dsp.rice.edu/software/model-based-compressive-sensing-toolbox-v11>
- Toolbox Sparsity (G. Peyre):
<http://www.ceremade.dauphine.fr/~peyre/matlab/sparsity/content.html>
- UNLocBoX (convex optimization toolbox):
<http://unlocbox.sourceforge.net/>

2.8.1.2 Matlab Code and Scripts

- SALSA:
<http://cascais.lx.it.pt/~mafonso/salsa.html>
- TwIST:
<http://www.lx.it.pt/~bioucas/TwIST/TwIST.htm>
- SpaRSA:
<http://www.lx.it.pt/~mtf/SpaRSA/>
- MATLAB scripts for ADMM (Stanford University):
<http://www.web.stanford.edu/~boyd/papers/admm!>
- YALL1 (Rice University):
<http://yall1.blogs.rice.edu/>
- Beginners code for CS (A. Weinstein):
http://control.mines.edu/mediawiki/upload/f/f4/Beginners_code.pdf
- MATLAB script for Chan-Vese segmentation (Fields Institute):
<http://www.math.ucla.edu/~wittman/Fields/cv.m>
- Low-Rank Matrix Recovery and Completion via Convex Optimization:
<http://perception.csl.illinois.edu/matrix-rank/home.html>

2.8.2 *Internet*

2.8.2.1 Web Sites

- Dave Donoho:
<http://www-stat.stanford.edu/~donoho>
- Emmanuel Candès (software):
<http://statweb.stanford.edu/~candes/software.html>
- Michael Elad:
www.cs.technion.ac.il/~elad/index.html
- A. Rakotomamonjy:
<http://asi.insa-rouen.fr/enseignants/~arakoto/>
- Mark Schmidt:
<http://www.cs.ubc.ca/~schmidtm/>
- Mark A. Davenport (CoSaMP):
<http://users.ece.gatech.edu/~mdavenport/software/>
- SeDuMi (Lehigh University):
<http://sedumi.ie.lehigh.edu/>
- Tiany Zhou (CS reconstruction algorithms):
<https://tianyizhou.wordpress.com/2010/08/23/compressed-sensing-review-1-reconstruction-algorithms/>
- CS Audio Demonstration:
<http://sunbeam.ece.wisc.edu/csaudio/>
- EPFL Signal Processing Lab (Lausanne):
<http://lts2www.epfl.ch/people/gilles/software>
- Bio Imaging & Signal Processing Lab.:
<http://bispl.weebly.com/software.html>
- J. Huang:
<http://ranger.uta.edu/~huang/index.html>
- Douglas-Rachford and projection methods:
<http://carma.newcastle.edu.au/DRmethods/>
- Bamdev Mishra (Riemannian matrix completion):
<http://sites.google.com/site/bamdevm/codes/qgeommc>
- Principal Component Pursuit papers and code:
<http://investigacion.pucp.edu.pe/grupos/gpsdi/publicaciones-2/>
- Thierry Bouwmans (surveys):
<https://sites.google.com/site/thierrybouwmans/recherche---background-subtraction---survey>

2.8.2.2 Link Lists

- Numerical-tours:
<http://www.numerical-tours.com/links/>
- Fast l1 Minimization Algorithms:
<http://www.eecs.berkeley.edu/~yang/software/l1benchmark/>
- Compressive sensing:
<https://sites.google.com/site/igorcarron2/compressivesensing2.0>
- The advanced matrix factorization jungle:
<https://sites.google.com/site/igorcarron2/matrixfactorizations>
- Compressive Sensing: The Big Picture:
<https://sites.google.com/site/igorcarron2/cs>
- Matrix completion solvers:
<http://www.ugcs.caltech.edu/~srbecker/wiki/Category:Matrix{ }Completion{ }Solvers>
- Research in Computational Science:
<http://www.csee.wvu.edu/~xinl/source.html>

References

1. S. Abeyruwan, *Least Angle Regression* (University of Miami, 2012). <https://sites.google.com/site/samindaa/lars>
2. R. Acar, C.R. Vogel, Analysis of bounded variation penalty methods for ill-posed problems. *Inverse Prob.* **10**(6), 1217 (1994)
3. B. Adcock, *An Introduction to Compressed Sensing* (Department of Mathematics, Simon Fraser University, 2015). <http://benadcock.org/wp-content/uploads/2015/11/AdcockBeijingTalk1.pdf>
4. B. Adcock, A.C. Hansen, C. Poon, B. Roman, *Breaking the Coherence Barrier: Asymptotic Incoherence and Asymptotic Sparsity in Compressed Sensing* (Department of Mathematics, Purdue University, 2013). arXiv preprint [arXiv:1302.0561](https://arxiv.org/abs/1302.0561)
5. M.V. Afonso, J.M. Bioucas-Dias, M.A. Figueiredo, Fast image recovery using variable splitting and constrained optimization. *IEEE Trans. Image Process.* **19**(9), 2345–2356 (2010)
6. M. Aharon, M. Elad, A. Bruckstein, K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE T. Signal Process.* **54**(11), 4311–4322 (2006)
7. S.I. Amari, A. Cichocki, Information geometry of divergence functions. *Bull. Polish Acad. Sci.: Tech. Sci.* **58**(1), 183–195 (2010)
8. F.J.A. Artacho, M. Jonathan, Recent results on Douglas-Rachford methods. *Serdica Math. J.* **39**, 313–330 (2013)
9. H. Attouch, *Alternating Minimization and Projection Algorithms. From Convexity to Non-convexity* (2009). <http://events.math.unipd.it/nactde09/sites/default/files/attouch.pdf>
10. J.F. Aujol, G. Aubert, L. Blanc-Féraud, A. Chambolle, Image decomposition into a bounded variation component and an oscillating component. *J. Math. Imag. Vision* **22**(1), 71–88 (2005)
11. J.F. Aujol, G. Gilboa, Implementation and parameter selection for BV-Hilbert space regularization. Technical report, UCLA, 2004. CAM04-66

12. J.F. Aujol, G. Gilboa, T. Chan, S. Osher, Structure-texture image decomposition—modeling, algorithms, and parameter selection. *Int. J. Comput. Vision* **67**(1), 111–136 (2006)
13. M. Ayazoglu, M. Sznaiar, O.I. Camps, Fast algorithms for structured robust principal component analysis, in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, pp. 1704–1711 (2012)
14. W.U. Bajwa, J. Haupt, A.M. Sayeed, R. Nowak, Compressed channel sensing: a new approach to estimating sparse multipath channels. *Proc. IEEE* **98**(6), 1058–1076 (2010)
15. A. Banerjee, S. Merugu, I.S. Dhillon, J. Ghosh, Clustering with Bregman divergences. *J. Mach. Learn. Res.* **6**, 1705–1749 (2005)
16. C. Bao, J.F. Cai, H. Ji, Fast sparsity-based orthogonal dictionary learning for image restoration, in *Proceedings International Conference Computer Vision*, 2013
17. T. Baran, D. Wei, A.V. Oppenheim, Linear programming algorithms for sparse filter design. *IEEE T. Signal Process.* **58**(3), 1605–1617 (2010)
18. R.G. Baraniuk, Compressive sensing. *IEEE Signal Process. Mag.* **24**(4) (2007)
19. M. Baumann, Real-time robust principal component analysis for video surveillance. Master's thesis, ETH Zurich, 2010
20. A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.* **2**(1), 183–202 (2009)
21. S. Becker, J. Bobin, E.J. Candès, NESTA: a fast and accurate first-order method for sparse recovery. *SIAM J. Imag. Sci.* **4**(1), 1–39 (2011)
22. C.R. Berger, Z. Wang, J. Huang, S. Zhou, Application of compressive sensing to sparse channel estimation. *IEEE Commun. Mag.* **48**(11), 164–174 (2010)
23. J.M. Bioucas-Dias, M.A. Figueiredo, A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Trans. Image Process.* **16**(12), 2992–3004 (2007)
24. J. Bobin, J.L. Starck, J.M. Fadili, Y. Moudden, D.L. Donoho, Morphological component analysis: an adaptive thresholding strategy. *IEEE T. Image Process.* **16**(11), 2675–2681 (2007)
25. J.D. Boissonnat, F. Nielsen, R. Nock, Bregman Voronoi diagrams. *Discrete Computat. Geometry* **44**(2), 281–307 (2010)
26. S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **3**(1), 1–122 (2010)
27. P. Breen, Algorithms for sparse approximation. Master's thesis, University of Edinburgh, 2009. Year 4 Project, School of Mathematics
28. L.M. Bregman, The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Comput. Math. Math. Phys.* **7**(3), 200–217 (1967)
29. A. Buades, T.M. Le, J.-M. Morel, L.A. Vese, Fast cartoon + texture image filters. *IEEE T. Image Process.* **19**(8), 1978–1986 (2010)
30. J. Bush, Bregman algorithms. Master's thesis, University of Santa Barbara, CA, USA, 2011. Senior Thesis
31. C.F. Cadieu, B.A. Olshausen, Learning transformational invariants from natural movies, in *Proceedings NIPS*, pp. 209–216 (2008)
32. J.F. Cai, E.J. Candès, Z. Shen, A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* **20**(4), 1956–1982 (2010)
33. X. Cai, R. Chan, T. Zeng, A two-stage image segmentation method using a convex variant of the Mumford-Shah model and thresholding. *SIAM J. Imag. Sci.* **6**(1), 368–390 (2013)
34. E. Candès, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory* **52**(2), 489–509 (2006)
35. E. Candès, T. Tao, Decoding by linear programming. *IEEE T. Inf. Theory* **51**(12), 4203–4215 (2005)
36. E. Candès, T. Tao, The Dantzig selector: statistical estimation when P is much larger than N . *Ann. Stat.* **35**(6), 2313–2351 (2007)
37. E.J. Candès, The restricted isometry property and its implications for compressed sensing. *C. R. Math.* **346**(9), 589–592 (2008)

38. E.J. Candès, X. Li, Y. Ma, J. Wright, Robust principal component analysis? J. ACM (JACM) **58**(3), 11 (2011)
39. E.J. Candès, Y. Plan, Matrix completion with noise. Proc. IEEE **98**(6), 925–936 (2010)
40. E.J. Candès, B. Recht, Exact matrix completion via convex optimization. Found. Comput. Math. **9**(6), 717–772 (2009)
41. E.J. Candès, T. Tao, Near-optimal signal recovery from random projections: Universal encoding strategies? IEEE T. Inf. Theory **52**(12), 5406–5425 (2006)
42. E.J. Candès, M.B. Wakin, An introduction to compressive sampling. IEEE Signal Process. Magz. 21–30 (2008)
43. S. Cao, Q. Wang, Y. Yuan, J. Yu, Anomaly event detection method based on compressive sensing and iteration in wireless sensor networks. J. Netw. **9**(3), 711–718 (2014)
44. C. Caramanis, S. Sanghavi, *Large Scale Optimization* (The University of Texas at Austin, Lecture 24 of EE381V Course, 2012). http://users.ece.utexas.edu/~sanghavi/courses/scribed_notes/Lecture_24_Scribe_Notes.pdf
45. V. Caselles, A. Chambolle, M. Novaga, Total variation in imaging, in *Handbook of Mathematical Methods in Imaging*, pp. 1016–1057 (Springer Verlag, 2011)
46. Y. Censor, A. Lent, An iterative row-action method for interval convex programming. J. Optim. Theory Appl. **34**(3), 321–353 (1981)
47. A. Chambolle, P.L. Lions, Image recovery via total variation minimization and related problems. Numer. Math. **76**(2), 167–188 (1997)
48. T. Chan, S. Esedoglu, F. Park, A. Yip, Recent developments in total variation image restoration. Math. Models Comput. Vision **17**, (2005)
49. T.F. Chan, S. Esedoglu, Aspects of total variation regularized L1 function approximation. SIAM J. Appl. Math. **65**(5), 1817–1837 (2005)
50. T.F. Chan, S. Esedoglu, F.E. Park, Image decomposition combining staircase reduction and texture extraction. J. Visual Commun. Image Represent. **18**(6), 464–486 (2007)
51. V. Chandrasekaran, S. Sanghavi, P.A. Parrilo, A.S. Willsky, Rank-sparsity incoherence for matrix decomposition. SIAM J. Optim. **21**(2), 572–596 (2011)
52. P. Chatterjee, I.P.P. Milanfar, *Denoising using the K-SVD Method* (EE 264 Course, University of California at Santa Cruz, 2007). https://users.soe.ucsc.edu/~priyam/ksvd_report.pdf
53. K.M. Cheman, Optimization techniques for solving basis pursuit problems. Master's thesis, North Carolina State Univ., Raleigh, NC, USA, 2006
54. F. Chen, C.C.P. Wei, Y.C. Wang, Low-rank matrix recovery with structural incoherence for robust face recognition, in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, pp. 2618–2625 (2012)
55. S.S. Chen, D.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit. SIAM J. Sci. Comput. **20**(1), 33–61 (1998)
56. E.C. Chi, H. Zhou, G.K. Chen, D.O. Del Vecchio, K. Lange, Genotype imputation via matrix completion. Genome Res. **23**(3), 509–518 (2013)
57. M.G. Christensen, S.H. Jensen, On compressed sensing and its application to speech and audio signals, in *Proceedings 43th IEEE Asilomar Conference on Signals, Systems and Computers*, pp. 356–360 (2009)
58. I. Cimrak, Analysis of the bounded variation and the G regularization for nonlinear inverse problems. Math. Meth. Appl. Sci. **33**(9), 1102–1111 (2010)
59. A. Cohen, W. Dahmen, I. Daubechies, R. DeVore, Harmonic analysis of the space BV. Revista Matemática Iberoamericana **19**(1), 235–262 (2003)
60. R. Coifman, F. Geshwind, Y. Meyer, Noiselets. Appl. Comput. Harmonic Anal. **10**, 27–44 (2001)
61. P.L. Combettes, J.C. Pesquet, Proximal splitting methods in signal processing, in *Fixed-point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212 (Springer, 2011)
62. S.B. Damelin, W. Jr, Miller, *The Mathematics of Signal Processing* (Cambridge University Press, 2012)
63. I. Daubechies, M. Fornasier, I. Loris, Accelerated projected gradient method for linear inverse problems with sparsity constraints. J. Fourier Anal. Appl. **14**(5–6), 764–792 (2008)

64. M.A. Davenport, M.F. Duarte, Y.C. Eldar, G. Kutyniok, Introduction to compressed sensing, eds. by Y.C. Eldar, G. Kutyniok. *Compressed Sensing* (Cambridge University Press, 2013)
65. F. De la Torre, M.J. Black, Robust principal component analysis for computer vision, in *Proceedings Eighth IEEE International Conference on Computer Vision, (ICCV)*, vol. 1, pp. 362–369 (2001)
66. G. Dogan, P. Morin, R.H. Nochetto, A variational shape optimization approach for image segmentation with a Mumford-Shah functional. *SIAM J. Sci. Comput.* **30**(6), 3028–3049 (2008)
67. D. Donoho, J. Tanner, Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing. *Philos. Trans. Roy. Soc. A: Math. Phys. Eng. Sci.* **367**(1906), 4273–4293 (2009)
68. D.L. Donoho, Compressed sensing. *IEEE T. Inf. Theory* **52**(4), 1289–1306 (2006)
69. D.L. Donoho, M. Gavish, A. Montanari, The phase transition of matrix recovery from gaussian measurements matches the minimax MSE of matrix denoising. *Proc. Natl. Acad. Sci.* **110**(21), 8405–8410 (2013)
70. D.L. Donoho, J. Tanner, Precise undersampling theorems. *Proc. IEEE* **98**(6), 913–924 (2010)
71. D.L. Donoho, Y. Tsaig, Fast solution of L1-norm minimization problems when the solution may be sparse. *IEEE T. Inf. Theory* **54**(11), 4789–4812 (2006)
72. I. Drori, D.L. Donoho, Solution of L1 minimization problems by LARS/homotopy methods, in *Proceedings IEEE International Conference Acoustics, Speech and Signal Processing, ICASSP*, vol. 3 (2006)
73. M.F. Duarte, Y.C. Eldar, Structured compressed sensing: from theory to applications. *IEEE Trans. Signal Process.* **59**(9), 4053–4085 (2011)
74. J. Eckstein, Splitting methods for monotone operators with applications to parallel optimization. Ph.D. thesis, MIT, 1989
75. B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression. *Ann. Stat.* **32**(2), 407–451 (2004)
76. M. Elad, *Sparse and Redundant Representations* (Springer Verlag, 2010)
77. M. Elad, Sparse and redundant representation modeling—what next? *IEEE Signal Process. Lett.* **19**(12), 922–928 (2012)
78. M. Elad, M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries. *IEEE T. Image Process.* **15**(12), 3736–3745 (2006)
79. M. Elad, M.A. Figueiredo, Y. Ma, On the role of sparse and redundant representations in image processing. *Proc. IEEE* **98**(6), 972–982 (2010)
80. J. Ender, A brief review of compressive sensing applied to radar, in *Proceedings 14th International Radar Symposium (IRS)*, pp. 3–16 (2013)
81. M.A. Figueiredo, R.D. Nowak, S.J. Wright, Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. *IEEE J. Selected Topics in Signal Process.* **1**(4), 586–597 (2007)
82. S. Foucart, H. Rauhut, *A Mathematical Introduction to Compressive Sensing* (Birkhäuser, 2010)
83. K. Fountoulakis, J. Gondzio, P. Zhlobich, Matrix-free interior point method for compressed sensing problems. *Math. Programm. Comput.* 1–31 (2012)
84. J. Friedman, T. Hastie, R. Tibshirani, *A note on the group Lasso and a sparse group Lasso* (Dept. Statistics, Stanford University, 2010) arXiv preprint [arXiv:1001.0736](https://arxiv.org/abs/1001.0736)
85. L. Gan, Block compressed sensing of natural images, in *Proceedings IEEE International Conference Digital Signal Processing*, pp. 403–406 (2007)
86. S. Gandy, I. Yamada, Convex optimization techniques for the efficient recovery of a sparsely corrupted low-rank matrix. *J. Math-for-Indus.* **2**(5), 147–156 (2010)
87. V. Ganesan, A study of compressive sensing for application to structural health monitoring. Master's thesis, University of Central Florida, 2014
88. J. Gao, Q. Shi, T.S. Caetano, Dimensionality reduction via compressive sensing. *Pattern Recogn. Lett.* **33**(9), 1163–1170 (2012)
89. P. Getreuer, Chan-Vese segmentation. *Image Processing On Line* (2012)

90. P. Getreuer, Rudin-osher-fatemi total variation denoising using split Bregman. *Image Processing On Line* **10** (2012)
91. J.R. Gilbert, C. Moler, R. Schreiber, Sparse matrices in MATLAB: design and implementation. *SIAM J. Matrix Anal. Appl.* **13**(1), 333–356 (1992)
92. J. Gilles, Noisy image decomposition: a new structure, texture and noise model based on local adaptivity. *J. Math. Imag. Vision* **28**(3), 285–295 (2007)
93. J. Gilles, Image decomposition: theory, numerical schemes, and performance evaluation. *Adv. Imag. Electron Phys.* **158**, 89–137 (2009)
94. T. Goldstein, B. O'Donoghue, S. Setzer, R. Baraniuk, Fast alternating direction optimization methods. *SIAM J. Imag. Sci.* **7**(3), 1588–1623 (2014)
95. T. Goldstein, S. Osher, The split Bregman method for L1 regularized problems. *SIAM J. Imag. Sci.* **2**(2), 323–343 (2009)
96. Y. Gousseau, J.M. Morel, Are natural images of bounded variation? *SIAM J. Math. Anal.* **33**(3), 634–648 (2001)
97. X. Guo, S. Li, X. Cao, Motion matters: a novel framework for compressing surveillance videos, in *Proceedings 21st ACM International Conference on Multimedia*, pp. 549–552 (2013)
98. M. Hardt, R. Meka, P. Raghavendra, B. Weitz, *Computational Limits for Matrix Completion* (IBM Research Almaden, 2014). arXiv preprint [arXiv:1402.2331](https://arxiv.org/abs/1402.2331)
99. J. Haupt, W.U. Bajwa, M. Rabbat, R. Nowak, Compressed sensing for networked data. *IEEE Signal Process. Mag.* **25**(2), 92–101 (2008)
100. K. Hayashi, M. Nagahara, T. Tanaka, A user's guide to compressed sensing for communications systems. *IEICE Trans. Commun.* **96**(3), 685–712 (2013)
101. A.E. Hoerl, R.W. Kennard, Ridge regression: biased estimation for nonorthogonal problems. *Technometrics* **12**(1), 55–67 (1970)
102. H. Huang, S. Misra, W. Tang, H. Barani, H. Al-Azzawi, *Applications of Compressed Sensing in Communications Networks* (New Mexico State University, USA, 2013). arXiv preprint [arXiv:1305.3002](https://arxiv.org/abs/1305.3002)
103. Y. Huang, J.L. Beck, S. Wu, H. Li, Robust Bayesian compressive sensing for signals in structural health monitoring. *Comput.-Aid. Civil Infrastruct. Eng.* **29**(3), 160–179 (2014)
104. H. Jung, K. Sung, K.S. Nayak, E.Y. Kim, J.C. Ye, K-t FOCUSS: a general compressed sensing framework for high resolution dynamic MRI. *Magn. Reson. Med.* **61**(1), 103–116 (2009)
105. O. Kardani, A.V. Lyamin, K. Krabbenhoft, A comparative study of preconditioning techniques for large sparse systems arising in finite element analysis. *IAENG Intl. J. Appl. Math.* **43**(4), 1–9 (2013)
106. S.J. Kim, K. Koh, M. Lustig, S. Boyd, D. Gorinevsky, An interior-point method for large-scale L1-regularized least squares. *IEEE J. Sel. Top. Sign. Process.* **1**(4), 606–617 (2007)
107. M. Kowalski, B. Torr sani, Structured sparsity: from mixed norms to structured shrinkage, in *Proceedings SPARS'09-Signal Processing with Adaptive Sparse Structured Representations* (2009)
108. G. Kutyniok, Theory and applications of compressed sensing. *GAMM-Mitteilungen* **36**(1), 79–101 (2013)
109. V. Le Guen, *Cartoon+ Texture Image Decomposition by the TV-L1 Model*. IPOL, *Image Processing On Line* (2014). http://www.ipol.im/pub/algo/gjmr_line_segment_detector/
110. F. Lenzen, F. Becker, J. Lellmann, Adaptive second-order total variation: an approach aware of slope discontinuities, in *Scale Space and Variational Methods in Computer Vision*, pp. 61–73 (Springer, 2013)
111. Z. Li, Y. Zhu, H. Zhu, M. Li, Compressive sensing approach to urban traffic sensing, in *Proceedings IEEE International Conference Distributed Computing Systems, (ICDCS)*, pp. 889–898 (2011)
112. H. Liebgott, A. Basarab, D. Kouame, O. Bernard, D. Friboulet, Compressive sensing in medical ultrasound, in *Proceedings IEEE International Ultrasonics Symposium, (IUS)*, pp. 1–6 (2012)
113. F. Lin, Z. Hu, S. Hou, J. Yu, C. Zhang, N. Guo, K. Currie, Cognitive radio network as wireless sensor network (ii): Security consideration, in *Proceedings IEEE National Aerospace and Electronics Conference, (NAECON)*, pp. 324–328 (2011)

114. Z. Lin. *Some Software Packages for Partial SVD Computation* (School of EECS, Peking University, 2011). arXiv preprint [arXiv:1108.1548](https://arxiv.org/abs/1108.1548)
115. Z. Lin, M. Chen, Y. Ma, *The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-rank Matrices* (Microsoft Research Asia, 2010). arXiv preprint [arXiv:1009.5055](https://arxiv.org/abs/1009.5055)
116. Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, Y. Ma. *Fast Convex Optimization Algorithms for Exact Recovery of a Corrupted Low-rank Matrix* (Microsoft Research Asia, 2009). http://yima.csl.illinois.edu/psfile/rpca_algorithms.pdf
117. D. Liu, T. Zhou, H. Qian, C. Xu, Z. Zhang, A nearly unbiased matrix completion approach, in *Machine Learning and Knowledge Discovery in Databases*, pp. 210–225 (Springer Verlag, 2013)
118. G. Liu, W. Kang, IDMA-Based compressed sensing for ocean monitoring information acquisition with sensor networks. *Math. Probl. Eng.* **2014**, 1–13 (2014)
119. C. Luo, F. Wu, J. Sun, C.W. Chen, Compressive data gathering for large-scale wireless sensor networks, in *Proceedings 15th ACM Annual International Conference on Mobile Computing and Networking*, pp. 145–156 (2009)
120. M. Lustig, D. Donoho, J.M. Pauly, Sparse MRI: the application of compressed sensing for rapid MR imaging. *Magn. Reson. Med.* **58**(6), 1182–1195 (2007)
121. M. Lysaker, X.C. Tai, Iterative image restoration combining total variation minimization and a second-order functional. *Int. J. Comput. Vision* **66**(1), 5–18 (2006)
122. J. Mairal, M. Elad, G. Sapiro, Sparse representation for color image restoration. *IEEE T. Image Process.* **17**(1), 53–69 (2008)
123. J. Mairal, B. Yu, *Complexity Analysis of the Lasso Regularization Path* (Department of Statistics, University of California at Berkeley, 2012) arXiv preprint [arXiv:1205.0079](https://arxiv.org/abs/1205.0079)
124. D. Mascarenas, A. Cattaneo, J. Theiler, C. Farrar, Compressed sensing techniques for detecting damage in structures. *Struct. Health Monit.* (2013)
125. P. Maurel, J.F. Aujol, G. Peyré, Locally parallel texture modeling. *SIAM J. Imag. Sci.* **4**(1), 413–447 (2011)
126. D. McMorro, Compressive sensing for DoD sensor systems. Technical report (MITRE Corp, 2012)
127. J. Meng, H. Li, Z. Han, Sparse event detection in wireless sensor networks using compressive sensing, in *Proceedings IEEE 43rd Annual Conference Information Sciences and Systems, (CISS)*, pp. 181–185 (2009)
128. Y. Meyer, Oscillating patterns in image processing and nonlinear evolution equations: the fifteenth Dean Jacqueline B. Lewis memorial lectures. *AMS Bookstore* **22** (2001)
129. M. Michenkova, *Numerical Algorithms for Low-rank Matrix Completion Problems* (Swiss Federal Institute of Technology, Zurich, 2011). <http://sma.epfl.ch/~anchpcommon/students/michenkova.pdf>
130. D. Mumford, J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Appl. Math.* **42**(5), 577–685 (1989)
131. S. Mun, J.E. Fowler, Block compressed sensing of images using directional transforms, in *Proceedings IEEE International Conference Image Processing, (ICIP)*, pp. 3021–3024 (2009)
132. M. Nagahara, T. Matsuda, K. Hayashi, Compressive sampling for remote control systems. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **95**(4), 713–722 (2012)
133. D. Needell, J.A. Tropp, CoSaMP: iterative signal recovery from incomplete and inaccurate samples. *Appl. Comput. Harmon. Anal.* **26**(3), 301–321 (2009)
134. Y. Nesterov, Smooth minimization of non-smooth functions. *Math. Programm.* **103**(1), 127–152 (2005)
135. M. Nikolova, A variational approach to remove outliers and impulse noise. *J. Math. Imag. Vision* **20**(1–2), 99–120 (2004)
136. R. Nowak, M. Figueiredo, Fast wavelet-based image deconvolution using the EM algorithm, in *Proceedings 35th Asilomar Conference Signals, Systems and Computers* (2001)
137. S. Oh, *Matrix Completion: Fundamental Limits and Efficient Algorithms*. Ph.D. thesis, Stanford University, 2010

138. B.A. Olshausen, D.J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* **381**(6583), 607–609 (1996)
139. B.A. Olshausen, D.J. Field, Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Res.* **37**(23), 3311–3325 (1997)
140. B.A. Olshausen, D.J. Field, Sparse coding of sensory inputs. *Curr. Opin. Neurobiol.* **14**(4), 481–487 (2004)
141. M.R. Osborne, B. Presnell, B.A. Turlach, A new approach to variable selection in least squares problems. *IMA J. Numer. Anal.* **20**(3), 389–403 (2000)
142. S. Osher, M. Burger, D. Goldfarb, J. Xu, W. Yin, An iterative regularization method for total variation-based image restoration. *Multiscale Model. Simul.* **4**(2), 460–489 (2005)
143. S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.* **79**(1), 12–49 (1988)
144. I. Papusha, *Fast Automatic Background Extraction Via Robust PCA* (Caltech, 2011). http://www.cds.caltech.edu/~ipapusha/pdf/robust_pca_apps.pdf
145. N. Parikh, S. Boyd, Proximal algorithms. *Found. Trends Optim.* **1**(3), 123–231 (2013)
146. J.C. Park, B. Song, J.S. Kim, S.H. Park, H.K. Kim, Z. Liu, W.Y. Song, Fast compressed sensing-based CBCT reconstruction using Barzilai-Borwein formulation for application to on-line IGRT. *Med. Phys.* **39**(3), 1207–1217 (2012)
147. S. Patterson, Y. C. Eldar, I. Keidar, *Distributed Compressed Sensing for Static and Time-varying Networks* (Rensselaer Polytechnic Institute, 2013). arXiv preprint [arXiv:1308.6086](https://arxiv.org/abs/1308.6086)
148. Y. Peng, A. Ganesh, J. Wright, W. Xu, Y. Ma, RASL: robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(11), 2233–2246 (2012)
149. P. Perona, J. Malik, Scale space and edge detection using anisotropic diffusion, in *Proceedings IEEE Computer Society Workshop on Computer Vision*, pp. 16–22 (1987)
150. P. Perona, J. Malik, Scale space and edge detection using anisotropic diffusion. *IEEE T. Pattern Anal. Mach. Intell.* **12**, 629–639 (1990)
151. G. Peyre, *Matrix Completion with Nuclear Norm Minimization*. A Numerical Tour of Signal Processing (2014). http://gpeyre.github.io/numerical-tours/matlab/sparsity_3_matrix_completion/
152. T. Pock, D. Cremers, H. Bischof, A. Chambolle, An algorithm for minimizing the Mumford-Shah functional, in *Proceedings IEEE 12th International Conference on Computer Vision*, pp. 1133–1140 (2009)
153. S. Qaisar, R.M. Bilal, W. Iqbal, M. Naureen, S. Lee, Compressive sensing: from theory to applications, a survey. *J. Commun. Netw.* **15**(5), 443–456 (2013)
154. C. Quinsac, A. Basarab, D. Kouamé, Frequency domain compressive sampling for ultrasound imaging. *Adv. Acoust. Vib.* **1–16**, 2012 (2012)
155. I. Ram, M. Elad, I. Cohen, Image processing using smooth ordering of its patches. *IEEE T. Image Process.* **22**(7), 2764–2774 (2013)
156. M.A. Rasmussen, R. Bro, A tutorial on the Lasso approach to sparse modeling. *Chemom. Intell. Lab. Syst.* **119**, 21–31 (2012)
157. H. Rauhut, Compressive sensing and structured random matrices, ed. by Formasier. *Theoretical Foundations and Numerical Methods for Sparse Recovery* (Walter de Gruyter, Berlin, 2010)
158. J. Richy, Compressive sensing in medical ultrasonography. Master's thesis, Kungliga Tekniska Högskolan, 2012
159. P. Rodríguez, B. Wohlberg, Fast principal component pursuit via alternating minimization, in *Proceedings IEEE International Conference on Image Processing, (ICIP)*, pp. 69–79 (2013)
160. J. Romberg, Imaging via compressive sampling (introduction to compressive sampling and recovery via convex programming). *IEEE Signal Process. Mag.* **25**(2), 14–20 (2008)
161. R. Rubinstein, A.M. Bruckstein, M. Elad, Dictionaries for sparse representation modeling. *Proc. IEEE* **98**(6), 1045–1057 (2010)
162. R. Rubinstein, M. Zibulevsky, M. Elad, Efficient implementation of the K-SVD algorithm and the Batch-OMP method. Technical report, Department of Computer Science, Technion, Israel, 2008. Technical CS–08

163. L.I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena* **60**(1), 259–268 (1992)
164. L. Ryzhik, *Lecture Notes for Math 221* (Stanford University, 2013). <http://math.stanford.edu/~ryzhik/STANFORD/STANF221-13/stanf221-13-notes.pdf>
165. M. Saunders, *PDCO – Primal-dual Interior Methods* (CME 338 Lecture Notes 7, Stanford University, 2013). <https://web.stanford.edu/group/SOL/software/pdco/pdco.pdf>
166. M Schmidt, Least squares optimization with L1-norm regularization. Technical report, University of British Columbia, 2005. Project Report
167. I. Selesnick, *Total Variation Denoising (an MM Algorithm)* (New York University, 2012). http://eeweb.poly.edu/iselesni/lecture_notes/TVDmm/TVDmm.pdf
168. I.W. Selesnick, *Sparse Signal Restoration* (New York University, 2010). http://eeweb.poly.edu/iselesni/lecture_notes/sparse_signal_restoration.pdf
169. I.W. Selesnick, I. Bayram, *Total Variation Filtering* (New York University, 2010). http://eeweb.poly.edu/iselesni/lecture_notes/TVDmm/
170. Y. Shen, Z. Wen, Y. Zhang, Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optim. Meth. Softw.* **29**(2), 239–263 (2014)
171. E.P. Simoncelli, B.A. Olshausen, Natural image statistics and neural representation. *Ann. Rev. Neurosci.* **24**(1), 1193–1216 (2001)
172. J.L. Starck, M. Elad, D. Donoho, Redundant multiscale transforms and their application for morphological component separation. *Adv. Imag. Electron Phys.* **132**(82), 287–348 (2004)
173. J.L. Starck, Y. Moudden, J. Bobin, M. Elad, D.L. Donoho, Morphological component analysis. *Opt. Photon.* 1–15 (2005)
174. J.L. Starck, F. Murtagh, J.M. Fadili, *Sparse Image and Signal Processing* (Cambridge University Press, 2010)
175. D. Sundman, Compressed sensing: algorithms and applications. Master’s thesis, KTH Electrical Engineering, Stockholm, 2012. Licentiate thesis
176. R. Tibshirani, Regression shrinkage and selection via the Lasso. *J. Royal Stat. Soc.* **58**, 267–288 1996). series B
177. J.A. Tropp, S.J. Wright, Computational methods for sparse solution of linear inverse problems. *Proc. IEEE* **98**(6), 948–958 (2010)
178. R. Valentine, *Image Segmentation with the Mumford Shah Functional* (2007). <http://coldstonelabs.org/files/science/math/Intro-MS-Valentine.pdf>
179. S.A. Van De Geer, P. Bühlmann, On the conditions used to prove oracle results for the Lasso. *Electron. J. Stat.* **3**, 1360–1392 (2009)
180. L.A. Vese, S.J. Osher, Modeling textures with total variation minimization and oscillating patterns in image processing. *J. Sci. Comput.* **19**(1–3), 553–572 (2003)
181. S. Wang, Z. Zhang, Colorization by matrix completion, in *Proceedings 26th AAAI Conference on Artificial Intelligence*, pp. 1169–1175 (2012)
182. Z. Wang, M.J. Lai, Z. Lu, J. Ye, *Orthogonal Rank-one Matrix Pursuit for Low Rank Matrix Completion* (The Biodesign Institue, Arizona State University, 2014) arXiv preprint [arXiv:1404.1377](https://arxiv.org/abs/1404.1377)
183. S.J. Wei, X.L. Zhang, J. Shi, G. Xiang, Sparse reconstruction for SAR imaging based on compressed sensing. *Prog. Electromagn. Res.* **109**, 63–81 (2010)
184. J. Weickert, *Anisotropic Diffusion in Image Processing*, vol. 1 (Teubner, Stuttgart, 1998)
185. S. Weisberg, *Applied Linear Regression* (Wiley, 1980)
186. S.J. Wright, R.D. Nowak, A.T. Figueiredo, Sparse reconstruction by separable approximation. *IEEE Trans. Signal Process.* **57**(7), 2479–2493 (2009)
187. H. Xu, C. Caramanis, S. Mannor, Robust regression and Lasso. *IEEE T. Inf. Theory* **56**(7), 3561–3574 (2010)
188. X. Xu, Online robust principal component analysis for background subtraction: a system evaluation on Toyota car data. Master’s thesis, University of Illinois at Urbana-Champaign, 2014
189. Y. Xu, W. Yin, A fast patch-dictionary method for whole-image recovery. Technical report, UCLA, 2013. CAM13-38

190. M. Yan, Y. Yang, S. Osher, Exact low-rank matrix completion from sparsely corrupted entries via adaptive outlier pursuit. *J. Sci. Comput.* **56**(3), 433–449 (2013)
191. S. Yan, C. Wu, W. Dai, M. Ghanem, Y. Guo, Environmental monitoring via compressive sensing, in *Proceedings ACM International Workshop on Knowledge Discovery from Sensor Data*, pp. 61–68 (2012)
192. J. Yang, Y. Zhang, Alternating direction algorithms for L1 problems in compressive sensing. *SIAM J. Sci. Comput.* **33**(1), 250–278 (2011)
193. W. Yin, D. Goldfarb, S. Osher, Total variation based image cartoon-texture decomposition. Technical report, Columbia Univ., Dep. Industrial Eng. & Operation Res., 2005. Rep. CU-CORC-TR-01
194. W. Yin, D. Goldfarb, S. Osher, A comparison of three total variation based texture extraction models. *J. Vis. Commun. Image Represent.* **18**(3), 240–252 (2007)
195. M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables. *J. Royal Stat Soc* **68**(1), 49–67 (2007). series B
196. X. Yuan, J. Yang, *Sparse and low-rank matrix decomposition via alternating direction methods* (Nanjing University, China, 2009). <http://math.nju.edu.cn/~jfyang/files/LRSD-09.pdf>
197. B. Zhang, X. Cheng, N. Zhang, Y. Cui, Y. Li, Q. Liang, Sparse target counting and localization in sensor networks based on compressive sensing, in *Proceedings IEEE INFOCOM*, pp. 2255–2263 (2011)
198. J. Zhang, T. Tan, Brief review of invariant texture analysis methods. *Pattern Recogn.* **35**(3), 735–747 (2002)
199. Z. Zhang, A. Ganesh, X. Liang, Y. Ma, TILT: Transform invariant low-rank textures. *Int. J. Comput. Vis.* **99**(1), 1–24 (2012)
200. C. Zhao, X. Wu, L. Huang, Y. Yao, Y.C. Chang, Compressed sensing based fingerprint identification for wireless transmitters. *Sci. World J.* **1–9**, 2014 (2014)
201. P. Zhao, B. Yu, On model selection consistency of Lasso. *J. Mach. Learn. Res.* **7**, 2541–2563 (2006)
202. S. Zhou, L. Kong, N. Xiu, New bounds for RIC in compressed sensing. *J. Oper. Res. Soc. China* **1**(2), 227–237 (2013)
203. X. Zhou, W. Yu, Low-rank modeling and its applications in medical image analysis. *SPIE Defense Security Sens.* 87500V (2013)
204. Z. Zhou, X. Li, J. Wright, E. Candes, Y. Ma, Stable principal component pursuit, in *Proceedings IEEE International Symposium on Information Theory, (ISIT)*, pp. 1518–1522 (2010)
205. J. Zhu, D. Baron. Performance regions in compressed sensing from noisy measurements, in *Proceedings IEEE 47th Annual Conference on Information Sciences and Systems, (CISS)*, pp. 1–6 (2013)
206. H. Zou, The adaptive Lasso and its oracle properties. *J. Am. Stat. Assoc.* **101**(476), 1418–1429 (2006)
207. H. Zou, T. Hastie, Regularization and variable selection via the elastic net. *J. Royal Stat. Soc.* **67**(2), 301–320 (2005). series B

Digital Signal Processing with Matlab Examples, Volume
3

Model-Based Actions and Sparse Representation

Giron-Sierra, J.M.

2017, XVI, 431 p. 201 illus., 80 illus. in color., Hardcover

ISBN: 978-981-10-2539-6