3 Sparsity and Compressed Sensing

The inherent structure observed in natural data implies that the data admits a sparse representation in an appropriate coordinate system. In other words, if natural data is expressed in a well-chosen basis, only a few parameters are required to characterize the modes that are active, and in what proportion. All of data compression relies on sparsity, whereby a signal is represented more efficiently in terms of the sparse vector of coefficients in a generic transform basis, such as Fourier or wavelet bases. Recent fundamental advances in mathematics have turned this paradigm upside down. Instead of collecting a high-dimensional measurement and then compressing, it is now possible to acquire *compressed* measurements and solve for the sparsest high-dimensional signal that is consistent with the measurements. This so-called *compressed sensing* is a valuable new perspective that is also relevant for complex systems in engineering, with potential to revolutionize data acquisition and processing. In this chapter, we discuss the fundamental principles of sparsity and compression as well as the mathematical theory that enables compressed sensing, all worked out on motivating examples.

Our discussion on sparsity and compressed sensing will necessarily involve the critically important fields of optimization and statistics. Sparsity is a useful perspective to promote *parsimonious* models that avoid overfitting and remain interpretable because they have the minimal number of terms required to explain the data. This is related to Occam's razor, which states that the simplest explanation is generally the correct one. Sparse optimization is also useful for adding robustness with respect to outliers and missing data, which generally skew the results of least-squares regression, such as the SVD. The topics in this chapter are closely related to randomized linear algebra discussed in Section 1.8, and they will also be used in several subsequent chapters. Sparse regression will be explored further in Chapter 4 and will be used in Section 7.3 to identify interpretable and parsimonious nonlinear dynamical systems models from data.

3.1 Sparsity and Compression

Most natural signals, such as images and audio, are highly compressible. This compressibility means that when the signal is written in an appropriate basis only a few modes are active, thus reducing the number of values that must be stored for an accurate representation. Said another way, a compressible signal $\mathbf{x} \in \mathbb{R}^n$ may be written as a sparse vector $\mathbf{s} \in \mathbb{R}^n$ (containing mostly zeros) in a transform basis $\boldsymbol{\Psi} \in \mathbb{R}^{n \times n}$:

$$\mathbf{x} = \boldsymbol{\Psi}\mathbf{s}. \tag{3.1}$$

Specifically, the vector $\mathbf{s}$ is called K-sparse in $\boldsymbol{\Psi}$ if there are exactly K nonzero elements. If the basis $\boldsymbol{\Psi}$ is generic, such as the Fourier or wavelet basis, then only the few active terms in $\mathbf{s}$ are required to reconstruct the original signal $\mathbf{x}$, reducing the data required to store or transmit the signal.

Images and audio signals are both compressible in Fourier or wavelet bases, so that after taking the Fourier or wavelet transform, most coefficients are small and may be set exactly equal to zero with negligible loss of quality. These few active coefficients may be stored and transmitted, instead of the original high-dimensional signal. Then, to reconstruct the original signal in the ambient space (i.e., in pixel space for an image), one need only take the inverse transform. As discussed in Chapter 2, the fast Fourier transform is the enabling technology that makes it possible to efficiently reconstruct an image $\mathbf{x}$ from the sparse coefficients in $\mathbf{s}$. This is the foundation of JPEG compression for images and MP3 compression for audio.

The Fourier modes and wavelets are *generic* or *universal* bases, in the sense that nearly all natural images or audio signals are sparse in these bases. Therefore, once a signal is compressed, one needs only store or transmit the sparse vector $\mathbf{s}$ rather than the entire matrix $\boldsymbol{\Psi}$, since the Fourier and wavelet transforms are already hard-coded on most machines. In Chapter 1 we found that it is also possible to compress signals using the SVD, resulting in a *tailored* basis. In fact, there are two ways that the SVD can be used to compress an image: 1) we may take the SVD of the image directly and only keep the dominant columns of $\mathbf{U}$ and $\mathbf{V}$ (Section 1.2), or 2) we may represent the image as a linear combination of *eigen* images, as in the eigenface example (Section 1.6). The first option is relatively inefficient, as the basis vectors $\mathbf{U}$ and $\mathbf{V}$ must be stored. However, in the second case, a tailored basis $\mathbf{U}$ may be computed and stored once, and then used to compress an entire class of images, such as human faces. This tailored basis has the added advantage that the modes are interpretable as correlation features that may be useful for learning. It is important to note that both the Fourier basis $\mathcal{F}$ and the SVD basis $\mathbf{U}$ are unitary transformations, which will become important in the following sections.

Although the majority of compression theory has been driven by audio, image, and video applications, there are many implications for engineering systems. The solution to a high-dimensional system of differential equations typically evolves on a low-dimensional manifold, indicating the existence of coherent structures that facilitate sparse representation. Even broadband phenomena, such as turbulence, may be instantaneously characterized by a sparse representation. This has a profound impact on how to sense and compute, as will be described throughout this chapter and the remainder of the book.

Example: Image Compression

Compression is relatively simple to implement on images, as described in Section 2.6 and revisited here (see Fig. 3.1). First, we load an image, convert to grayscale, and plot:

```
A=imread('jelly', 'jpeg'); % Load image
Abw=rgb2gray(A);           % Convert image to grayscale
imshow(Abw).               % Plot image
```

Next, we take the fast Fourier transform and plot the coefficients on a logarithmic scale:

```
At=fft2(Abw);
F = log(abs(fftshift(At))+1);  % put FFT on log-scale
imshow(mat2gray(F),[]);
```

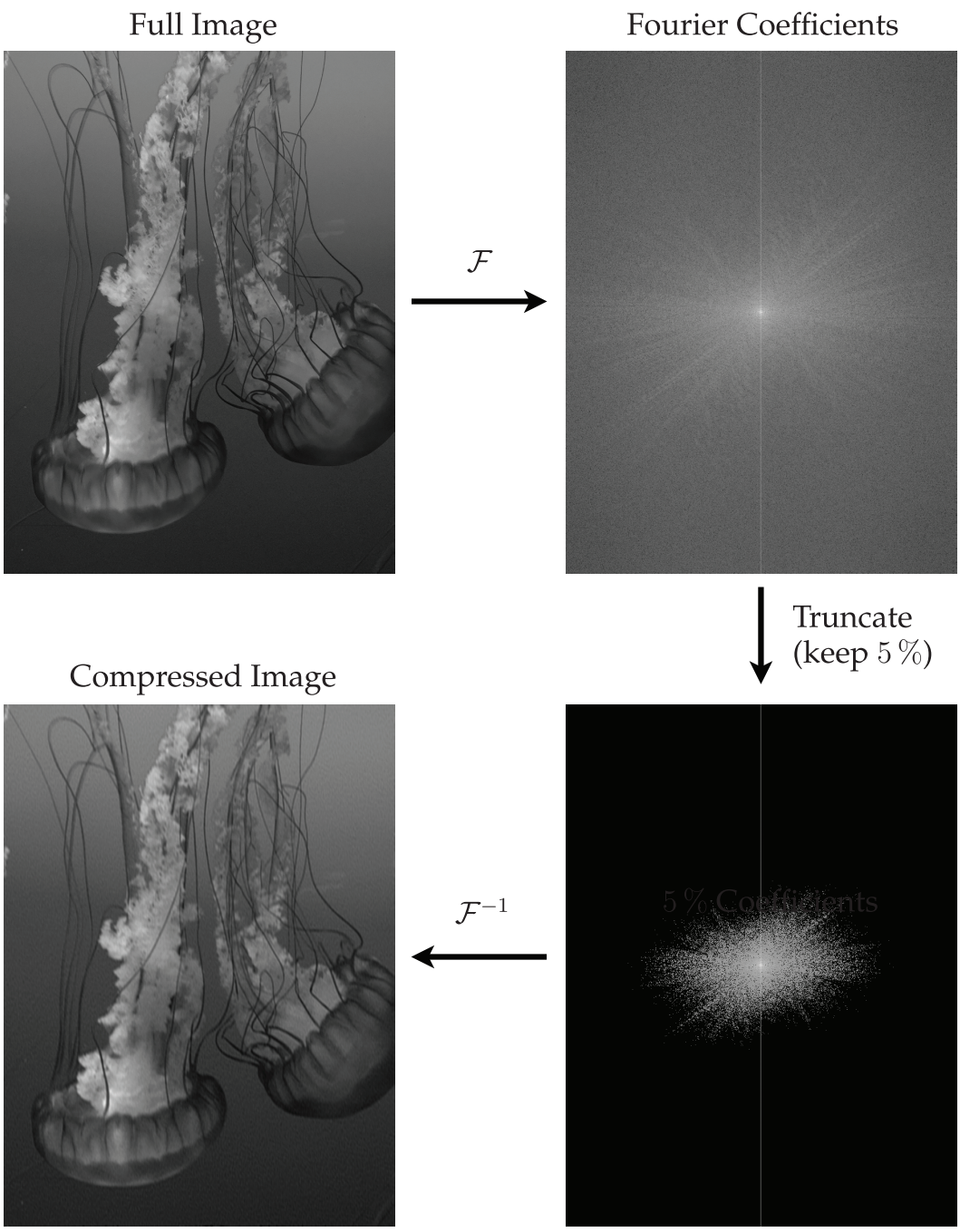

Figure 3.1 Illustration of compression with the fast Fourier transform (FFT) $\mathcal{F}$.

To compress the image, we first arrange all of the Fourier coefficients in order of magnitude and decide what percentage to keep (in this case 5 %). This sets the threshold for truncation:

```
Bt = sort(abs(At(:)));
keep = 0.05;
thresh = Bt(floor((1-keep)*length(Bt)));
ind = abs(At)>thresh;
Atlow = At.*ind;
```

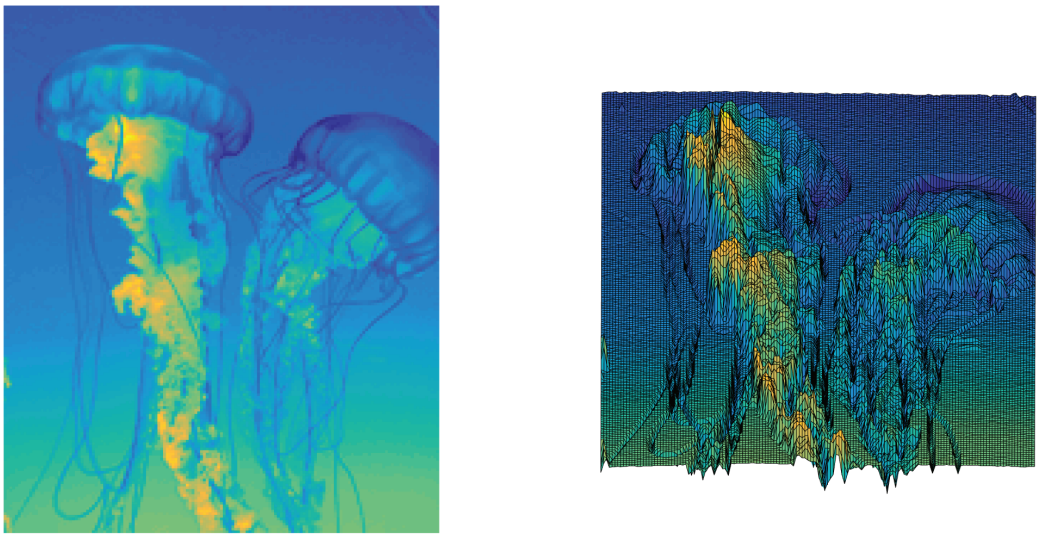

Figure 3.2 Compressed image (left), and viewed as a surface (right).

Finally, we plot the compressed image by taking the inverse FFT (iFFT):

```
Alow=uint8(ifft2(Atlow));
imshow(Alow)
```

To understand the role of the sparse Fourier coefficients in a compressed image, it helps to view the image as a surface, where the height of a point is given by the brightness of the corresponding pixel. This is shown in Fig. 3.2. Here we see that the surface is relatively simple, and may be represented as a sum of a few spatial Fourier modes.

```
Anew = imresize(Abw,.2);
surf(double(Anew));
shading flat, view(-168,86)
```

Why Signals Are Compressible: The Vastness of Image Space

It is important to note that the compressibility of images is related to the overwhelming dimensionality of image space. For even a simple 20×20 pixel black and white image, there are 2^{400} distinct possible images, which is larger than the number of nucleons in the known universe. The number of images is considerably more staggering for higher resolution images with greater color depth.

In the space of one megapixel images (i.e., 1000×1000 pixels), there is an image of us each being born, of me typing this sentence, and of you reading it. However vast the space of these *natural* images, they occupy a tiny, minuscule fraction of the total image space. The majority of the images in image space represent random noise, resembling television static. For simplicity, consider grayscale images, and imagine drawing a random number for the gray value of each of the pixels. With exceedingly high probability, the resulting image will look like noise, with no apparent significance. You could draw these random images for an

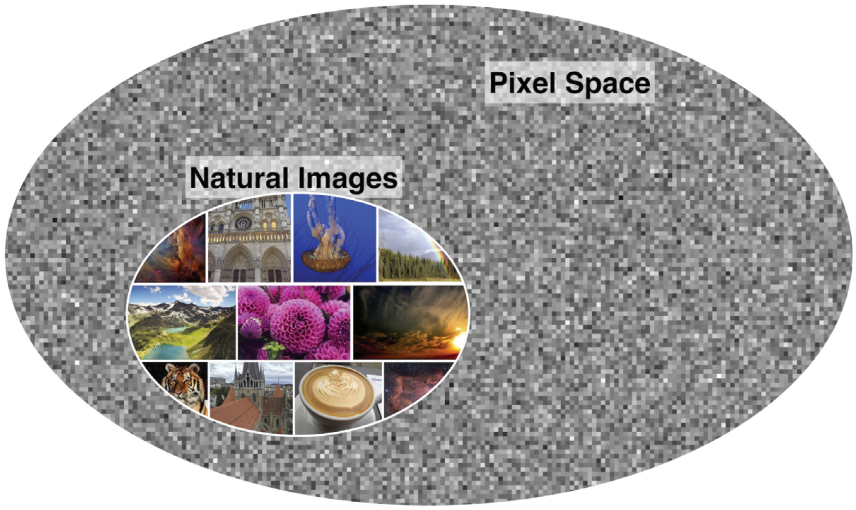

Figure 3.3 Illustration of the vastness of image (pixel) space, with natural images occupying a vanishingly small fraction of the space.

entire lifetime and never find an image of a mountain, or a person, or anything physically recognizable[1].

In other words, natural images are extremely rare in the vastness of image space, as illustrated in Fig. 3.3. Because so many images are unstructured or random, most of the dimensions used to encode images are only necessary for these random images. These dimensions are redundant if all we cared about was encoding natural images. An important implication is that the images we care about (i.e., natural images) are highly compressible, if we find a suitable transformed basis where the redundant dimensions are easily identified.

3.2 Compressed Sensing

Despite the considerable success of compression in real-world applications, it still relies on having access to full high-dimensional measurements. The recent advent of compressed sensing [150, 112, 111, 113, 115, 109, 39, 114, 40] turns the compression paradigm upside down: instead of collecting high-dimensional data just to compress and discard most of the information, it is instead possible to collect surprisingly few *compressed* or *random* measurements and then infer what the sparse representation is in the transformed basis. The idea behind compressed sensing is relatively simple to state mathematically, but until recently finding the sparsest vector consistent with measurements was a non-polynomial (NP) hard problem. The rapid adoption of compressed sensing throughout the engineering and applied sciences rests on the solid mathematical framework[2] that provides conditions

[1] The vastness of signal space was described in Borges's "The Library of Babel" in 1944, where he describes a library containing all possible books that could be written, of which actual coherent books occupy a nearly immeasurably small fraction [69]. In Borges's library, there are millions of copies of this very book, with variations on this single sentence. Another famous variation on this theme considers that given enough monkeys typing on enough typewriters, one would eventually recreate the works of Shakespeare. One of the oldest related descriptions of these combinatorially large spaces dates back to Aristotle.

[2] Interestingly, the incredibly important collaboration between Emmanuel Candès and Terrance Tao began with them discussing the odd properties of signal reconstruction at their kids' daycare.

for when it is possible to reconstruct the full signal with high probability using convex algorithms.

Mathematically, compressed sensing exploits the sparsity of a signal in a generic basis to achieve full signal reconstruction from surprisingly few measurements. If a signal $\mathbf{x}$ is K-sparse in $\boldsymbol{\Psi}$, then instead of measuring $\mathbf{x}$ directly (n measurements) and then compressing, it is possible to collect dramatically fewer randomly chosen or *compressed* measurements and then solve for the nonzero elements of $\mathbf{s}$ in the transformed coordinate system. The measurements $\mathbf{y} \in \mathbb{R}^p$, with $K < p \ll n$ are given by

$$\mathbf{y} = \mathbf{C}\mathbf{x}. \tag{3.2}$$

The measurement matrix[3] $\mathbf{C} \in \mathbb{R}^{p \times n}$ represents a set of p linear measurements on the state $\mathbf{x}$. The choice of measurement matrix $\mathbf{C}$ is of critical importance in compressed sensing, and is discussed in Section 3.4. Typically, measurements may consist of random projections of the state, in which case the entries of $\mathbf{C}$ are Gaussian or Bernoulli distributed random variables. It is also possible to measure individual entries of $\mathbf{x}$ (i.e., single pixels if $\mathbf{x}$ is an image), in which case $\mathbf{C}$ consists of random rows of the identity matrix.

With knowledge of the sparse vector $\mathbf{s}$ it is possible to reconstruct the signal $\mathbf{x}$ from (3.1). Thus, the goal of compressed sensing is to find the sparsest vector $\mathbf{s}$ that is consistent with the measurements $\mathbf{y}$:

$$\mathbf{y} = \mathbf{C}\boldsymbol{\Psi}\mathbf{s} = \boldsymbol{\Theta}\mathbf{s}. \tag{3.3}$$

The system of equations in (3.3) is underdetermined since there are infinitely many consistent solutions $\mathbf{s}$. The *sparsest* solution $\hat{\mathbf{s}}$ satisfies the following optimization problem:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} \|\mathbf{s}\|_0 \ \text{ subject to } \ \mathbf{y} = \mathbf{C}\boldsymbol{\Psi}\mathbf{s}, \tag{3.4}$$

where $\|\cdot\|_0$ denotes the ℓ_0 pseudo-norm, given by the number of nonzero entries; this is also referred to as the cardinality of $\mathbf{s}$.

The optimization in (3.4) is non-convex, and in general the solution can only be found with a brute-force search that is combinatorial in n and K. In particular, all possible K-sparse vectors in $\mathbb{R}^n$ must be checked; if the exact level of sparsity K is unknown, the search is even broader. Because this search is combinatorial, solving (3.4) is intractable for even moderately large n and K, and the prospect of solving larger problems does not improve with Moore's law of exponentially increasing computational power.

Fortunately, under certain conditions on the measurement matrix $\mathbf{C}$, it is possible to relax the optimization in (3.4) to a convex ℓ_1-minimization [112, 150]:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} \|\mathbf{s}\|_1 \ \text{ subject to } \ \mathbf{y} = \mathbf{C}\boldsymbol{\Psi}\mathbf{s}, \tag{3.5}$$

where $\|\cdot\|_1$ is the ℓ_1 norm, given by

$$\|\mathbf{s}\|_1 = \sum_{k=1}^{n} |s_k|. \tag{3.6}$$

[3] In the compressed sensing literature, the measurement matrix is often denoted $\boldsymbol{\Phi}$; instead, we use $\mathbf{C}$ to be consistent with the output equation in control theory. $\boldsymbol{\Phi}$ is also already used to denote DMD modes in Chapter 7.

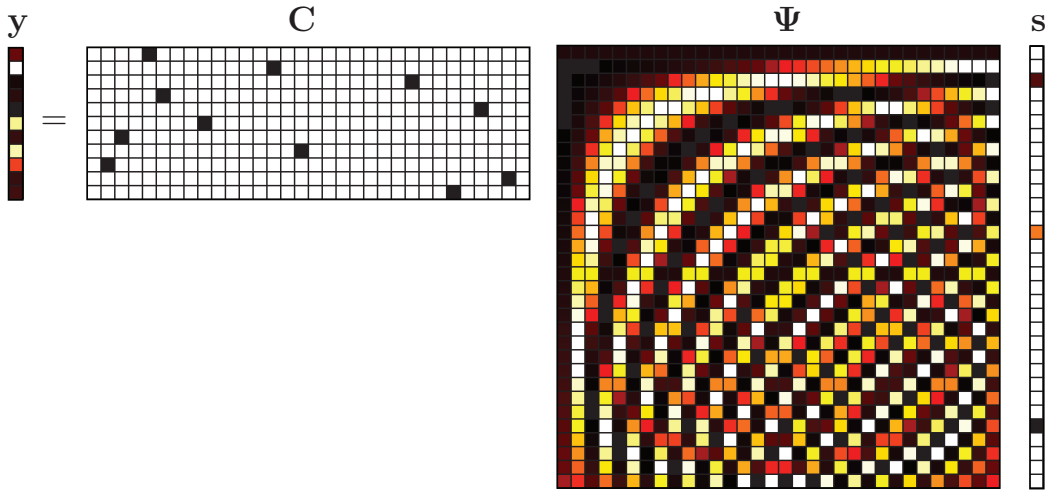

Figure 3.4 Schematic of measurements in the compressed sensing framework.

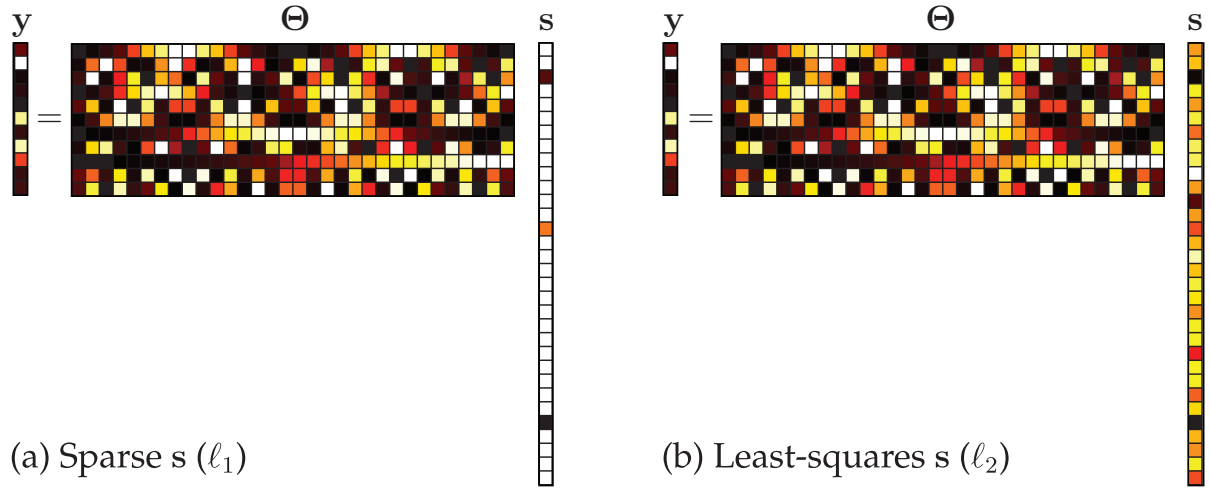

Figure 3.5 ℓ_1 and ℓ_2 minimum norm solutions to compressed sensing problem. The difference in solutions for this regression are further considered in Chapter 4.

The ℓ_1 norm is also known as the taxicab or Manhattan norm because it represents the distance a taxi would take between two points on a rectangular grid. The overview of compressed sensing is shown schematically in Fig. 3.4. The ℓ_1 minimum-norm solution is sparse, while the ℓ_2 minimum norm solution is not, as shown in Fig. 3.5.

There are very specific conditions that must be met for the ℓ_1-minimization in (3.5) to converge with high probability to the sparsest solution in (3.4) [109, 111, 39]. These will be discussed in detail in Sec. 3.4, although they may be summarized as:

1. The measurement matrix $\mathbf{C}$ must be *incoherent* with respect to the sparsifying basis $\mathbf{\Psi}$, meaning that the rows of $\mathbf{C}$ are not correlated with the columns of $\mathbf{\Psi}$,
2. The number of measurements p must be sufficiently large, on the order of

$$p \approx \mathcal{O}(K \log(n/K)) \approx k_1 K \log(n/K). \tag{3.7}$$

The constant multiplier k_1 depends on how incoherent $\mathbf{C}$ and $\mathbf{\Psi}$ are.

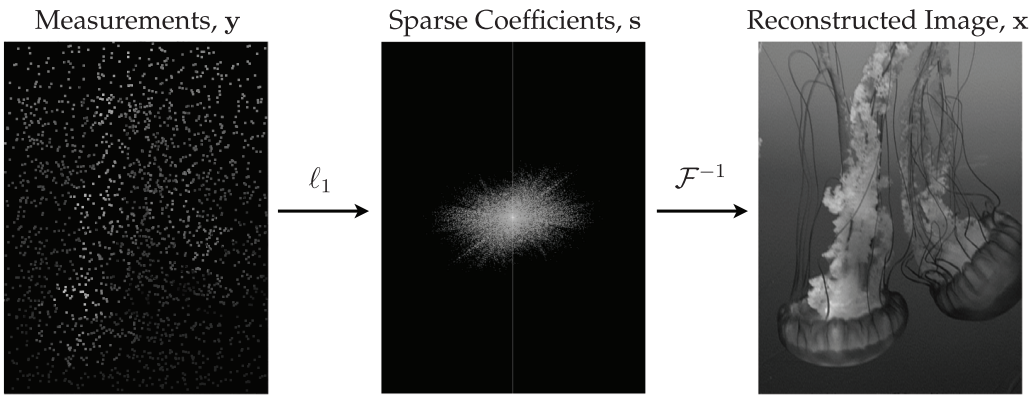

Figure 3.6 Schematic illustration of compressed sensing using ℓ_1 minimization. Note, this is a dramatization, and is not actually based on a compressed sensing calculation. Typically, compressed sensing of images requires a significant number of measurements and is computationally prohibitive.

Roughly speaking, these two conditions guarantee that the matrix $\mathbf{C\Psi}$ acts as a unitary transformation on K sparse vectors $\mathbf{s}$, preserving relative distances between vectors and enabling almost certain signal reconstruction with ℓ_1 convex minimization. This is formulated precisely in terms of the restricted isometry property (RIP) in Sec. 3.4.

The idea of compressed sensing may be counterintuitive at first, especially given classical results on sampling requirements for exact signal reconstruction. For instance, the Shannon-Nyquist sampling theorem [486, 409] states that perfect signal recovery requires that it is sampled at twice the rate of the highest frequency present. However, this result only provides a strict bound on the required sampling rate for signals with broadband frequency content. Typically, the only signals that are truly broadband are those that have already been compressed. Since an uncompressed signal will generally be sparse in a transform basis, the Shannon-Nyquist theorem may be relaxed, and the signal may be reconstructed with considerably fewer measurements than given by the Nyquist rate. However, even though the number of measurements may be decreased, compressed sensing does still rely on precise *timing* of the measurements, as we will see. Moreover, the signal recovery via compressed sensing is not strictly speaking guaranteed, but is instead possible with high probability, making it foremost a statistical theory. However, the probability of successful recovery becomes astronomically large for moderate sized problems.

Disclaimer

A rough schematic of compressed sensing is shown in Fig. 3.6. However, this schematic is a dramatization, and is not actually based on a compressed sensing calculation since using compressed sensing for image reconstruction is computationally prohibitive. It is important to note that for the majority of applications in imaging, compressed sensing is not practical. However, images are often still used to motivate and explain compressed sensing because of their ease of manipulation and our intuition for pictures. In fact, we are currently guilty of this exact misdirection.

Upon closer inspection of this image example, we are analyzing an image with 1024×768 pixels and approximately $5\,\%$ of the Fourier coefficients are required for accurate compression. This puts the sparsity level at $K = 0.05 \times 1024 \times 768 \approx 40,000$. Thus,

a back of the envelope estimate using (3.7), with a constant multiplier of $k_1 = 3$, indicates that we need $p \approx 350,000$ measurements, which is about 45 % of the original pixels. Even if we had access to these 45 % random measurements, inferring the correct sparse vector of Fourier coefficients is computationally prohibitive, much more so than the efficient FFT based image compression in Section 3.1.

Compressed sensing for images is typically only used in special cases where a reduction of the number of measurements is significant. For example, an early application of compressed sensing technology was for infant MRI (magnetic resonance imaging), where reduction of the time a child must be still could reduce the need for dangerous heavy sedation.

However, it is easy to see that the number of measurements p scales with the sparsity level K, so that if the signal is *more* sparse, then fewer measurements are required. The viewpoint of sparsity is still valuable, and the mathematical innovation of convex relaxation of combinatorially hard ℓ_0 problems to convex ℓ_1 problems may be used much more broadly than for compressed sensing of images.

Alternative Formulations

In addition to the ℓ_1-minimization in (3.5), there are alternative approaches based on *greedy algorithms* [525, 526, 528, 527, 530, 243, 529, 207, 531, 205, 398, 206] that determine the sparse solution of (3.3) through an iterative matching pursuit problem. For instance, the compressed sensing matching pursuit (CoSaMP) [398] is computationally efficient, easy to implement, and freely available.

When the measurements $\mathbf{y}$ have additive noise, say white noise of magnitude ε, there are variants of (3.5) that are more robust:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} \|\mathbf{s}\|_1, \quad \text{subject to} \quad \|\mathbf{C\Psi s} - \mathbf{y}\|_2 < \varepsilon. \tag{3.8}$$

A related convex optimization is the following:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmin}} \|\mathbf{C\Psi s} - \mathbf{y}\|_2 + \lambda \|\mathbf{s}\|_1, \tag{3.9}$$

where $\lambda \geq 0$ is a parameter that weights the importance of sparsity. Eqs. (3.8) and (3.9) are closely related [528].

3.3 Compressed Sensing Examples

This section explores concrete examples of compressed sensing for sparse signal recovery. The first example shows that the ℓ_1 norm promotes sparsity when solving a generic underdetermined system of equations, and the second example considers the recovery of a sparse two-tone audio signal with compressed sensing.

ℓ_1 and Sparse Solutions to an Underdetermined System

To see the sparsity promoting effects of the ℓ_1 norm, we consider a generic underdetermined system of equations. We build a matrix system of equations $\mathbf{y} = \mathbf{\Theta s}$ with $p = 200$ rows (measurements) and $n = 1000$ columns (unknowns). In general, there are infinitely many solutions $\mathbf{s}$ that are consistent with these equations, unless we are very unfortunate

and the row equations are linearly dependent while the measurements are inconsistent in these rows. In fact, this is an excellent example of the probabilistic thinking used more generally in compressed sensing: if we generate a linear system of equations at random, that has sufficiently many more unknowns than knowns, then the resulting equations will have infinitely many solutions with *high probability*.

In MATLAB, it is straightforward to solve this underdetermined linear system for both the minimum ℓ_1 norm and minimum ℓ_2 norm solutions. The minimum ℓ_2 norm solution is obtained using the pseudo-inverse (related to the SVD from Chapters 1 and 4). The minimum ℓ_1 norm solution is obtained via the **cvx** (ConVeX) optimization package. Fig. 3.7 shows that the ℓ_1-minimum solution is in fact sparse (with most entries being nearly zero), while the ℓ_2-minimum solution is *dense*, with a bit of energy in each vector coefficient.

Code 3.1 Solutions to underdetermined linear system $\mathbf{y} = \mathbf{\Theta s}$.

```
% Solve y = Theta * s for "s"
n = 1000;  % dimension of s
p = 200;   % number of measurements, dim(y)
Theta = randn(p,n);
y = randn(p,1);

% L1 minimum norm solution s_L1
cvx_begin;
    variable s_L1(n);
    minimize( norm(s_L1,1) );
    subject to
        Theta*s_L1 == y;
cvx_end;

s_L2 = pinv(Theta)*y;  % L2 minimum norm solution s_L2
```

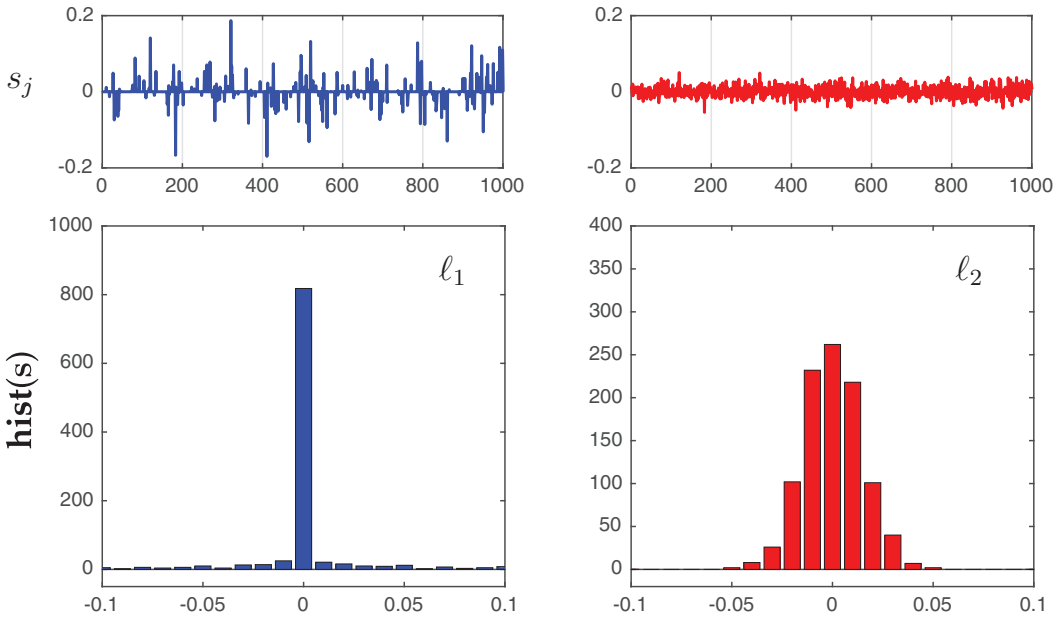

Figure 3.7 Comparison of ℓ_1-minimum (blue, left) and ℓ_2-minimum norm (red, right) solutions to an underdetermined linear system.

Recovering an Audio Signal from Sparse Measurements

To illustrate the use of compressed sensing to reconstruct a high-dimensional signal from a sparse set of random measurements, we consider a signal consisting of a two-tone audio signal:

$$x(t) = \cos(2\pi \times 97t) + \cos(2\pi \times 777t). \tag{3.10}$$

This signal is clearly sparse in the frequency domain, as it is defined by a sum of exactly two cosine waves. The highest frequency present is 777 Hz, so that the Nyquist sampling rate is 1554 Hz. However, leveraging the sparsity of the signal in the frequency domain, we can accurately reconstruct the signal with random samples that are spaced at an average sampling rate of 128 Hz, which is well below the Nyquist sampling rate. Fig. 3.8 shows the result of compressed sensing, as implemented in Code 3.2. In this example, the full signal is generated from $t = 0$ to $t = 1$ with a resolution of $n = 4,096$ and is then randomly sampled at $p = 128$ locations in time. The sparse vector of coefficients in the discrete cosine transform (DCT) basis is solved for using matching pursuit.

Code 3.2 Compressed sensing reconstruction of two-tone cosine signal.

```
%% Generate signal, DCT of signal
n = 4096;     % points in high resolution signal
```

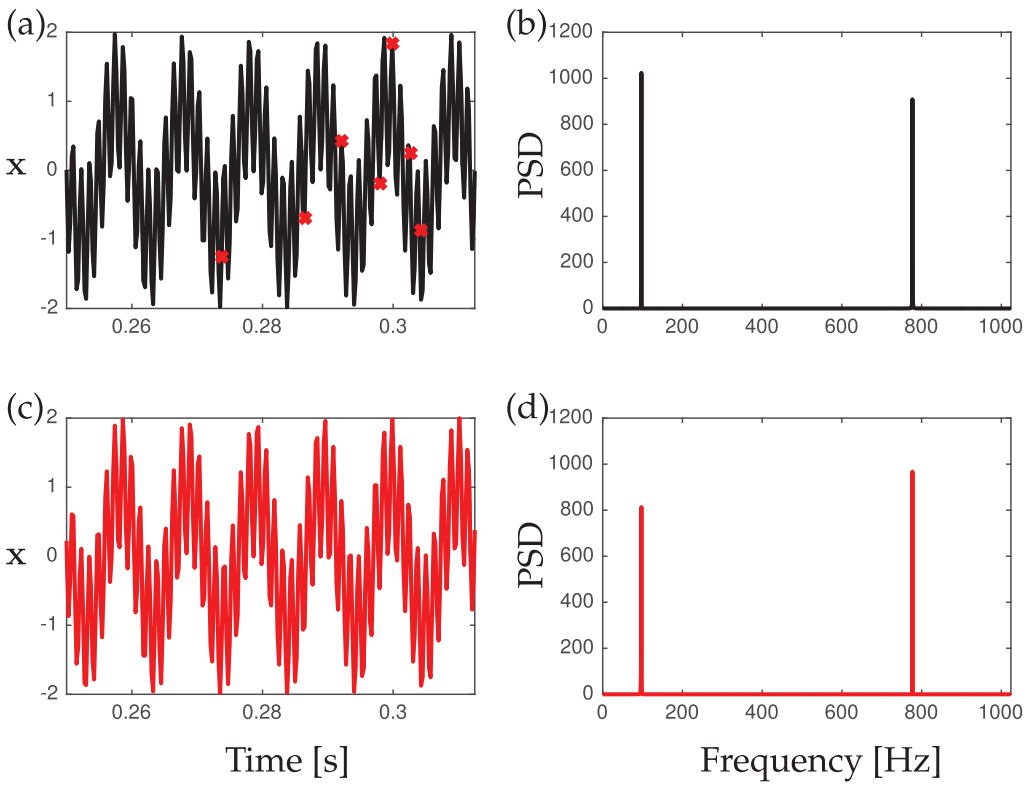

Figure 3.8 Compressed sensing reconstruction of a two-tone audio signal given by $x(t) = \cos(2\pi \times 97t) + \cos(2\pi \times 777t)$. The full signal and power spectral density are shown in panels (a) and (b), respectively. The signal is measured at random sparse locations in time, demarcated by red points in (a), and these measurements are used to build the compressed sensing estimate in (c) and (d). The time series shown in (a) and (c) are a zoom-in of the entire time range, which is from $t = 0$ to $t = 1$.

```
t = linspace(0, 1, n);
x = cos(2* 97 * pi * t) + cos(2* 777 * pi * t);
xt = fft(x); % Fourier transformed signal
PSD = xt.*conj(xt)/n;  % Power spectral density

%% Randomly sample signal
p = 128; % num. random samples, p=n/32
perm = round(rand(p, 1) * n);
y = x(perm); % compressed measurement

%% Solve compressed sensing problem
Psi = dct(eye(n, n));  % build Psi
Theta = Psi(perm, :);  % Measure rows of Psi

s = cosamp(Theta,y',10,1.e-10,10); % CS via matching pursuit
xrecon = idct(s);      % reconstruct full signal
```

It is important to note that the $p = 128$ measurements are randomly chosen from the $4,096$ resolution signal. Thus, we know the precise timing of the sparse measurements at a much higher resolution than our sampling rate. If we chose $p = 128$ measurements uniformly in time, the compressed sensing algorithm fails. Specifically, if we compute the PSD directly from these uniform measurements, the high-frequency signal will be aliased resulting in erroneous frequency peaks.

Finally, it is also possible to replace the matching pursuit algorithm

```
s = cosamp(Theta,y',10,1.e-10,10); % CS via matching pursuit
```

with an ℓ_1 minimization using the **CVX** package [218]:

```
%% L1-Minimization using CVX
cvx_begin;
    variable s(n);
    minimize( norm(s,1) );
    subject to
        Theta*s == y';
cvx_end;
```

In the compressed sensing matching pursuit (CoSaMP) code, the desired level of sparsity K must be specified, and this quantity may not be known ahead of time. The ℓ_1 minimization routine does not require knowledge of the desired sparsity level *a priori*, although convergence to the sparsest solution relies on having sufficiently many measurements p, which indirectly depends on K.

3.4 The Geometry of Compression

Compressed sensing can be summarized in a relatively simple statement: A given signal, if it is sufficiently sparse in a known basis, may be recovered (with high probability) using significantly fewer measurements than the signal length, if there are sufficiently many measurements and these measurements are sufficiently random. Each part of this statement can be made precise and mathematically rigorous in an overarching framework that describes the geometry of sparse vectors, and how these vectors are transformed through random measurements. Specifically, enough good measurements will result in a matrix

$$\mathbf{\Theta} = \mathbf{C}\mathbf{\Psi} \tag{3.11}$$

that preserves the distance and inner product structure of sparse vectors $\mathbf{s}$. In other words, we seek a measurement matrix $\mathbf{C}$ so that $\boldsymbol{\Theta}$ acts as a near isometry map on sparse vectors. Isometry literally means *same distance*, and is closed related to unitarity, which not only preserves distance, but also angles between vectors. When $\boldsymbol{\Theta}$ acts as a near isometry, it is possible to solve the following equation for the sparsest vector $\mathbf{s}$ using convex ℓ_1 minimization:

$$\mathbf{y} = \boldsymbol{\Theta}\mathbf{s}. \tag{3.12}$$

The remainder of this section describes the conditions on the measurement matrix $\mathbf{C}$ that are required for $\boldsymbol{\Theta}$ to act as a near isometry map with high probability. The geometric properties of various norms are shown in Fig. 3.9.

Determining how many measurements to take is relatively simple. If the signal is K-sparse in a basis $\boldsymbol{\Psi}$, meaning that all but K coefficients are zero, then the number of measurements scales as $p \sim \mathcal{O}(K \log(n/K)) = k_1 K \log(n/K)$, as in (3.7). The constant multiplier k_1, which defines *exactly* how many measurements are needed, depends on the quality of the measurements. Roughly speaking, measurements are good if they are *incoherent* with respect to the columns of the sparsifying basis, meaning that the rows of $\mathbf{C}$ have small inner product with the columns of $\boldsymbol{\Psi}$. If the measurements are coherent with columns of the sparsifying basis, then a measurement will provide little information unless that basis mode happens to be non-zero in $\mathbf{s}$. In contrast, incoherent measurements are excited by nearly any active mode, making it possible to infer the active modes. Delta functions are incoherent with respect to Fourier modes, as they excite a broadband fre-

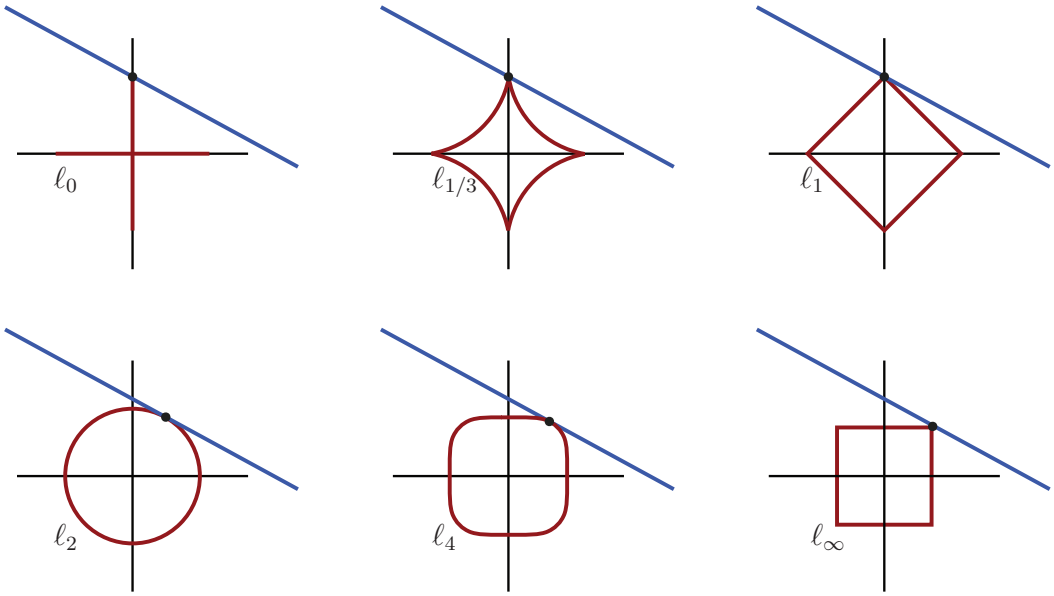

Figure 3.9 The minimum norm point on a line in different ℓ_p norms. The blue line represents the solution set of an under-determined system of equations, and the red curves represent the minimum-norm level sets that intersect this blue line for different norms. In the norms between ℓ_0 and ℓ_1, the minimum-norm solution also corresponds to the sparsest solution, with only one coordinate active. In the ℓ_2 and higher norms, the minimum-norm solution is not sparse, but has all coordinates active.

quency response. The more *incoherent* the measurements, the smaller the required number of measurements p.

The incoherence of measurements $\mathbf{C}$ and the basis $\mathbf{\Psi}$ is given by $\mu(\mathbf{C}, \mathbf{\Psi})$:

$$\mu(\mathbf{C}, \Psi) = \sqrt{n} \max_{j,k} |\langle \mathbf{c}_k, \boldsymbol{\psi}_j \rangle|, \tag{3.13}$$

where $\mathbf{c}_k$ is the kth row of the matrix $\mathbf{C}$ and $\boldsymbol{\psi}_j$ is the jth column of the matrix $\mathbf{\Psi}$. The coherence μ will range between 1 and $\sqrt{n}$.

The Restricted Isometry Property (RIP)

When measurements are incoherent, the matrix $\mathbf{C\Psi}$ satisfies a *restricted isometry property* (RIP) for sparse vectors $\mathbf{s}$,

$$(1 - \delta_K)\|\mathbf{s}\|_2^2 \leq \|\mathbf{C\Psi s}\|_2^2 \leq (1 + \delta_K)\|\mathbf{s}\|_2^2,$$

with restricted isometry constant δ_K [114]. The constant δ_K is defined as the smallest number that satisfies the above inequality for *all* K-sparse vectors $\mathbf{s}$. When δ_K is small, then $\mathbf{C\Psi}$ acts as a near isometry on K-sparse vectors $\mathbf{s}$. In practice, it is difficult to compute δ_K directly; moreover, the measurement matrix $\mathbf{C}$ may be chosen to be random, so that it is more desirable to derive statistical properties about the bounds on δ_K for a family of measurement matrices $\mathbf{C}$, rather than to compute δ_K for a specific $\mathbf{C}$. Generally, increasing the number of measurements will decrease the constant δ_K, improving the property of $\mathbf{C\Psi}$ to act isometrically on sparse vectors. When there are sufficiently many incoherent measurements, as described above, it is possible to accurately determine the K nonzero elements of the n-length vector $\mathbf{s}$. In this case, there are bounds on the constant δ_K that guarantee exact signal reconstruction for noiseless data. An in-depth discussion of incoherence and the RIP can be found in [39, 114].

Incoherence and Measurement Matrices

Another significant result of compressed sensing is that there are generic sampling matrices $\mathbf{C}$ that are sufficiently incoherent with respect to nearly all transform bases. Specifically, Bernouli and Gaussian random measurement matrices satisfy the RIP for a generic basis $\mathbf{\Psi}$ with high probability [113]. There are additional results generalizing the RIP and investigating incoherence of sparse matrices [205].

In many engineering applications, it is advantageous to represent the signal $\mathbf{x}$ in a generic basis, such as Fourier or wavelets. One key advantage is that single-point measurements are incoherent with respect to these bases, exciting a broadband frequency response. Sampling at random point locations is appealing in applications where individual measurements are expensive, such as in ocean monitoring. Examples of random measurement matrices, including single pixel, Gaussian, Bernoulli, and sparse random, are shown in Fig. 3.10.

A particularly useful transform basis for compressed sensing is obtained by the SVD[4], resulting in a tailored basis in which the data is optimally sparse [316, 80, 81, 31, 98]. A truncated SVD basis may result in a more efficient signal recovery from fewer measurements. Progress has been made developing a compressed SVD and PCA based on the

[4] The SVD provides an optimal low-rank matrix approximation, and it is used in principal components analysis (PCA) and proper orthogonal decomposition (POD).

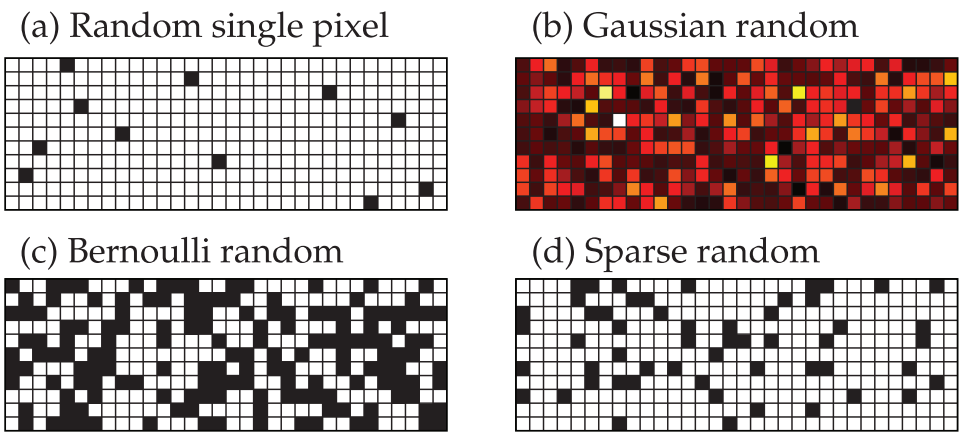

Figure 3.10 Examples of good random measurement matrices $\mathbf{C}$.

Johnson-Lindenstrauss (JL) lemma [267, 187, 436, 206]. The JL lemma is closely related to the RIP, indicating when it is possible to embed high-dimensional vectors in a low-dimensional space while preserving spectral properties.

Bad Measurements

So far we have described how to take *good* compressed measurements. Fig. 3.11 shows a particularly poor choice of measurements $\mathbf{C}$, corresponding to the last p columns of the sparsifying basis $\mathbf{\Psi}$. In this case, the product $\mathbf{\Theta} = \mathbf{C\Psi}$ is a $p \times p$ identity matrix padded with zeros on the left. In this case, any signal $\mathbf{s}$ that is not active in the last p columns of $\mathbf{\Psi}$ is in the null-space of $\mathbf{\Theta}$, and is completely invisible to the measurements $\mathbf{y}$. In this case, these measurements incur significant information loss for many sparse vectors.

3.5 Sparse Regression

The use of the ℓ_1 norm to promote sparsity significantly predates compressed sensing. In fact, many benefits of the ℓ_1 norm were well-known and oft-used in statistics decades earlier. In this section, we show that the ℓ_1 norm may be used to *regularize* statistical regression, both to penalize statistical outliers and also to promote *parsimonious* statistical models with as few factors as possible. The role of ℓ_2 versus ℓ_1 in regression is further detailed in Chapter 4.

Outlier Rejection and Robustness

Least squares regression is perhaps the most common statistical model used for data fitting. However, it is well known that the regression fit may be arbitrarily corrupted by a single large outlier in the data; outliers are weighted more heavily in least-squares regression because their distance from the fit-line is squared. This is shown schematically in Fig. 3.12.

In contrast, ℓ_1-minimum solutions give equal weight to all data points, making it potentially more robust to outliers and corrupt data. This procedure is also known as least absolute deviations (LAD) regression, among other names. A script demonstrating the

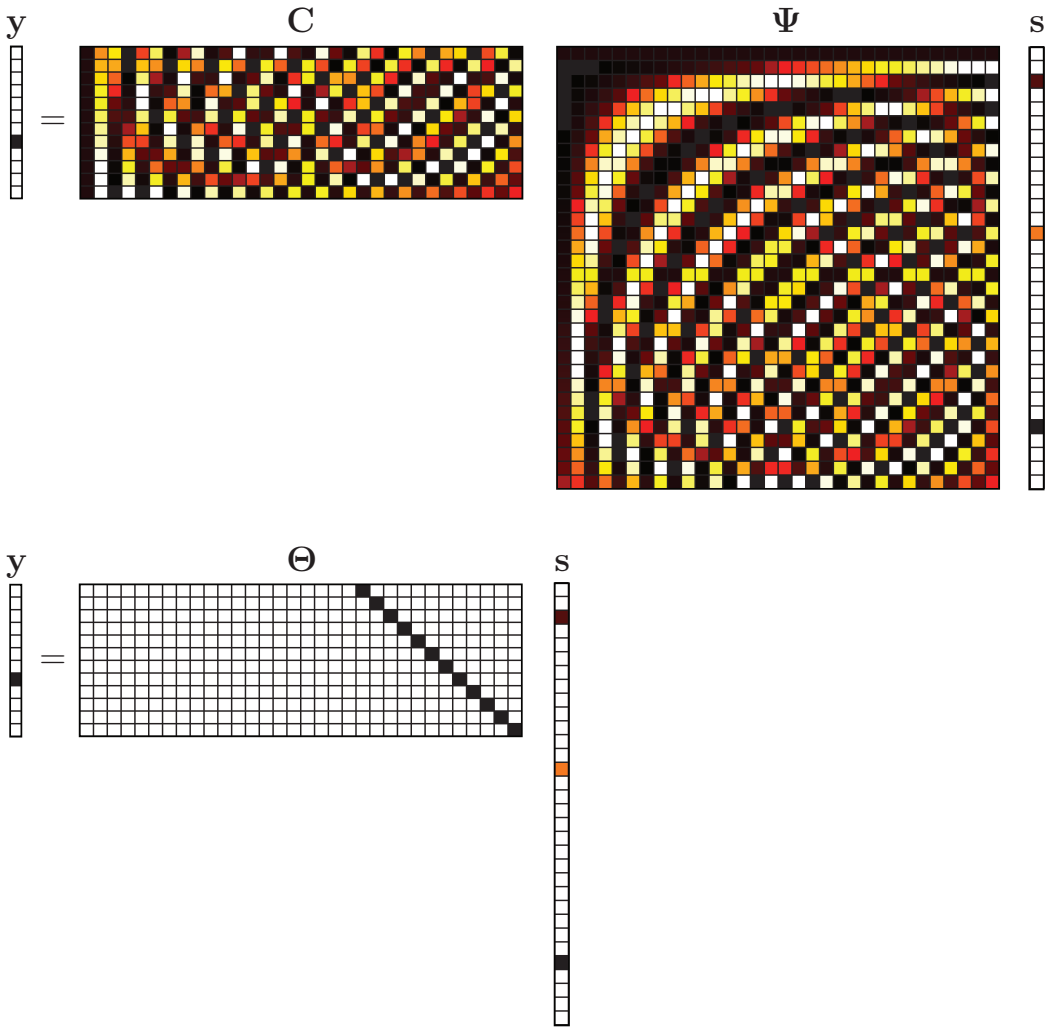

Figure 3.11 Examples of a bad measurement matrix $\mathbf{C}$.

use of least-squares (ℓ_2) and LAD (ℓ_1) regression for a dataset with an outlier is given in Code 3.3.

Code 3.3 Use of ℓ_1 norm for robust statistical regression.

```
x = sort(4*(rand(25,1)-.5)); % Random data from [-2,2]
b = .9*x + .1*randn(size(x)); % Line y=.9x with noise
atrue = x\b;           % Least-squares slope (no outliers)

b(end) = -5.5;                  % Introduce outlier
acorrupt = x\b;                 % New slope

cvx_begin;        % L1 optimization to reject outlier
    variable aL1;     % aL1 is slope to be optimized
    minimize( norm(aL1*x-b,1) );      % aL1 is robust
cvx_end;
```

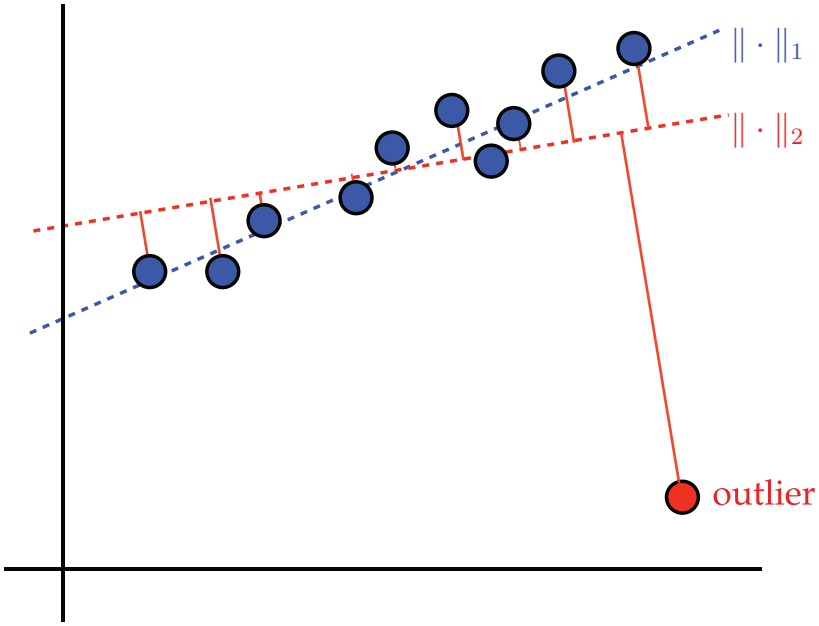

Figure 3.12 Least-squares regression is sensitive to outliers (red), while minimum ℓ_1-norm regression is robust to outliers (blue).

Feature Selection and LASSO Regression

Interpretability is important in statistical models, as these models are often communicated to a non-technical audience, including business leaders and policy makers. Generally, a regression model is more interpretable if it has fewer terms that bear on the outcome, motivating yet another perspective on sparsity.

The least absolute shrinkage and selection operator (LASSO) is an ℓ_1 penalized regression technique that balances model complexity with descriptive capability [518]. This principle of *parsimony* in a model is also a reflection of Occam's razor, stating that among all possible descriptions, the simplest correct model is probably the true one. Since its inception by Tibshirani in 1996 [518], the LASSO has become a cornerstone of statistical modeling, with many modern variants and related techniques [236, 558, 264]. The LASSO is closely related to the earlier nonnegative garrote of Breimen [76], and is also related to earlier work on soft-thresholding by Donoho and Johnstone [153, 154]. LASSO may be thought of as a sparsity-promoting regression that benefits from the stability of the ℓ_2 regularized ridge regression [249], also known as Tikhonov regularization. The elastic net is a frequently used regression technique that combines the ℓ_1 and ℓ_2 penalty terms from LASSO and ridge regression [573]. Sparse regression will be explored in more detail in Chapter 4.

Given a number of observations of the predictors and outcomes of a system, arranged as rows of a matrix $\mathbf{A}$ and a vector $\mathbf{b}$, respectively, regression seeks to find the relationship between the columns of $\mathbf{A}$ that is most consistent with the outcomes in $\mathbf{b}$. Mathematically, this may be written as:

$$\mathbf{Ax} = \mathbf{b}. \tag{3.14}$$

Least-squares regression will tend to result in a vector $\mathbf{x}$ that has nonzero coefficients for all entries, indicating that *all* columns of $\mathbf{A}$ must be used to predict $\mathbf{b}$. However, we often believe that the statistical model should be *simpler*, indicating that $\mathbf{x}$ may be sparse. The LASSO adds an ℓ_1 penalty term to *regularize* the least-squares regression problem; i.e., to prevent overfitting:

$$\mathbf{x} = \underset{\mathbf{x}'}{\operatorname{argmin}} \|\mathbf{A}\mathbf{x}' - \mathbf{b}\|_2 + \lambda\|\mathbf{x}\|_1. \tag{3.15}$$

Typically, the parameter λ is varied through a range of values and the fit is *validated* against a test set of holdout data. If there is not enough data to have a sufficiently large training and test set, it is common to repeatedly train and test the model on random selection of the data (often 80 % for training and 20 % for testing), resulting in a *cross-validated* performance. This cross-validation procedure enables the selection of a parsimonious model that has relatively few terms and avoids overfitting.

Many statistical systems are overdetermined, as there are more observations than candidate predictors. Thus, it is not possible to use standard compressed sensing, as measurement noise will guarantee that no exact sparse solution exists that minimizes $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$. However, the LASSO regression works well with overdetermined problems, making it a general regression method. Note that an early version of the geometric picture in Fig. 3.9 to explain the sparsity-promoting nature of the ℓ_1 norm was presented in Tibshirani's 1996 paper [518].

LASSO regression is frequently used to build statistical models for disease, such as cancer and heart failure, since there are many possible predictors, including demographics, lifestyle, biometrics and genetic information. Thus, LASSO represents a clever version of the *kitchen-sink* approach, whereby nearly all possible predictive information is thrown into the mix, and afterwards these are then sifted and sieved through for the truly relevant predictors.

As a simple example, we consider an artificial data set consisting of 100 observations of an outcome, arranged in a vector $\mathbf{b} \in \mathbb{R}^{100}$. Each outcome in $\mathbf{b}$ is given by a combination of exactly 2 out of 10 candidate predictors, whose observations are arranged in the rows of a matrix $\mathbf{A} \in \mathbb{R}^{100\times 10}$:

```
A = randn(100,10);      % Matrix of possible predictors
x = [0; 0; 1; 0; 0; 0; -1; 0; 0; 0]; % 2 nonzero predictors
b = A*x + 2*randn(100,1);     % Observations (with noise)
```

The vector $\mathbf{x}$ is sparse by construction, with only two nonzero entries, and we also add noise to the observations in $\mathbf{b}$. The least-squares regression is:

```
>>xL2 = pinv(A)*b

  xL2 = -0.0232
        -0.3395
         0.9591
        -0.1777
         0.2912
        -0.0525
        -1.2720
        -0.0411
         0.0413
        -0.0500
```

Note that all coefficients are nonzero.

Implementing the LASSO, with 10-fold cross-validation, is a single straightforward command in MATLAB:

```
[XL1 FitInfo] = lasso(A,b,'CV',10);
```

The **lasso** command sweeps through a range of values for λ, and the resulting **x** are each stored as columns of the matrix in **XL1**. To select the most parsimonious model that describes the data while avoiding overfitting, we may plot the cross-validated error as a function of λ, as in Fig. 3.13:

```
lassoPlot(XL1,FitInfo,'PlotType','CV')
```

The green point is at the value of λ that minimizes the cross-validated mean-square error, and the blue point is at the minimum cross-validated error plus one standard deviation. The resulting model is found via **FitInfo.Index1SE**:

```
>> xL1 = XL1(:,FitInfo.Index1SE)

  xL1 =     0
            0
       0.7037
            0
            0
            0
      -0.4929
            0
            0
            0
```

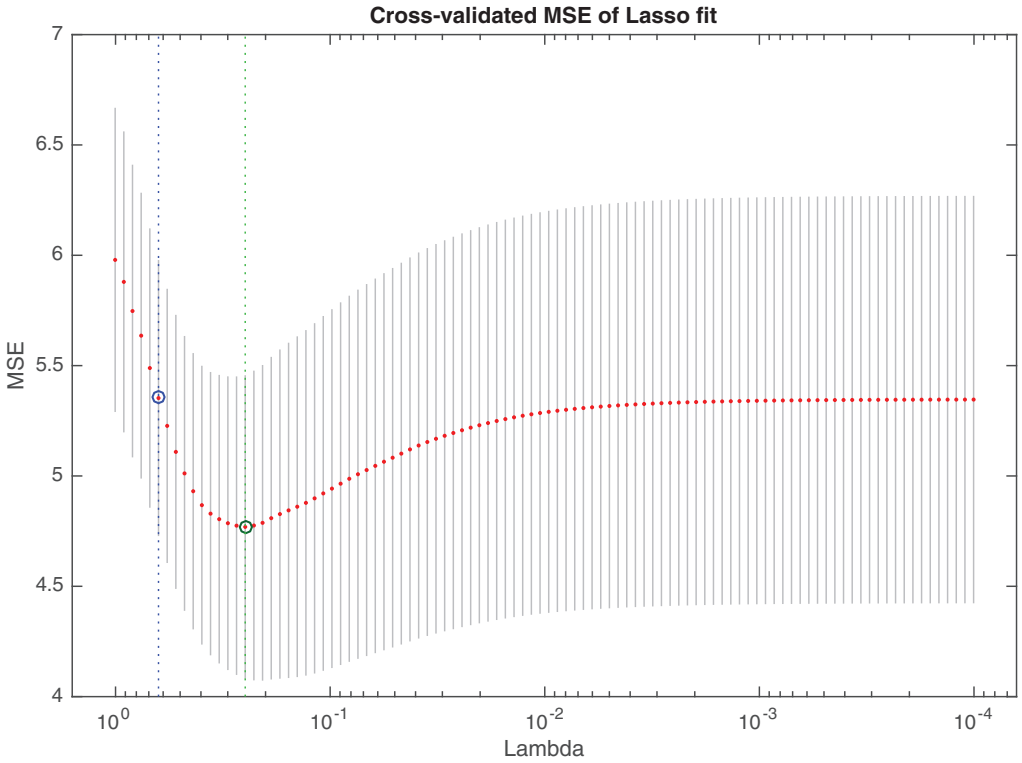

Figure 3.13 Output of **lassoPlot** command to visualize cross-validated mean-squared error (MSE) as a function of λ.

Note that the resulting model is sparse and the correct terms are active. However, the regression values for these terms are not accurate, and so it may be necessary to *de-bias* the LASSO by applying a final least-squares regression to the nonzero coefficients identified:

```
>>xL1DeBiased = pinv(A(:,abs(xL1)>0))*b
  xL1DeBiased =  1.0980
                -1.0671
```

3.6 Sparse Representation

Implicit in our discussion on sparsity is the fact that when high-dimensional signals exhibit low-dimensional structure, they admit a *sparse representation* in an appropriate basis or dictionary. In addition to a signal being sparse in an SVD or Fourier basis, it may also be sparse in an overcomplete dictionary whose columns consist of the training data itself. In essence, in addition to a test signal being sparse in generic feature library $\mathbf{U}$ from the SVD, $\mathbf{X} = \mathbf{U\Sigma V}^*$, it may also have a sparse representation in the dictionary $\mathbf{X}$.

Wright et al. [560] demonstrated the power of sparse representation in a dictionary of test signals for robust classification of human faces, despite significant noise and occlusions. The so-called sparse representation for classification (SRC) has been widely used in image processing, and more recently to classify dynamical regimes in nonlinear differential equations [98, 433, 191, 308].

The basic schematic of SRC is shown in Fig. 3.14, where a library of images of faces is used to build an overcomplete library $\mathbf{\Theta}$. In this example, 30 images are used for each of 20 different people in the Yale B database, resulting in 600 columns in $\mathbf{\Theta}$. To use compressed sensing, i.e. ℓ_1-minimization, we need $\mathbf{\Theta}$ to be underdetermined, and so we downsample each image from 192×168 to 12×10, so that the flattened images are 120-component vectors. The algorithm used to downsample the images has an impact on the classification accuracy. A new test image $\mathbf{y}$ corresponding to class c, appropriately downsampled to match the columns of $\mathbf{\Theta}$, is then sparsely represented as a sum of the columns of $\mathbf{\Theta}$ using the compressed sensing algorithm. The resulting vector of coefficients $\mathbf{s}$ should be sparse, and ideally will have large coefficients primarily in the regions of the library corresponding to the correct person in class c. The final classification stage in the algorithm is achieved by computing the ℓ_2 reconstruction error using the coefficients in the $\mathbf{s}$ vector corresponding to each of the categories separately. The category that minimizes the ℓ_2 reconstruction error is chosen for the test image.

Code 3.4 Load Yale faces data and build training and test sets.

```
load ../../CH01_SVD/DATA/allFaces.mat
X = faces;
%% Build Training and Test sets
nTrain = 30;  nTest = 20;  nPeople = 20;
Train = zeros(size(X,1),nTrain*nPeople);
Test = zeros(size(X,1),nTest*nPeople);
for k=1:nPeople
    baseind = 0;
    if(k>1) baseind = sum(nfaces(1:k-1));
    end
    inds = baseind + (1:nfaces(k));
    Train(:,(k-1)*nTrain+1:k*nTrain)=X(:,inds(1:nTrain));
```

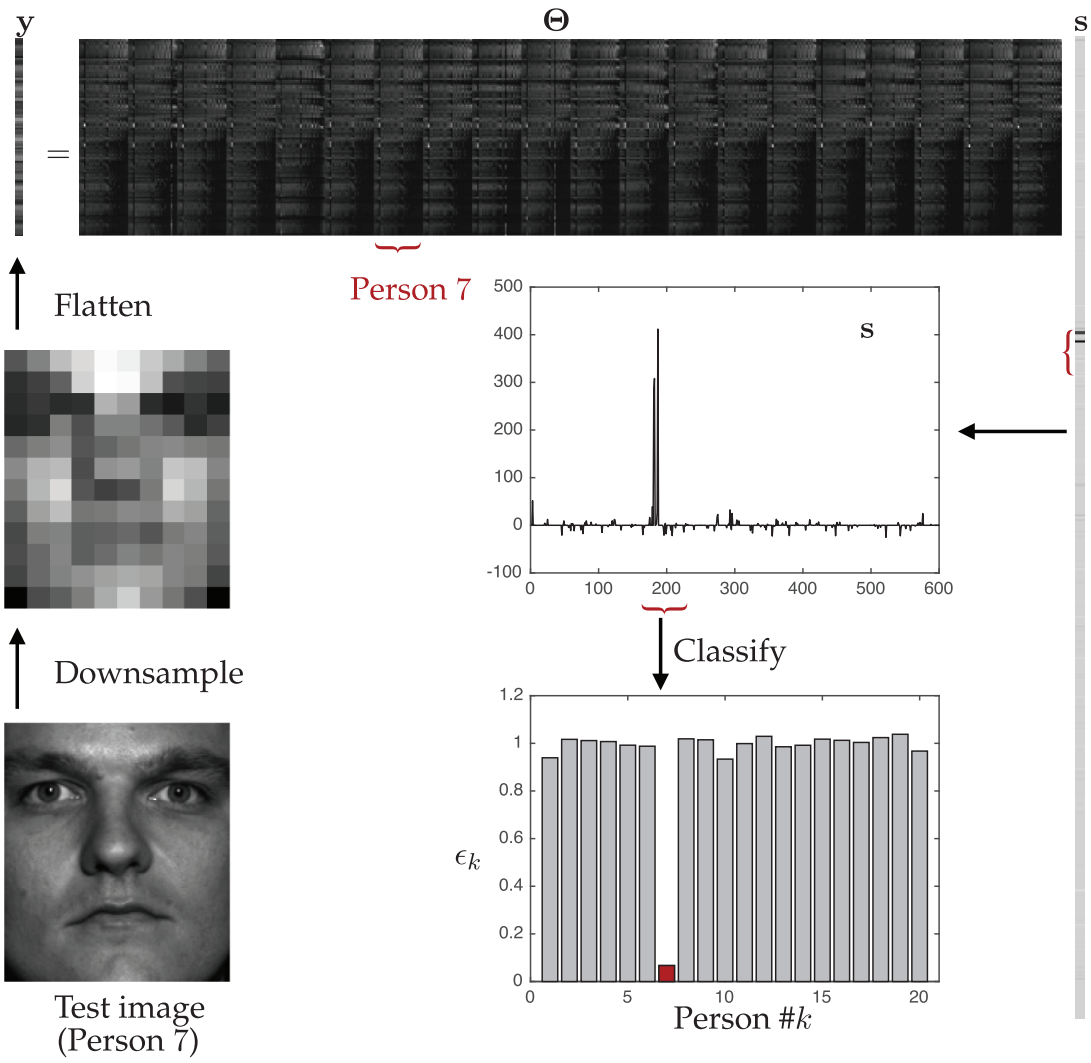

Figure 3.14 Schematic overview of sparse representation for classification.

```
    Test(:,(k-1)*nTest+1:k*nTest)=X(:,inds(nTrain+1:nTrain+nTest
        ));
end
```

Code 3.5 Downsample training images to build $\boldsymbol{\Theta}$ library.

```
M = size(Train,2);
Theta = zeros(120,M);
for k=1:M
    temp = reshape(Train(:,k),n,m);
    tempSmall = imresize(temp,[12 10],'lanczos3');
    Theta(:,k) = reshape(tempSmall,120,1);
end
for k=1:M  % Normalize columns of Theta
    Theta(:,k) = Theta(:,k)/norm(Theta(:,k));
end
```

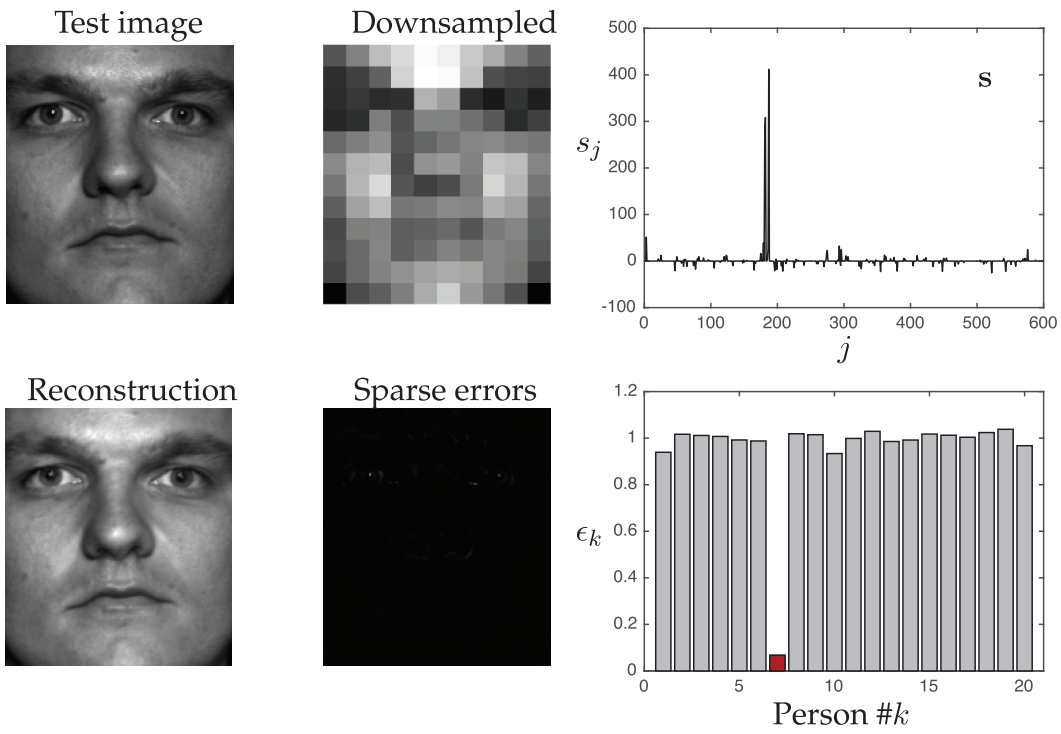

Figure 3.15 Sparse representation for classification demonstrated using a library of faces. A clean test image is correctly identified as the 7*th* person in the library.

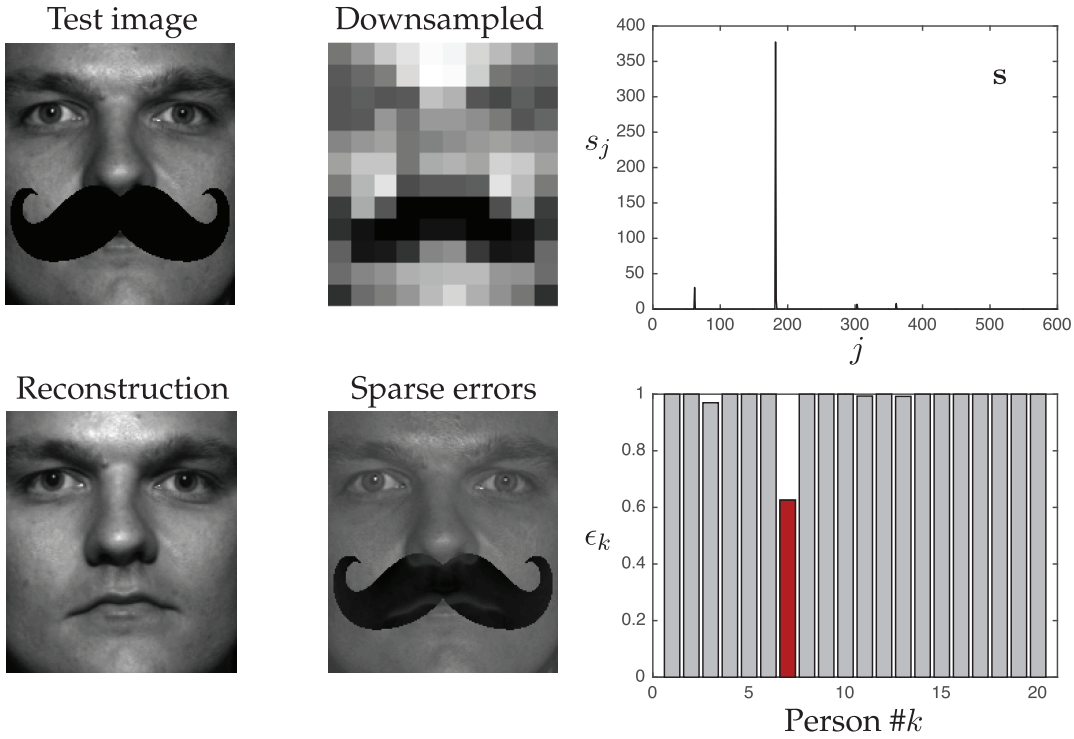

Figure 3.16 Sparse representation for classification demonstrated on example face from person #7 occluded by a fake mustache.

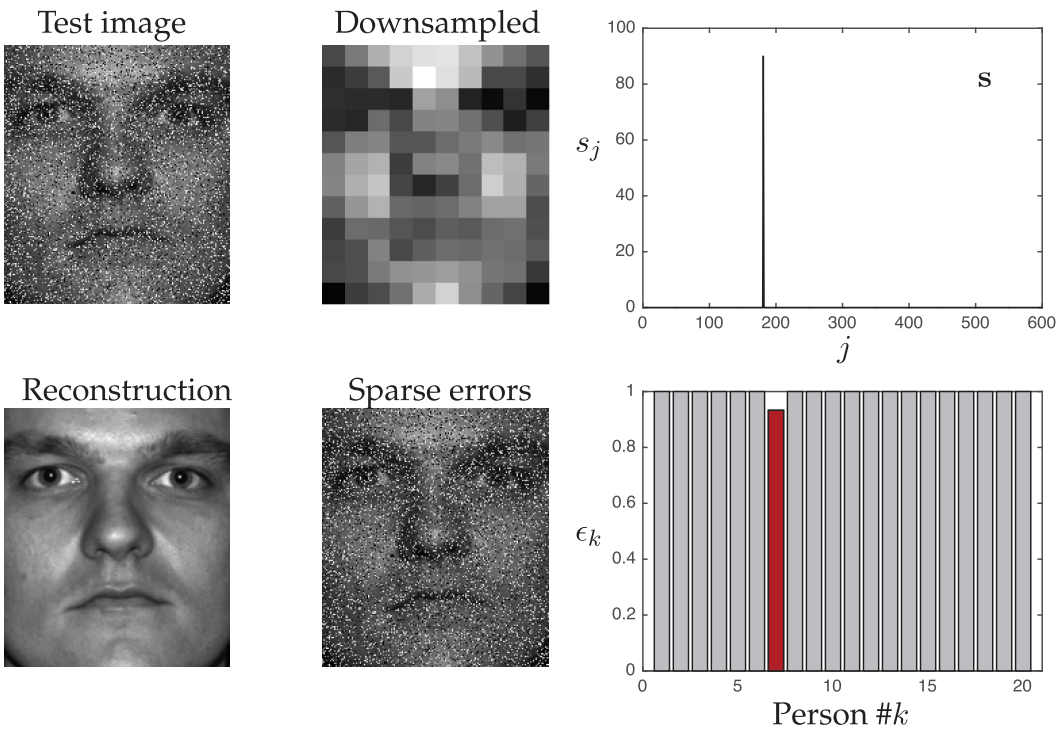

Figure 3.17 Sparse representation for classification demonstrated on example image with 30% occluded pixels (randomly chosen and uniformly distributed).

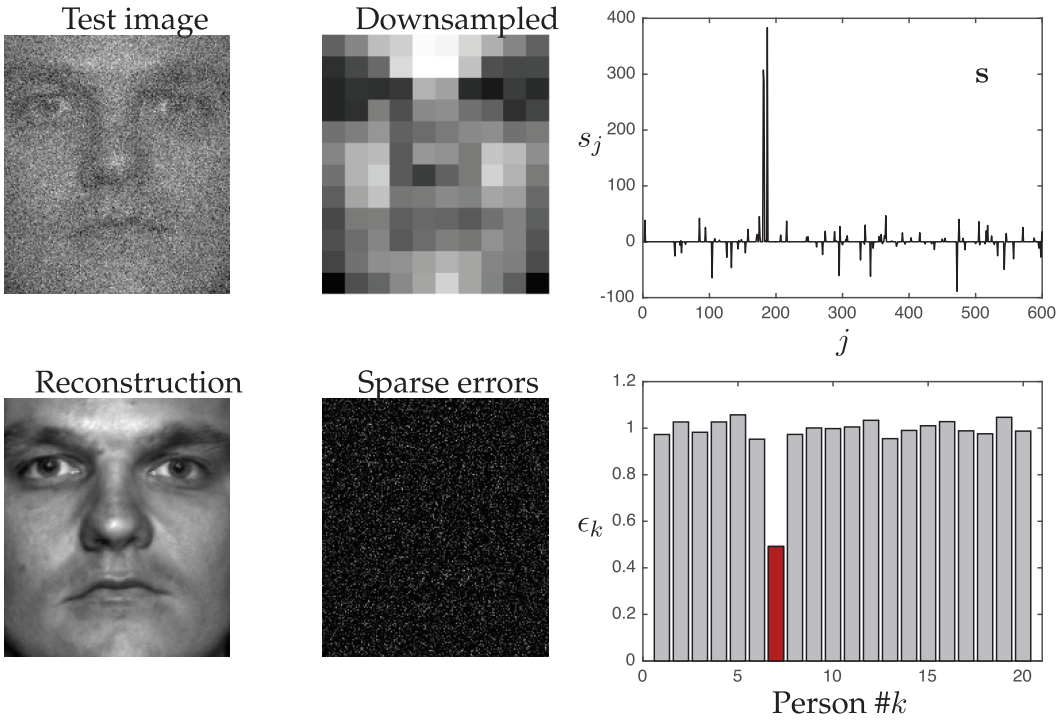

Figure 3.18 Sparse representation for classification demonstrated on example with white noise added to image.

Code 3.6 Build test images and downsample to obtain $\mathbf{y}$.

```
x1 = Test(:,126);  % clean image
mustache = double(rgb2gray(imread('mustache.jpg'))/255);
x2 = Test(:,126).*reshape(mustache,n*m,1); % mustache
randvec = randperm(n*m);
first30 = randvec(1:floor(.3*length(randvec)));
vals30 = uint8(255*rand(size(first30)));
x3 = x1;
x3(first30) = vals30; % 30% occluded
x4 = x1 + 50*randn(size(x1));  % random noise

%% DOWNSAMPLE TEST IMAGES
X = [x1 x2 x3 x4];
Y = zeros(120,4);
for k=1:4
    temp = reshape(X(:,k),n,m);
    tempSmall = imresize(temp,[12 10],'lanczos3');
    Y(:,k) = reshape(tempSmall,120,1);
end

%% L1 SEARCH, TESTCLEAN
```

Code 3.7 Search for sparse representation of test image. The same code is used for each of the test images $\mathbf{y}_1$ through $\mathbf{y}_4$.

```
y1 = Y(:,1);
eps = .01;
cvx_begin;
    variable s1(M);   % sparse vector of coefficients
    minimize( norm(s1,1) );
    subject to
        norm(Theta*s1 - y1,2) < eps;
cvx_end;

plot(s1)
imagesc(reshape(Train*(s1./normTheta'),n,m))
imagesc(reshape(x1-(Train*(s1./normTheta')),n,m))

binErr = zeros(nPeople,1);
for k=1:nPeople
    L = (k-1)*nTrain+1:k*nTrain;
    binErr(k)=norm(x1-(Train(:,L)*(s1(L)./normTheta(L)')))/norm(
        x1)
end
bar(binErr)
```

3.7 Robust Principal Component Analysis (RPCA)

As mentioned earlier in Section 3.5, least-squares regression models are highly susceptible to outliers and corrupted data. Principal component analysis (PCA) suffers from the same weakness, making it *fragile* with respect to outliers. To ameliorate this sensitivity, Candès et al. [110] have developed a robust principal component analysis (RPCA) that seeks to decompose a data matrix $\mathbf{X}$ into a structured low-rank matrix $\mathbf{L}$ and a sparse matrix $\mathbf{S}$ containing outliers and corrupt data:

$$\mathbf{X} = \mathbf{L} + \mathbf{S}. \tag{3.16}$$

The principal components of $\mathbf{L}$ are *robust* to the outliers and corrupt data in $\mathbf{S}$. This decomposition has profound implications for many modern problems of interest, including video surveillance (where the background objects appear in $\mathbf{L}$ and foreground objects appear in $\mathbf{S}$), face recognition (eigenfaces are in $\mathbf{L}$ and shadows, occlusions, etc. are in $\mathbf{S}$), natural language processing and latent semantic indexing, and ranking problems[5].

Mathematically, the goal is to find $\mathbf{L}$ and $\mathbf{S}$ that satisfy the following:

$$\min_{\mathbf{L},\mathbf{S}} \text{rank}(\mathbf{L}) + \|\mathbf{S}\|_0 \text{ subject to } \mathbf{L} + \mathbf{S} = \mathbf{X}. \tag{3.17}$$

However, neither the rank($\mathbf{L}$) nor the $\|\mathbf{S}\|_0$ terms are convex, and this is not a tractable optimization problem. Similar to the compressed sensing problem, it is possible to solve for the optimal $\mathbf{L}$ and $\mathbf{S}$ with *high probability* using a convex relaxation of (3.17):

$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 \text{ subject to } \mathbf{L} + \mathbf{S} = \mathbf{X}. \tag{3.18}$$

Here, $\|\cdot\|_*$ denotes the nuclear norm, given by the sum of singular values, which is a proxy for rank. Remarkably, the solution to (3.18) converges to the solution of (3.17) with high probability if $\lambda = 1/\sqrt{\max(n, m)}$, where n and m are the dimensions of $\mathbf{X}$, given that $\mathbf{L}$ and $\mathbf{S}$ satisfy the following conditions:

1. $\mathbf{L}$ is not sparse
2. $\mathbf{S}$ is not low-rank; we assume that the entries are randomly distributed so that they do not have low-dimensional column space.

The convex problem in (3.17) is known as *principal component pursuit* (PCP), and may be solved using the augmented Lagrange multiplier (ALM) algorithm. Specifically, an augmented Lagrangian may be constructed:

$$\mathcal{L}(\mathbf{L}, \mathbf{S}, \mathbf{Y}) = \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \langle \mathbf{Y}, \mathbf{X} - \mathbf{L} - \mathbf{S}\rangle + \frac{\mu}{2}\|\mathbf{X} - \mathbf{L} - \mathbf{S}\|_F^2. \tag{3.19}$$

A general solution would solve for the $\mathbf{L}_k$ and $\mathbf{S}_k$ that minimize $\mathcal{L}$, update the Lagrange multipliers $\mathbf{Y}_{k+1} = \mathbf{Y}_k + \mu(\mathbf{X} - \mathbf{L}_k - \mathbf{S}_k)$, and iterate until the solution converges. However, for this specific system, the alternating directions method (ADM) [337, 566] provides a simple procedure to find $\mathbf{L}$ and $\mathbf{S}$.

First, a shrinkage operator $\mathcal{S}_\tau(x) = \text{sign}(x)\max(|x| - \tau, 0)$ is constructed (MATLAB function **shrink** below):

```
function out = shrink(X,tau)
    out = sign(X).*max(abs(X)-tau,0);
end
```

Next, the singular value threshold operator $\text{SVT}_\tau(\mathbf{X}) = \mathbf{U}\mathcal{S}_\tau(\mathbf{\Sigma})\mathbf{V}^*$ is constructed (MATLAB function **SVT** below):

```
function out = SVT(X,tau)
    [U,S,V] = svd(X,'econ');
    out = U*shrink(S,tau)*V';
end
```

[5] The ranking problem may be thought of in terms of the Netflix prize for matrix completion. In the Netflix prize, a large matrix of preferences is constructed, with rows corresponding to users and columns corresponding to movies. This matrix is sparse, as most users only rate a handful of movies. The Netflix prize seeks to accurately fill in the missing entries of the matrix, revealing the likely user rating for movies the user has not seen.

Finally, it is possible to use $\mathcal{S}_\tau$ and SVT operators iteratively to solve for $\mathbf{L}$ and $\mathbf{S}$:

Code 3.8 RPCA using alternating directions method (ADM).

```
function [L,S] = RPCA(X)
[n1,n2] = size(X);
mu = n1*n2/(4*sum(abs(X(:))));
lambda = 1/sqrt(max(n1,n2));
thresh = 1e-7*norm(X,'fro');

L = zeros(size(X));
S = zeros(size(X));
Y = zeros(size(X));
count = 0;
while((norm(X-L-S,'fro')>thresh)&&(count<1000))
    L = SVT(X-S+(1/mu)*Y,1/mu);
    S = shrink(X-L+(1/mu)*Y,lambda/mu);
    Y = Y + mu*(X-L-S);
    count = count + 1
end
```

This is demonstrated on the eigenface example with the following code:

```
load allFaces.mat
X = faces(:,1:nfaces(1));
[L,S] = RPCA(X);
```

In this example, the original columns of $\mathbf{X}$, along with the low-rank and sparse components, are shown in Fig. 3.19. Notice that in this example, RPCA effectively fills in occluded regions of the image, corresponding to shadows. In the low-rank component $\mathbf{L}$, shadows are removed and filled in with the most consistent low-rank features from the eigenfaces. This technique can also be used to remove other occlusions such as fake mustaches, sunglasses, or noise.

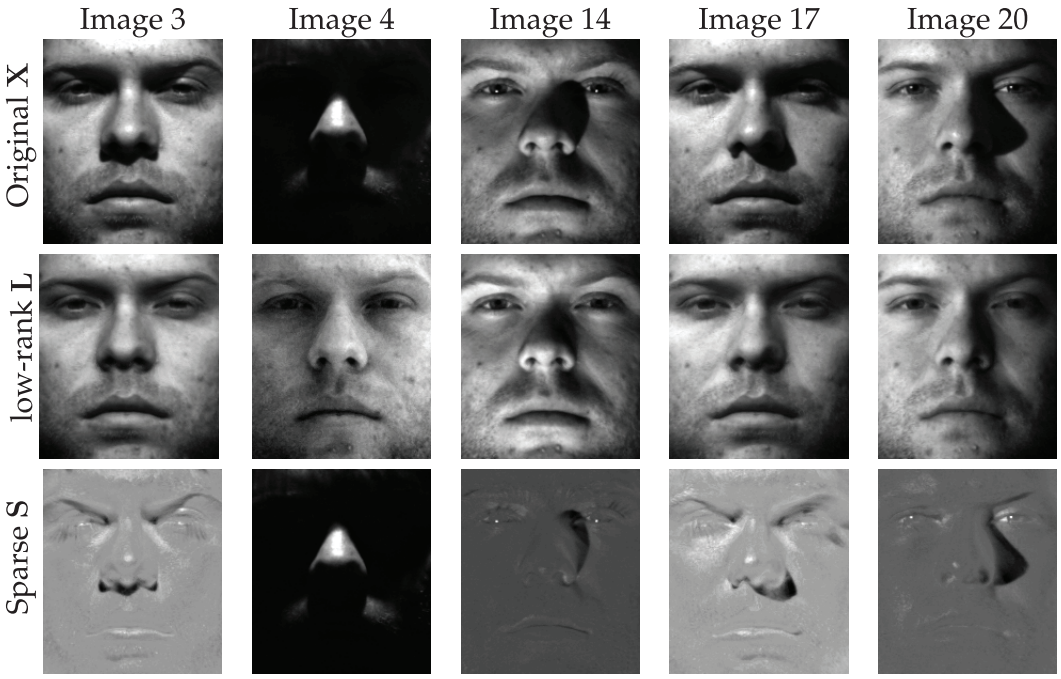

Figure 3.19 Output of RPCA for images in the Yale B database.

3.8 Sparse Sensor Placement

Until now, we have investigated signal reconstruction in a generic basis, such as Fourier or wavelets, with random measurements. This provides considerable flexibility, as no prior structure is assumed, except that the signal is sparse in a known basis. For example, compressed sensing works equally well for reconstructing an image of a mountain, a face, or a cup of coffee. However, if we know that we will be reconstructing a human face, we can dramatically reduce the number of sensors required for reconstruction or classification by optimizing sensors for a particular feature library $\boldsymbol{\Psi}_r = \tilde{\mathbf{U}}$ built from the SVD.

Thus, it is possible to design *tailored* sensors for a particular library, in contrast to the previous approach of random sensors in a generic library. Near-optimal sensor locations may be obtained using fast greedy procedures that scale well with large signal dimension, such as the matrix QR factorization. The following discussion will closely follow Manohar et al. [366] and B. Brunton et al. [89], and the reader is encouraged to find more details there. Similar approaches will be used for efficient sampling of reduced-order models in Chapter 12, where they are termed *hyper-reduction*. There are also extensions of the following for sensor and actuator placement in control [365], based on the balancing transformations discussed in Chapter 9.

Optimizing sensor locations is important for nearly all downstream tasks, including classification, prediction, estimation, modeling, and control. However, identifying optimal locations involves a brute force search through the combinatorial choices of p sensors out of n possible locations in space. Recent greedy and sparse methods are making this search tractable and scalable to large problems. Reducing the number of sensors through principled selection may be critically enabling when sensors are costly, and may also enable faster state estimation for low latency, high bandwidth control.

Sparse Sensor Placement for Reconstruction

The goal of optimized sensor placement in a tailored library $\boldsymbol{\Psi}_r \in \mathbb{R}^{n\times r}$ is to design a sparse measurement matrix $\mathbf{C} \in \mathbb{R}^{p\times n}$, so that inversion of the linear system of equations

$$\mathbf{y} = \mathbf{C}\boldsymbol{\Psi}_r\mathbf{a} = \boldsymbol{\theta}\mathbf{a} \tag{3.20}$$

is as well-conditioned as possible. In other words, we will design $\mathbf{C}$ to minimize the condition number of $\mathbf{C}\boldsymbol{\Psi}_r = \boldsymbol{\theta}$, so that it may be inverted to identify the low-rank coefficients $\mathbf{a}$ given noisy measurements $\mathbf{y}$. The condition number of a matrix $\boldsymbol{\theta}$ is the ratio of its maximum and minimum singular values, indicating how sensitive matrix multiplication or inversion is to errors in the input. Larger condition numbers indicate worse performance inverting a noisy signal. The condition number is a measure of the worst-case error when the signal $\mathbf{a}$ is in the singular vector direction associated with the minimum singular value of $\boldsymbol{\theta}$, and noise is added which is aligned with the maximum singular vector:

$$\boldsymbol{\theta}(\mathbf{a} + \epsilon_{\mathbf{a}}) = \sigma_{\min}\mathbf{a} + \sigma_{\max}\epsilon_{\mathbf{a}}. \tag{3.21}$$

Thus, the signal-to-noise ratio decreases by the condition number after mapping through $\boldsymbol{\theta}$. We therefore seek to minimize the condition number through a principled choice of $\mathbf{C}$. This is shown schematically in Fig. 3.20 for $p = r$.

When the number of sensors is equal to the rank of the library, i.e. $p = r$, then $\boldsymbol{\theta}$ is a square matrix, and we are choosing $\mathbf{C}$ to make this matrix as well-conditioned for inversion

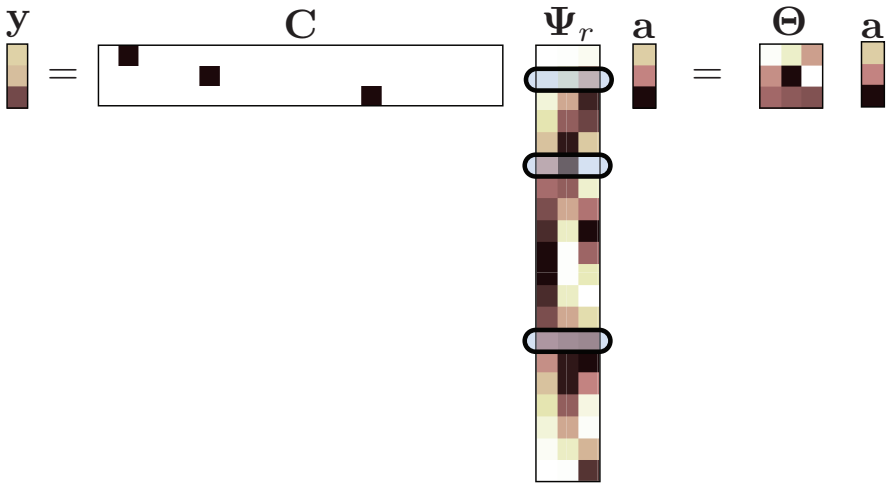

Figure 3.20 Least squares with r sparse sensors provides a unique solution to $\mathbf{a}$, hence $\mathbf{x}$. *Reproduced with permission from Manohar et al. [366].*

as possible. When $p > r$, we seek to improve the condition of $\mathbf{M} = \boldsymbol{\theta}^T\boldsymbol{\theta}$, which is involved in the pseudo-inverse. It is possible to develop optimization criteria that optimize the minimum singular value, the trace, or the determinant of $\boldsymbol{\theta}$ (resp. $\mathbf{M}$). However, each of these optimization problems is np-hard, requiring a combinatorial search over the possible sensor configurations. Iterative methods exist to solve this problem, such as convex optimization and semidefinite programming [74, 269], although these methods may be expensive, requiring iterative $n \times n$ matrix factorizations. Instead, greedy algorithms are generally used to approximately optimize the sensor placement. These *gappy POD* [179] methods originally relied on random sub-sampling. However, significant performance advances where demonstrated by using principled sampling strategies for reduced order models (ROMs) [53] in fluid dynamics [555] and ocean modeling [565]. More recently, variants of the so-called *empirical interpolation method* (EIM, DEIM and Q-DEIM) [41, 127, 159] have provided near optimal sampling for interpolative reconstruction of nonlinear terms in ROMs.

Random sensors. In general, randomly placed sensors may be used to estimate mode coefficients $\mathbf{a}$. However, when $p = r$ and the number of sensors is equal to the number of modes, the condition number is typically very large. In fact, the matrix $\boldsymbol{\Theta}$ is often numerically singular and the condition number is near 10^{16}. Oversampling, as in Sec. 1.8, rapidly improves the condition number, and even $p = r + 10$ usually has much better reconstruction performance.

QR Pivoting for sparse sensors. The greedy matrix QR factorization with column pivoting of $\boldsymbol{\Psi}_r^T$, explored by Drmac and Gugercin [159] for reduced-order modeling, provides a particularly simple and effective sensor optimization. The QR pivoting method is fast, simple to implement, and provides nearly optimal sensors tailored to a specific SVD/POD basis. QR factorization is optimized for most scientific computing libraries, including Matlab, LAPACK, and NumPy. In addition QR can be sped-up by ending the procedure after the first p pivots are obtained.

The reduced matrix QR factorization with column pivoting decomposes a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ into a unitary matrix $\mathbf{Q}$, an upper-triangular matrix $\mathbf{R}$ and a column permutation matrix $\mathbf{C}^T$ such that $\mathbf{AC}^T = \mathbf{QR}$. The pivoting procedure provides an approximate greedy solution method to minimize the matrix volume, which is the absolute value of the determinant. QR column pivoting increments the volume of the submatrix constructed from the pivoted columns by selecting a new pivot column with maximal 2-norm, then subtracting from every other column its orthogonal projection onto the pivot column.

Thus QR factorization with column pivoting yields r point sensors (pivots) that best sample the r basis modes $\mathbf{\Psi}_r$

$$\mathbf{\Psi}_r^T \mathbf{C}^T = \mathbf{QR}. \tag{3.22}$$

Based on the same principle of pivoted QR, which controls the condition number by minimizing the matrix volume, the oversampled case is handled by the pivoted QR factorization of $\mathbf{\Psi}_r \mathbf{\Psi}_r^T$,

$$(\mathbf{\Psi}_r \mathbf{\Psi}_r^T)\mathbf{C}^T = \mathbf{QR}. \tag{3.23}$$

The code for handling both cases is give by

```
if (p==r)     % QR sensor selection, p=r
    [Q,R,pivot] = qr(Psi_r','vector');
elseif (p>r) % Oversampled QR sensors, p>r
    [Q,R,pivot] = qr(Psi_r*Psi_r','vector');
end
C = zeros(p,n);
for j=1:p
    C(j,pivot(j))=1;
end
```

Example: Reconstructing a Face with Sparse Sensors

To demonstrate the concept of signal reconstruction in a tailored basis, we will design optimized sparse sensors in the library of eigenfaces from Section 1.6. Fig. 3.21 shows the QR sensor placement and reconstruction, along with the reconstruction using random sensors. We use $p = 100$ sensors in a $r = 100$ mode library. This code assumes that

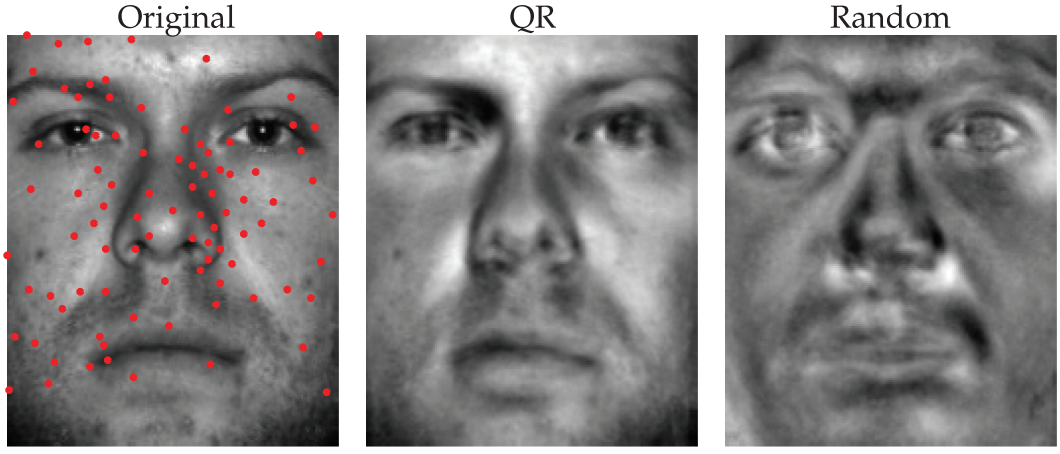

Figure 3.21 (left) Original image and $p = 100$ QR sensors locations in a $r = 100$ mode library. (middle) Reconstruction with QR sensors. (right) Reconstruction with random sensors.

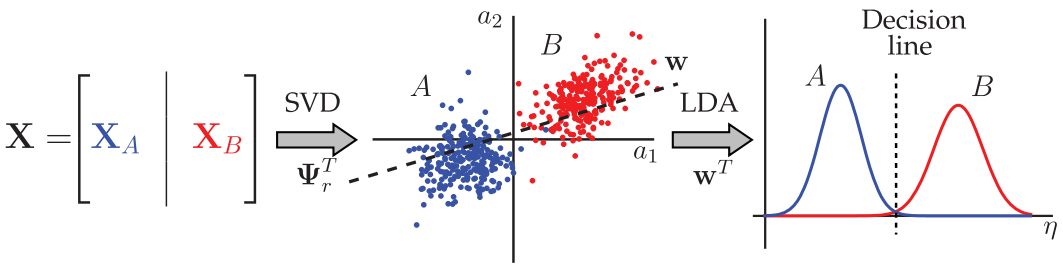

Figure 3.22 Schematic illustrating SVD for feature extraction, followed by LDA for the automatic classification of data into two categories A and B. *Reproduced with permission from Bai et al. [29].*

the faces have been loaded and the singular vectors are in a matrix $\mathbf{U}$. Optimized QR sensors result in a more accurate reconstruction, with about three times less reconstruction error. In addition, the condition number is orders of magnitude smaller than with random sensors. Both QR and random sensors may be improved by oversampling. The following code computes the QR sensors and the approximate reconstruction from these sensors.

```
r = 100; p = 100;  % # of modes r, # of sensors p
Psi = U(:,1:r);
[Q,R,pivot] = qr(Psi','vector');
C = zeros(p,n*m);
for j=1:p
    C(j,pivot(j))=1;
end
%
Theta = C*Psi;
y = faces(pivot(1:p),1); % Measure at pivot locations
a = Theta\y;             % Estimate coefficients
faceRecon = U(:,1:r)*a;  % Reconstruct face
```

Sparse Classification

For image classification, even fewer sensors may be required than for reconstruction. For example, sparse sensors may be selected that contain the most discriminating information to characterize two categories of data [89]. Given a library of r SVD modes $\mathbf{\Psi}_r$, it is often possible to identify a vector $\mathbf{w} \in \mathbb{R}^r$ in this subspace that maximally distinguishes between two categories of data, as described in Section 5.6 and shown in Fig. 3.22. Sparse sensors $\mathbf{s}$ that map into this discriminating direction, projecting out all other information, are found by:

$$\mathbf{s} = \underset{\mathbf{s}'}{\operatorname{argmin}} \|\mathbf{s}'\|_1 \quad \text{subject to} \quad \mathbf{\Psi}_r^T \mathbf{s}' = \mathbf{w}. \tag{3.24}$$

This sparse sensor placement optimization for classification (SSPOC) is shown in Fig. 3.23 for an example classifying dogs versus cats. The library $\mathbf{\Psi}_r$ contains the first r *eigenpets* and the vector $\mathbf{w}$ identifies the key differences between dogs and cats. Note that this vector does not care about the degrees of freedom that characterize the various features within the dog or cat clusters, but rather only the differences between the two categories. Optimized sensors are aligned with regions of interest, such as the eyes, nose, mouth, and ears.

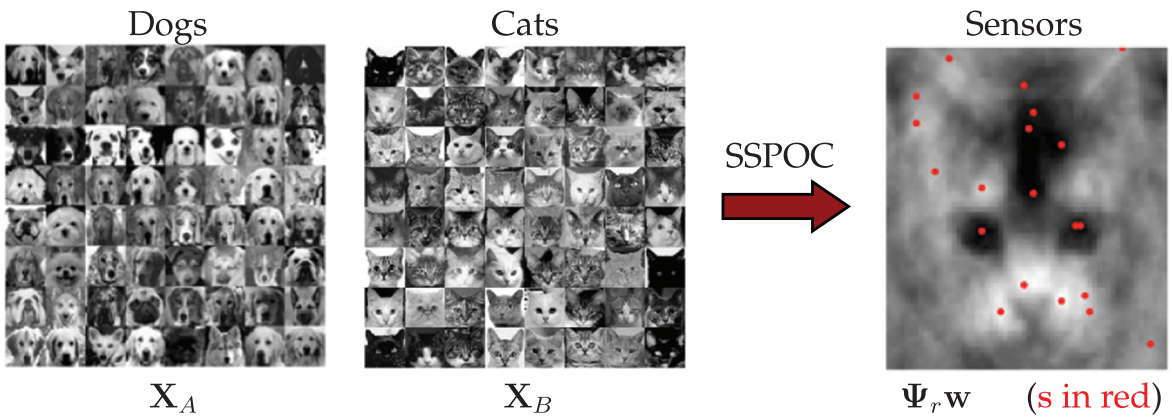

Figure 3.23 Sparse sensor placement optimization for classification (SSPOC) illustrated for optimizing sensors to classify dogs and cats. *Reproduced with permission from B. Brunton et al. [89].*

Suggested Reading

Papers and Reviews

(1) **Regression shrinkage and selection via the lasso**, by R. Tibshirani, *Journal of the Royal Statistical Society B*, 1996 [518].

(2) **Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information**, by E. J. Candès, J. Romberg, and T. Tao, *IEEE Transactions on Automatic Control*, 2006 [111].

(3) **Compressed sensing**, by D. L. Donoho, *IEEE Transactions on Information Theory*, 2006 [150].

(4) **Compressive sensing**, by R. G. Baraniuk, *IEEE Signal Processing Magazine*, 2007 [39].

(5) **Robust face recognition via sparse representation**, by J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009 [560].

(6) **Robust principal component analysis?**, by E. J. Candès, X. Li, Y. Ma, and J. Wright, *Journal of the ACM*, 2011 [110].

(7) **Signal recovery from random measurements via orthogonal matching pursuit**, by J. A. Tropp and A. C. Gilbert, *IEEE Transactions on Information Theory*, 2007 [529].

(8) **Data-driven sparse sensor placement**, by K. Manohar, B. W. Brunton, J. N. Kutz, and S. L. Brunton, *IEEE Control Systems Magazine*, 2018 [366].