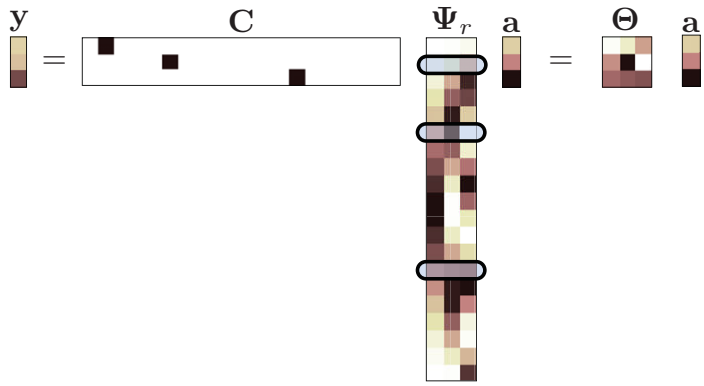

Figure 3.21 Least-squares with r sparse sensors provides a unique solution to $\mathbf{a}$, hence $\mathbf{x}$. Reproduced with permission from Manohar et al. [480].

The examples in this section are in MATLAB. However, extensive Python code for sparse sensing is available at the following:
https://github.com/dynamicslab/pysensors.

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 that 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. 3.21 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 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 $\boldsymbol{\theta}$ (respectively $\mathbf{M}$). 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 semi-definite programming [101, 352], 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* [238] methods originally relied on random subsampling. However, significant performance advances were demonstrated by using principled sampling strategies for reduced-order models (ROMs) [75] in fluid dynamics [751] and ocean modeling [764]. More recently, variants of the so-called *empirical interpolation method* (EIM, DEIM, and Q-DEIM) [55, 170, 214] 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 Section 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 [214] 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 $\boldsymbol{\Psi}_r$:

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

Sensor selection based on the pivoted QR algorithm is quite simple for $p = r$. In MATLAB, the code is

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

In Python the code is

```
from scipy import linalg
Q,R,pivot = linalg.qr(Psi_r.T,pivoting=True)
C = np.zeros_like(Psi_r.T)
C[pivot[:r]] = 1
for k in range(r):
    C[k,pivot[k]] = 1
```

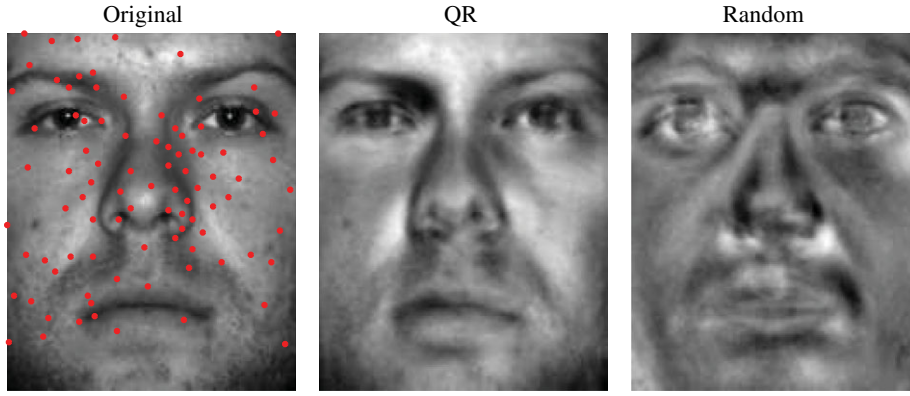

Figure 3.22 (left) Original image and locations of $p = 100$ QR sensors in a $r = 100$ mode library. (middle) Reconstruction with QR sensors. (right) Reconstruction with random sensors.

It may also be advantageous to use oversampling [555], choosing more sensors than modes, so $p > r$. In this case, there are several strategies, and random oversampling is a robust choice.

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. Figure 3.22 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 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. From the QR sensors $\mathbf{C}$ based on $\mathbf{\Psi}_r$, it is possible to reconstruct an approximate image from these sensors. In MATLAB, the reconstruction is given by

```
Theta = C*Psi_r;
y = faces(pivot(1:p),1); % Measure at pivot locations
a = Theta\y;             % Estimate coefficients
faceRecon = Psi_r * a;   % Reconstruct face
```

In Python, the reconstruction is given by

```
Theta = np.dot(C , Psi_r)
y = faces[pivot[:r]]            # Measure at pivot locations
a = np.dot(np.linalg.pinv(Theta),y) # Estimate coefficients
faceRecon = np.dot(Psi_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 [122]. 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

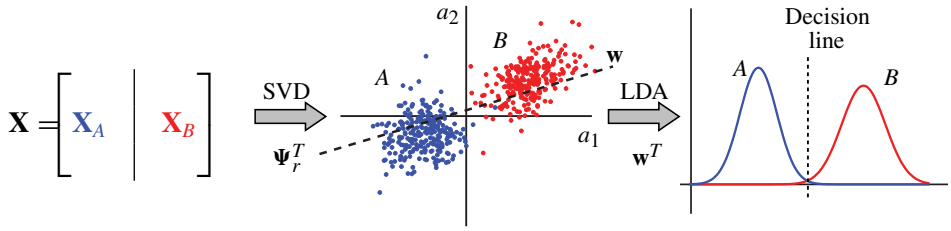

Figure 3.23 Schematic illustrating SVD for feature extraction, followed by linear discriminant analysis (LDA) for the automatic classification of data into two categories A and B. Reproduced with permission from Bai et al. [40].

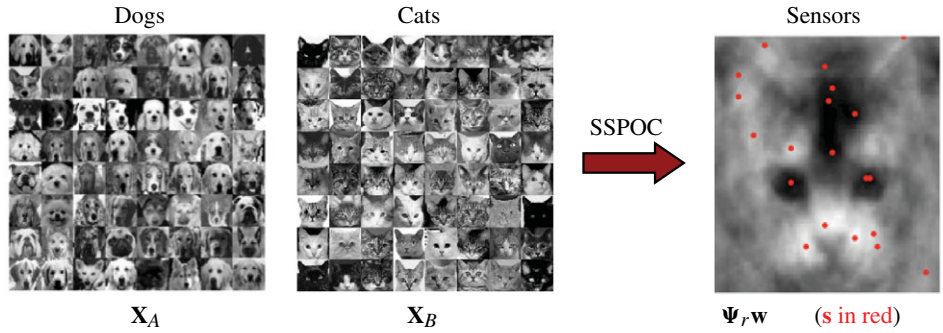

Figure 3.24 Sparse sensor placement optimization for classification (SSPOC) illustrated for optimizing sensors to classify dogs and cats. Reproduced with permission from B. Brunton et al. [122].

between two categories of data, as described in Section 5.6 and shown in Fig. 3.23. Sparse sensors $\mathbf{s}$ that map into this discriminating direction, projecting out all other information, are found by

$$\mathbf{s} = \underset{\mathbf{s}'}{\operatorname{argmin}} \|\mathbf{s}'\|_1 \quad \text{subject to} \quad \boldsymbol{\Psi}_r^T \mathbf{s}' = \mathbf{w}. \tag{3.23}$$

This sparse sensor placement optimization for classification (SSPOC) is shown in Fig. 3.24 for an example classifying dogs versus cats. The library $\boldsymbol{\Psi}_r$ contains the first r *eigen-pets* 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.

Suggested Reading

Papers and reviews

(1) **Regression shrinkage and selection via the lasso**, by R. Tibshirani, *Journal of the Royal Statistical Society B*, 1996 [700].

(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 [155].