Chapter 10
Signal models and basis functions

Contents

10.1 Introduction (s,basis,intro) ................................. 10.1
10.2 (Linear) basis function choices (s,basis,linear) .............................. 10.3
10.3 Nonlinear object parameterizations (s,basis,nonlin) ............................... 10.3
10.4 Synthesis models and dictionary learning (s,basis,synth) ................. 10.5
  10.4.1 Subspace model .............................................. 10.5
  10.4.2 Union of subspaces models .................................... 10.6
  10.4.3 Over-complete dictionary model ................................. 10.6
  10.4.4 Dictionary learning ............................................. 10.7
    10.4.4.1 Dictionary learning: image domain ......................... 10.8
    10.4.4.2 Image reconstruction using a dictionary ................. 10.9
    10.4.4.3 Dictionary learning: data domain ......................... 10.10
10.4.5 Summary .................................................. 10.10
10.5 Bibliography ................................................. 10.11

10.1 Introduction (s,basis,intro)

Most of the reconstruction methods described in this book assume that the underlying continuous-space object of interest is approximated by a series expansion in terms of basis functions [1]:

\[ f(\vec{x}) = \sum_{j=1}^{n_p} x_j b_j(\vec{x}), \]  

(10.1.1)

or equivalently

\[ f = B_\times \vec{x}, \]

where \( B_\times \) is a mapping from \( \mathbb{R}^{n_p} \) or \( \mathbb{C}^{n_p} \) into the space of functions of \( \vec{x} \). The \( j \)th “column” of the operator \( B_\times \) is the basis function \( b_j(\vec{x}) \). Although we often write equality in the model (10.1.1), in practice this representation is an approximation. See Fig. 10.1.1.

The model (10.1.1) is equivalent to assuming that \( f(\vec{x}) \) lies in the \( n_p \)-dimensional subspace of continuous-space functions spanned by the basis functions \( \{b_j(\vec{x})\} \). We usually use such finite-dimensional subspaces because it facilitates computation and because the available data vector \( \vec{y} \) is always finite dimensional.

Nevertheless, in principle it is at least conceivable to try to reconstruct a continuous \( f(\vec{x}) \) from a finite-dimensional measurement vector. This is done routinely in the field of nonparametric regression [2], the generalization of linear regression that allows for fitting smooth functions discussed in 1D in §2.2. However, nonparametric estimation becomes more complicated in 2D problems like tomography.

Van De Walle, Barrett, et al. [3] proposed a pseudo-inverse approach to MRI reconstruction in a continuous-object / discrete-data formulation, based on the general principles of Bertero et al. [4]. If the pseudo-inverse could truly be computed once-and-for-all then such an approach could be practically appealing. However, in practice often there are object-dependent effects, such as nonuniform attenuation in SPECT and magnetic field inhomogeneity in MRI, and these effects preclude precomputation of the required SVDs. So pseudo-inverse approaches are impractical computationally for typical realistic physical models. See also [5, 6].

This chapter summarizes some of the choices for basis functions \( \{b_j(\vec{x})\} \) that have been investigated in the image reconstruction literature, as well as some of the alternatives to the finite-dimensional linear subspace model (10.1.1).
Figure 10.1.1: Illustration of a 1D function $f(x)$ and its approximation by two finite-dimensional subspaces, one using rectangular basis functions, and the other using quadratic B-splines.

Figure 10.1.2: Illustration of a simple 2D function $f(\vec{x})$ and its approximation by a finite-dimensional subspaces using square-pixel basis functions.
10.2 (Linear) basis function choices \((s,\text{basis,linear})\)

Numerous families of basis functions \(\{b_j(\vec{x})\}\) have been investigated in the image reconstruction literature for use with finite-dimensional linear subspace models of the form (10.1.1). This section enumerates several of the options.

- Fourier series (complex / not sparse) [7]
- Circular harmonics (complex / not sparse) [8–11]
- Wavelets (negative values / not sparse) [12]
- Overlapping circles (disks) [13]
- Overlapping spheres (balls) [14] (approximately ellipsoids under nonrigid motion)
- Kaiser-Bessel window functions (blobs) [15]
- Rectangular pixels / voxels (rect functions)
- Dirac impulses (point masses / bed-of-nails / lattice of points / “comb” function)
- “Natural pixels” \(\{s_i(\vec{x})\}\) [16–20]
- B-splines (pyramids) [21–24]
- Polar grid [25–29],
- Logarithmic polar grid [30]
- gaussian functions [31,32]
- Radial basis functions (circularly symmetry) such as [33] \(b(\vec{x}) = (1 - \|\vec{x}/r\|^2)^1_{\{\|\vec{x}\| \leq r\}}\).
- Organ-based voxels (e.g., for PET reconstruction using anatomy from CT image in a PET-CT system) [34–39]

There are many considerations when choosing between the many options listed above. (See [22, 40] for early discussions.)

**Mathematical considerations**

- The subspace should represent \(f(\vec{x})\) “well” with moderate \(n_p\) (approximation accuracy). One meaning of “well” is that the approximation error should be much less than the estimation error.
- In particular, it is desirable for the subspace to be able to represent perfectly a constant (uniform) function.
- It is desirable for the functions \(\{b_j(\vec{x})\}\) to be linearly independent, ensuring that the expansion (10.1.1) is unique. Uniqueness is not essential though because our ultimate goal is finding \(f^{\text{true}}\) and having multiple values of \(\vec{x}\) for which \(f^{\text{true}} = Bx\) (or approximately so) is acceptable. There is really never a \(x^{\text{true}}\) in practice! However, the redundancy of linearly dependent functions \(\{b_j(\vec{x})\}\) may increase computation time so usually should be avoided.
- Orthogonality of basis functions is often advocated for signal modeling because it greatly simplifies computing the coefficients \(x_j\) in (10.1.1) given \(f\). However, in inverse problems we are not given \(f\) so orthogonality is not essential!
- It can be desirable for the representation (10.1.1) to be insensitive to a shift of the basis-function grid, i.e., approximate shift invariance. Smooth basis functions are preferable to discontinuous functions in this respect.
- Similarly, it is desirable for the basis functions to be symmetric and circularly symmetric so that the representation (10.1.1) has approximate rotation invariance and invariance to coordinate system reflections.
- Many bases have the desirable approximation property that one can form arbitrarily accurate approximations to \(f(\vec{x})\) by taking \(n_j\) sufficiently large. (This is related to completeness.) Exceptions include “natural pixels” (a finite set) and the point-lattice “basis” (usually).

**Computational considerations**

- Choice of basis functions affects how “easy” it is to compute elements of the system matrix \(a_{ij}\) and/or perform matrix vector multiplication \(Ax\) (forward projection).
- If the system matrix \(A\) is to be precomputed and stored, then it should be sparse (mostly zeros). Narrower basis functions usually are preferable in this respect.
- In applications where \(f(\vec{x})\) is nonnegative, it should be easy to represent nonnegative functions e.g., if \(x_j \geq 0\), then \(f(\vec{x}) \geq 0\). A sufficient condition is \(b_j(\vec{x}) \geq 0\).

As detailed in Appendix 25, basis function choice, combined with the system model \(\vec{y}_i = \int s_i(\vec{x}) f(\vec{x}) \, d\vec{x}\) determines the elements of the system matrix \(A\) as \(a_{ij} = \int s_i(\vec{x}) b_j(\vec{x}) \, d\vec{x}\). However, many published “projector / backprojector pairs” are not based explicitly on any particular choice of basis.

Some pixel-driven backprojectors could be interpreted implicitly as point-mass object models. This model works fine for FBP, but causes artifacts for iterative methods [41].

Mazur et al. [42] approximate the shadow of each pixel by a rect function, instead of by a trapezoid. “As the shapes of pixels are artifacts of our digitisation of continuous real-world images, consideration of alternative orientation or shapes for them seems reasonable.” However, they observe slightly worse results that worsen with iteration!

10.3 Nonlinear object parameterizations \((s,\text{basis,nonlin})\)

Linear models like (10.1.1) are the most common in image reconstruction, but numerous alternatives have also been investigated, including nonlinear parametric models. These models are often considered in problems with very limited data, necessitating strong assumptions about the object with few degrees of freedom. Usually these models involve estimating both object intensity and shape parameters (e.g., location, radius, etc.), the latter leading to nonlinearity. This section enumerates some of the parametric models that have been used for image reconstruction problems.
Surface-based (homogeneous) models
- Circles / spheres [43, 44]
- Ellipses / ellipsoids [45]
- Superquadrics
- Rectangles with unknown position, size, amplitude [46, 47]
- Polygons [48]
- Bi-quadratic triangular Bezier patches [49]
- Triangulated 3D surface [50–52].

Other models
- Generalized series $f(\vec{x}) = \sum_j x_j b_j(\vec{x}, \theta)$
- Deformable templates [53–55] $f(\vec{x}) = b(T_\theta(\vec{x}))$
- Adaptive tetrahedral meshes (defined by a point cloud) [56]

These type of parametric models have the advantage that they can be considerably more parsimonious than voxelized models. Thus, if they are accurate, they can yield greatly reduced estimation error. These models are particularly compelling in limited-data problems, even if they are oversimplified. (All models are wrong but some models are useful.) The nonlinear dependence of $f(\vec{x})$ on shape parameters and location leads to non-convex cost functions, complicating optimization.

Region of interest (ROI) or “focus of attention” [57–59]
10.4 Synthesis models and dictionary learning *(s,basis,synth)*

Even after using the finite-series expansion (10.1.1), image reconstruction problems are usually under-determined *(i.e.,* \( n_p > n_d \)) or at least very badly ill-conditioned. Thus, attempting to estimate \( x \) solely by finding a minimizer of a data-fit term \( L(x) \) is rarely adequate. Chapter 1 and Chapter 2 described using *regularization or prior models* to (indirectly) "constrain" the estimate \( \hat{x} \). This section describes synthesis-based approaches to modeling \( x \). The basic idea underlying all of these methods is that we must find some way to reduce the *degrees of freedom* of the estimator \( \hat{x} \).

### 10.4.1 Subspace model

The classical linear approach to reducing degrees of freedom is to assume that \( x \) lies in the subspace \( S \) spanned by some \( n_p \times r \) basis matrix \( B \) with \( r \ll n_p \) (having full column rank):

\[
x \approx Bz,
\]

(10.4.1)

for some unknown coefficient vector \( z \in \mathbb{R}^r \). (This is equivalent mathematically to using a different set of \( r \) basis functions in (10.1.1).) Then one estimates \( z \) from the data by minimizing the data-fit term:

\[
\hat{x} \triangleq B\hat{z}, \quad \hat{z} \triangleq \arg \min_{z \in \mathbb{R}^r} L(Bz).
\]

If \( r \) is sufficiently small relative to the data dimension \( n_d \), then this problem may be well conditioned and have a unique minimizer.

One can “learn” the subspace basis \( B \) from training data \( X = [x_1 \ldots x_M] \), provided \( M > r \). We want to find a matrix \( B \) (with linearly independent columns) such that each \( x_m \) approximately lies in its span, *(i.e.,* \( \|x_m - Bz_m\| \) should be small, where \( z_m = \arg \min_z \|x_m - Bz\| \)). It is simpler (and loses no generality) to require that the columns of \( B \) be orthonormal, leading to the optimization problem:

\[
\arg \min_{B:B^TB=I_r} \min_{Z \in \mathbb{R}^{r \times M}} \|X - BZ\|_F^2.
\]

(10.4.2)

This is a non-convex problem and there is not a unique solution for \( B \), but one of the global minimizers is to let \( B \) be the first \( r \) eigenvectors of the \( n_p \times n_p \) sample covariance matrix \( XX^T = \sum_{m=1}^M x_m x_m^T \), assuming the eigenvalues \( \lambda_k \) are ordered from largest to smallest. If \( \lambda_r > \lambda_{r+1} \) then the subspace spanned by this \( B \) is unique even if \( B \) itself is not unique (one can permute its columns for example). But in the rare case where \( \lambda_r = \lambda_{r+1} \), then the subspace is not unique but still \( B \) is a global minimizer. This approach is called *principal components analysis* *(PCA)*. In practice, one computes \( B \) by finding the *singular value decomposition* *(SVD)* of \( X \) and taking the first \( r \) left singular vectors.

It is difficult to visualize subspaces in dimensions higher than \( n_p = 3 \). Fig. 10.4.1 illustrates an \( n_p = 2 \) case where a \( r = 1 \) dimensional subspace is a reasonable model.

![Figure 10.4.1: Illustration of (synthetic) data that is well approximated by a 1D subspace.](fig_basis_synch_subspace1)

The subspace approach works fine for regression problems like fitting a line \( (r = 2) \) to a scatter plot of data. Unfortunately, in image reconstruction problems rarely is there a single low-dimensional subspace that adequately describes the images of interest. The remainder of this section describes models that are more flexible yet retain some aspects of the “low rank” nature of the classical subspace model.
10.4.2 Union of subspaces models

A more general model is to assume \( x \) belongs (at least approximately) to one of \( K \) subspaces where the \( k \)th subspace is spanned by the columns of a \( n_p \times r_k \) matrix \( B_k \), where the rank \( r_k \ll n_p \), as follows:

\[
x \in \bigcup_{k=1}^{K} S_k, \quad S_k = \{ B_k z_k : z_k \in \mathbb{R}^{r_k} \}.
\]

If we can somehow determine which subspace is appropriate, then essentially we have only a \( r_k \)-dimensional estimation problem within that subspace. One possible formulation is

\[
\hat{x} \triangleq B_k \hat{z}_k, \quad \hat{z}_k \triangleq \arg \min_{z \in \mathbb{R}^{r_k}} L(B_k z), \quad \hat{k} \triangleq \arg \min_k L(B_k \hat{z}_k).
\]

Methods for learning the subspace bases \( \{ B_k \} \) from training data are given in [60–63]. Often the models include hierarchical or tree-based structure [64].

A union of \( K \)-subspaces is a generalization of the \( K \)-means approach to data clustering.

Fig. 10.4.2 illustrates an \( n_p = 3 \) case with \( K = 3 \) subspaces where \( r_1 = 2 \) and \( r_2 = r_3 = 1 \).

![Figure 10.4.2: Illustration of (synthetic) data in \( \mathbb{R}^{n_p} \) where \( n_p = 3 \) that is well approximated by \( K = 3 \) subspaces, the first of dimension \( r_1 = 2 \) and \( r_2 = r_3 = 1 \) for the other two subspaces.](image)

10.4.3 Over-complete dictionary model

An alternative to the union-of-subspaces model is to represent \( x \) using a linear combination of a small number of atoms from an over-complete dictionary \( D \) of size \( n_p \times K \) where \( K > n_p \) as follows [65]:

\[
x \approx Dz, \quad \|z\|_0 = \sum_{k=1}^{K} 1\{z_k \neq 0\} \leq r.
\]

In other words, we represent (or approximate) \( x \) using a linear model with an \( r \)-sparse coefficient vector. Certainly this model can be more expressive than the single subspace model (10.4.1). Strictly speaking, (10.4.3) is also a union of subspaces model where there are \( \binom{K}{r} \) possible \( r \)-dimensional subspaces. This representation is “more flexible” yet “less structured.”

For a given dictionary \( D \), one can approach an image reconstruction problem as follows:

\[
\hat{x} \triangleq D \hat{z}, \quad \hat{z} \triangleq \arg \min_{z \in \mathbb{R}^K} \|Dz\|_0 \text{ sub. to } \|z\|_0 \leq r.
\]

The idea here is that after we somehow determine the appropriate subspace (i.e., the span of the columns of \( D \) corresponding to the non-zero elements of \( z \)), then the minimization problem is “only” \( r \)-dimensional, where typically
r \ll n_p$, so the problem may be well conditioned within that subspace. Note that the length of the coefficient vector $z$ in the subspace model (10.4.1) is only $r$ whereas in the over-complete dictionary model (10.4.3) it is $K \geq n_p \gg r$.

Fig. 10.4.3 illustrates an $n_p = 2$ case with $K = 3$ atoms where $r = 1$ provides a reasonable model. In the case $n_p = 2$, the union-of-subspaces model and the over-complete dictionary model are identical for $r = 1$. To illustrate the distinction between these models requires $n_p > 2$. For example, for $n_p = 3$, if we have a dictionary with $K$ atoms and $r = 2$, then (10.4.3) corresponds to the union of all $K\choose 2$ pairs of atoms in the dictionary. In contrast a typical union-of-subspaces model would consist of just a few such pairs.

![Illustration of (synthetic) data for the distinction between these models requires $n_p > 2$.](https://example.com/fig_basis_synth_subspace2)

**Figure 10.4.3:** Illustration of (synthetic) data for $n_p = 2$ that is well approximated by $K = 3$ subspaces of dimension $r = 1$.

There are many issues to address when using an over-complete dictionary like (10.4.3).

- One must choose the allowed sparsity level $r$, which is rarely known *a priori* for a given application.
- For problems with gaussian noise of known variance $\sigma^2$, one can avoid picking $r$ by replacing the optimization problem (10.4.4) with the following alternative formulation:

$$\hat{x} \doteq D\hat{z}, \quad \hat{z} \doteq \arg\min_{z \in \mathbb{R}^K} \|z\|_0 \text{ sub. to } \|y - ADz\|_2^2 \leq n_d\sigma^2.$$ 

Here we find the sparsest coefficient vector that leads to a data-fit term within a given error tolerance. The tolerance $n_d\sigma^2$ is appropriate if $x_{\text{true}} = Dz$, but we have seen in §2.5.2.1 that in general the *discrepancy principle* yields suboptimal regularization levels when fitting data.

- The minimization problem in (10.4.3) is non-convex and combinatorial. Often this problem is circumvented by using greedy algorithms or replacing $\|z\|_0$ with $\|z\|_1$ leading to a convex optimization problem.
- The size of the dictionary $D$ would be enormous if it were used to represent the entire image. For a $256^2$ image (tiny by modern digital photography standards), $n_p = 2^{16}$ and if $K = 2n_p$ then $D$ has $2^{33}$ elements and storing $D$ as single-precision (4-byte) floating point values would need $2^{35}$ bytes or 32 GBytes. Therefore usually the image $x$ is partitioned into small patches, e.g., each of $8 \times 8$ pixels, and each patch of 64 values is represented individually by a sparse coefficient vector for an over-complete dictionary $D_0$ of size $8^2 \times K$ where $K > 8^2$. If $K = 128$ then only 32 KBytes are needed to store $D_0$. Mathematically, in this representation the overall model for $x$ corresponds to the *direct sum* [wiki] of the subspaces for each image patch.

Issues with patch-wise representations include possible image artifacts at the boundaries between patches, leading to “blocky” appearance, particularly if $r$ is small. Furthermore, this representation is not *shift invariant*.

To avoid these drawbacks, one could use overlapping patches, and then one must choose how to combine the values from each patch to make the final image estimate $\hat{x}$.

- Finally, one must select the dictionary $D$, including $K$, the number of atoms. The next section describes a popular method called the *K-SVD* approach.

### 10.4.4 Dictionary learning

In the context of image reconstruction, there are two main types of methods for *dictionary learning*, i.e., for estimating $D$. One approach is to start with image-domain training data, typically obtained from “fully sampled” and/or “high SNR” cases, and then we use the learned dictionary $D$ to help reconstructed “under sampled” and/or “low SNR” data. The other approach is to try to learn the dictionary $D$ adaptively from the given measurement vector $y$ (sinogram, k-space, etc.) [66, 67]. The following sections consider both types of methods, starting with the simpler the image-domain approach.
10.4.4.1 Dictionary learning: image domain

Given training data \( X = [x_1 \ldots x_M] \) (e.g., patches from good quality images), we can learn a dictionary \( D \) using the K-SVD method [68]. We start by picking \( K \); clearly we need \( M \gg K \) or the dictionary learning problem would degenerate to just using the training data.

The goal is to find a dictionary \( D = [d_1 \ldots d_K] \) such that every example in the training data is well approximated by the dictionary with a \( r \)-sparse coefficient vector. In other words, for each \( x_m \) there should be some coefficient vector \( z_m \) with \( \| z_m \|_0 \leq r \) for which \( \| x_m - Dz_m \| \) is small. A reasonable mathematical criterion is

\[
\arg\min_{D \in \mathbb{R}^{np \times K}} \sum_{m=1}^{M} \min_{z_m \in \mathbb{R}^r} \| x_m - Dz_m \|_2^2, \quad Z_r \triangleq \{ z \in \mathbb{R}^K : \| z \|_0 \leq r \}.
\]

Often this minimization problem is written concisely by grouping together the coefficient vectors \( Z \triangleq [z_1 \ldots z_M] \) and writing

\[
\arg\min_{D \in \mathbb{R}^{np \times K}} \min_{Z \in \mathbb{R}^M} \| X - DZ \|_{Frob} \quad \text{sub. to } z_m \in Z, \forall m.
\]

This expression is slightly more concise, but somewhat obscures the fact that for a given \( D \), we can compute each coefficient vector \( z_m \) in parallel (independently).

The K-SVD approach [68] alternates between updating the coefficients \( Z \) for a given candidate dictionary \( D \), and then updating sequentially each atom \( d_m \) in the dictionary \( D \) using an SVD. This alternating minimization method decreases the cost function \( \| X - DZ \|_{Frob} \) every iteration and thus the cost function values converge to some non-negative value. However, contrary to the claims in [68], this monotonicity is insufficient to ensure convergence of \( D \) to a local minimizer without further proof. (See Example 11.1.2.)

For a given candidate dictionary \( D \) in this alternating minimization process, the problem of updating \( Z \) has many names in the literature, including sparse coding, sparse approximation, sparse synthesis, and atom decomposition. All of these terms refer to the following minimization problem:

\[
\hat{z}_m \triangleq \arg\min_{z \in \mathbb{R}^r} \| x_m - Dz \|_2^2, \quad m = 1, \ldots, M. \tag{10.4.5}
\]

Methods for solving this combinatorial optimization problem are called pursuit algorithms. The simplest methods are greedy algorithms that select one atom at a time. See [69] for a survey.

The matching pursuit (MP) method chooses the atom having the largest (absolute) inner product with the residual, as follows [70] [wiki].

\begin{verbatim}
Matching pursuit
Input: Dictionary D = [d_1 \ldots d_K] and signal vector x ∈ \mathbb{R}^p.
 r = x (initialize residual)
z = 0_K (initialize sparse coefficient vector)
for n = 1, \ldots, r
    k_n = \arg \max_k |⟨d_k, r⟩| \quad \text{(atom index selection)}
    z_{k_n} := ⟨d_{k_n}, r⟩ \quad \text{(coefficient)}
    r := r - z_{k_n} d_{k_n} \quad \text{(updated residual)}
end

Output: atom indices k_1, \ldots, k_r and coefficients z_{k_1}, \ldots, z_{k_r} such that x ≈ Dz = \sum_{n=1}^{r} z_{k_n} d_{k_n}
\end{verbatim}

The simple inner product \( ⟨d_m, r⟩ \) provides an appropriate coefficient if the selected atoms are orthonormal, but if they are not then one can obtain a better fit (smaller residual) more generally performing a least-squares fit of the coefficients after each new atom is selected. Equivalently, we project the residual onto the span of all selected atoms at each step. The orthogonal matching pursuit (OMP) method uses this variation, along with the (natural) constraint that each atom can be picked only once [71] [wiki].

Being greedy algorithms, they typically find local minimizers of the sparse coding problem (10.4.5). An alternative is to replace the non-convex problem (10.4.5) with one of the following convex problems:

\[
\min_{z \in \mathbb{R}^K} \| z \|_1 \quad \text{sub. to } \| x_m - Dz \|_2^2 \leq \varepsilon
\]

\[
\min_{z \in \mathbb{R}^K} \frac{1}{2} \| x_m - Dz \|_2^2 + \beta \| z \|_1. \tag{10.4.6}
\]

These optimization problems are called basis pursuit denoising [wiki]. As \( \beta \to 0 \) the problems become basis pursuit [wiki]. To use (10.4.6) for updating \( Z \) for the K-SVD method, one can take the \( r \) largest non-zero coefficients of \( \hat{z} \), or adjust \( \beta \) so that \( \hat{z} \) has \( r \) non-zero coefficients.
Having updated the sparse coefficients $Z$, the next step of the K-SVD method is to update the dictionary $D$. The K-SVD uses block coordinate descent to update $D$ by minimizing $\|X - DZ\|_{\text{Frob}}$ with respect to one column of $D$ (and the corresponding non-zero elements of $Z$) sequentially. To update $d_k$, we consider all training samples for which the $k$th element of the corresponding coefficient vector is nonzero, i.e., $M_k \triangleq \{m : z_{mk} \neq 0\}$. We then compute the residual associated with those training samples, excluding $d_k$, i.e., $r_m \triangleq \sum_{j \neq k} z_{mj} d_j$, $m \in M_k$.

We then find a new vector $d_k$ that best spans these residual vectors by taking the first singular vector of the SVD of $\{r_m\}$. This step is analogous to solving (10.4.2) with $r = 1$. The corresponding elements of $Z$ are then updated with this new atom $d_k$, without changing the sparsity pattern of $Z$. This process is repeated for $k = 1, \ldots, K$, after which the algorithm returns to the sparse coding step to update $Z$. Several implementation tricks related to initialization and pruning are given in [68], such as always including the constant vector $1_n$ as one of the atoms.

Fig. 10.4.4, taken from [68, Fig. 5], compares a dictionary learned from 500 $8 \times 8$ patches from face images, compared to Harr and DCT bases.

Figure 10.4.4: A dictionary for $8 \times 8$ patches learned from face images, compared to (over-complete) Harr and DCT bases.

For large $M$ (many training samples) one can use stochastic gradient descent [72]. For some applications, constraints such as nonnegativity on the atoms is appropriate [73].

For improvements see [74–76].

For software, see http://www.cs.technion.ac.il/~elad/software.

10.4.4.2 Image reconstruction using a dictionary

Having selected a dictionary $D$, e.g., by learning it from training data, the image reconstruction problem is to estimate $x$ from data $y$ using the sparsity model (10.4.3). There are several possible approaches.

The traditional synthesis approach is to estimate the coefficients as follows:

$$\hat{x} = D\hat{z}, \quad \hat{z} = \arg \min_{z} \|z\|_p \text{ sub. to } \|y - ADz\|_2^2 \leq \varepsilon.$$ 

Instead of requiring that $x$ be synthesized exactly from the dictionary, another option is to encourage $\hat{x}$ to be similar to such an image: [67]:

$$(\hat{x}, \hat{z}) = \arg \min_{x, z} \frac{1}{2} \|y - Ax\|_{W_{1/2}}^2 + \beta \|x - Dz\|_2^2 \text{ sub. to } \|z\|_0 \leq r.$$ 

Another option is the following convex relaxation of the previous problem [67]:

$$(\hat{x}, \hat{z}) = \arg \min_{x, z} \frac{1}{2} \|y - Ax\|_{W_{1/2}}^2 + \beta \|x - Dz\|_2^2 + \alpha \|z\|_1.$$ 

For these last two formulations, a natural optimization approach is to alternate between updating the image $x$, which is a quadratic problem, and the coefficient vector $z$, which is a sparse coding problem. (One could apply OMP or FISTA.) For further examples, see [77–83]. Interestingly, the convex form above is closely related to the augmented Lagrangian for the convex (but non-smooth) optimization problem

$$\hat{x} = D\hat{z}, \quad \hat{z} = \arg \min_{z} \frac{1}{2} \|y - ADz\|_{W_{1/2}}^2 + \alpha \|z\|_1.$$
10.4.4.3 Dictionary learning: data domain

A potential drawback of any dictionary learned from training data is that the object \( x \) being reconstructed might not have a sparse representation in that dictionary. Furthermore, there will always be noise in training data that will affect dictionary learning. Such considerations have motivated research on jointly reconstructing an image and learning a dictionary (or part of a dictionary) \([66, 67, 77]\). An example formulation is

\[
\arg\min_{x, D, Z} \frac{1}{2} \| y - Ax \|_2^2 + \sum_{l=1}^{L} \frac{\beta_l}{2} \| R_l x - D z_l \|_2^2 \quad \text{sub. to} \quad \| z_l \|_0 \leq r,
\]

where \( R_l \) is a matrix that extracts the \( l \)th patch from the image \( x \). In this setting it is essential to use the dictionary to represent patches rather than the entire image. One can imagine many variations (including convex relaxations) of such formulations, as well as combining adaptively learned dictionaries with dictionaries learned from training data and some predetermined atoms (such as a DC component); one could also include conventional analysis regularizers, perhaps with a small regularization parameter.

An formulation that is convex with respect to each of the parameters individually (but not collectively) is \([84]\):

\[
\arg\min_{x, D, Z} \frac{1}{2} \| y - Ax \|_2^2 + \sum_{l=1}^{L} \left( \frac{\beta_l}{2} \| R_l x - D z_l \|_2^2 + \alpha \| z_l \|_1 \right)
\]

To avoid scale ambiguity, here one must constrain \( D \), e.g., by requiring that each column \( d_k \) have unit norm. An alternating minimization approach is natural.

One can also learn transforms for analysis formulations \([85–87]\).

10.4.5 Summary

Unions of subspaces, such as over-complete dictionaries with sparse coefficient vectors, provide a mechanism for dimension reduction that is somewhat akin to classical subspaces but more flexible. The drawback is that the resulting optimization problems often have non-convex aspects.

Analysis formulations are “negative” in the sense that they discourage the reconstructed image \( \hat{x} \) from departing from prior assumptions about the relationships between groups of pixels (such as neighbors). In contrast, synthesis formulations are “positive” in the sense that they encourage (or require) the reconstructed image \( \hat{x} \) to be expressible in terms of the prior information (e.g., linear combinations of dictionary atoms). Identifying which formulation (or combination thereof) is best for a given application is an open problem; see \([88, 89]\).
\section*{10.5 Bibliography}

\begin{thebibliography}{10}
\bibitem{Cormack:63} A. M. Cormack. Representation of a function by its line integrals, with some radiological applications. \textit{J. Appl. Phys.}, 34(9):2722–7, September 1963. 10.3
\bibitem{Cormack:64} A. M. Cormack. Representation of a function by its line integrals, with some radiological applications II. \textit{jappliph}, 35(10):2908–13, October 1964. 10.3
\bibitem{Hanson:83} K. M. Hanson and G. W. Wecksung. Local basis-function approach to computed tomography. \textit{Appl. Optics}, 24(23):4028–39, December 1985. 10.3
\end{thebibliography}


