13 Transforms

The coming chapters will revisit the ground we just covered, finding significant problems and remarkable capabilities lurking behind apparently innocuous assumptions made in last chapter's introduction to function fitting. Here we ask the easily overlooked question of whether data is best analyzed in the form that it is given (*hint:* the answer is frequently no). This is a question about *representation* – what's the best way to view the data to highlight the features of interest? The goal will be to boil a set of measurements down to a smaller set that is more independent, freeing subsequent analysis from having to rediscover the structure. A good representation can go a long way towards solving a difficult problem, and conversely a bad one can doom an otherwise well-intentioned effort.

13.1 ORTHOGONAL TRANSFORMS

We will frequently be concerned with *orthogonal transformations*. These are ones that are particularly simple to undo, an important feature since we don't want our transformation to throw away information in the data unless we tell it to.

A matrix is *orthogonal* if its inverse is equal to its transpose,

$$\mathbf{M}^T \cdot \mathbf{M} = \mathbf{I} \quad , \tag{13.1}$$

where as usual the *transpose* is denoted by

$$\mathbf{M}^T_{ij} \equiv \mathbf{M}_{ji} \tag{13.2}$$

and $\mathbf{I}$ is the identity matrix with 1s on the diagonal and 0s elsewhere. Multiplication of a vector by an orthogonal matrix defines an orthogonal transformation on the vector. For a complex matrix the *adjoint* is the complex conjugate of the transpose

$$\mathbf{M}^\dagger_{ij} \equiv \mathbf{M}^*_{ji} \quad . \tag{13.3}$$

If the adjoint is the inverse,

$$\mathbf{M}^\dagger \cdot \mathbf{M} = \mathbf{I} \quad , \tag{13.4}$$

then $\mathbf{M}$ is *unitary*. The column or row vectors $\vec{v}_i$ of an orthogonal matrix are not only *orthogonal*, $\vec{v}_i \cdot \vec{v}_j = 0$ $(i \neq j)$, they are *orthonormal*, $\vec{v}_i \cdot \vec{v}_j = \delta_{ij}$, but by convention the matrix itself is still usually just called orthogonal.

An important property of the adjoint is that it interchanges the order of a product of matrices:

$$(\mathbf{A} \cdot \mathbf{B})^\dagger = \mathbf{B}^\dagger \cdot \mathbf{A}^\dagger \quad . \tag{13.5}$$

Remember also that matrix multiplication is *distributive* ($\mathbf{A} \cdot (\mathbf{B} + \mathbf{C}) = \mathbf{A} \cdot \mathbf{B} + \mathbf{A} \cdot \mathbf{C}$) and *associative* ($\mathbf{A} \cdot (\mathbf{B} \cdot \mathbf{C}) = (\mathbf{A} \cdot \mathbf{B}) \cdot \mathbf{C}$), but need not be *commutative* ($\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$).

Now consider a linear transformation on a column vector $\vec{x}$ to a new one $\vec{y} = \mathbf{M} \cdot \vec{x}$. The *Euclidean norm* of $\vec{x}$ is its length as measured by the sum of the squares of the elements,

$$|\vec{x}|^2 = \vec{x}^\dagger \cdot \vec{x} = \sum_i x_i^* x_i = \sum_i |x_i|^2 \quad . \tag{13.6}$$

If $\mathbf{M}$ is unitary, then the norm of $\vec{y}$ is

$$\begin{aligned} |\vec{y}|^2 &= |(\mathbf{M} \cdot \vec{x})^\dagger \cdot (\mathbf{M} \cdot \vec{x})| \\ &= |(\vec{x}^\dagger \cdot \mathbf{M}^\dagger) \cdot (\mathbf{M} \cdot \vec{x})| \\ &= |\vec{x}^\dagger \cdot (\mathbf{M}^\dagger \cdot \mathbf{M}) \cdot \vec{x}| \\ &= |\vec{x}^\dagger \cdot \vec{x}| \\ &= |\vec{x}|^2 \quad . \end{aligned} \tag{13.7}$$

A unitary (or orthogonal) transformation preserves the norm of a vector. It rotates a data point to a new location, but doesn't change its distance from the origin. This means that it can rearrange the points but not do something as nasty as make some of them disappear.

13.2 FOURIER TRANSFORMS

The *Discrete Fourier Transformation* (*DFT*) is a familiar example of a unitary transformation (Problem 12.1). Given a data vector $\{x_0, x_1, \ldots, x_{N-1}\}$, the DFT is defined by

$$\begin{aligned} X_f &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} e^{2\pi i f n/N} x_n \\ &\equiv \sum_{n=0}^{N-1} M_{fn} x_n \\ &= \mathbf{M} \cdot \vec{x} \quad , \end{aligned} \tag{13.8}$$

and the corresponding inverse transform by

$$x_n = \frac{1}{\sqrt{N}} \sum_{f=0}^{N-1} e^{-2\pi i f n/N} X_f \quad . \tag{13.9}$$

The X_f are the coefficients for an expansion of the vector in a basis of periodic complex functions. For real-valued signals and transforms the related *Discrete Cosine Transformation* (*DCT*) can be used (Problem 12.2).

Computing the DFT requires multiplying the data vector by the transform matrix. Finding one element needs N multiplies and adds, and there are N elements, so this

appears to be an $\mathcal{O}(N^2)$ algorithm. Remarkably, and significantly, this is not the case. Notice the the DFT can be split into two sums as follows:

$$\begin{aligned}
X_f &= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} e^{2\pi i f n/N} x_n \\
&= \frac{1}{\sqrt{N}} \sum_{n=0}^{N/2-1} e^{2\pi i f(2n)/N} x_{2n} + \frac{1}{\sqrt{N}} \sum_{n=0}^{N/2-1} e^{2\pi i f(2n+1)/N} x_{2n+1} \\
&= \frac{1}{\sqrt{N}} \sum_{n=0}^{N/2-1} e^{2\pi i f(2n)/N} x_{2n} + \frac{e^{2\pi i f/N}}{\sqrt{N}} \sum_{n=0}^{N/2-1} e^{2\pi i f(2n)/N} x_{2n+1} \\
&= \frac{1}{\sqrt{N}} \sum_{n=0}^{N/2-1} e^{2\pi i f n/(N/2)} x_{2n} + \frac{e^{2\pi i f/N}}{\sqrt{N}} \sum_{n=0}^{N/2-1} e^{2\pi i f n/(N/2)} x_{2n+1} \\
&= X_f^{even} + e^{2\pi i f/N} X_f^{odd} \quad . \qquad (13.10)
\end{aligned}$$

Instead of one N-point transform we've broken it into two $N/2$-point transforms, one on the even points and one on the odd ones. This requires $\mathcal{O}[(N/2)^2] + \mathcal{O}[(N/2)^2] = \mathcal{O}(N^2/2)$ steps to do the transforms and one final multiplication and addition to combine each element, instead of the original $\mathcal{O}(N^2)$ steps. The even and odd transforms can likewise be split, and so forth, until we've broken the calculation into N single-point transforms. Reassembling each of them through the hierarchical factoring takes $\log_2 N$ adds and multiplies, for a total of $\mathcal{O}(N \log_2 N)$ steps. If $N = 10^8$, doing this requires $\mathcal{O}(10^9)$ steps (about a second at 1 GFlop), versus $\mathcal{O}(10^{16})$ operations for the DFT (a few months at a GFlop). Quite a savings! The modern incarnation of this clever idea is called the *Fast Fourier Transform* (*FFT*) and is associated with Cooley and Tukey [Cooley & Tukey, 1965], but it has a long history dating all the way back to Gauss in 1805. It is an example of the powerful algorithm design principle of *divide-and-conquer*: if you can't solve a difficult problem, split it into successively smaller problems until they can be solved and then recombine them to find the answer [Aho *et al.*, 1974].

The clarity of the FFT implementation hides many subtleties in its application [Oppenheim & Schafer, 2009]. The highest frequency possible in a DFT is $f = 1/2$; beyond that the 2π periodicity of the exponential will wrap still higher components in x_n onto lower frequencies. This is the phenomenon of *aliasing* and requires that a signal be sampled at more than twice the highest frequency of interest (called the *Nyquist frequency*). And since the transform is done over a finite time it is equivalent to transforming an infinite series multipied by a finite-length pulse. Since multiplication in the time domain is equal to convolution in the frequency domain, and the Fourier transform of a pulse is a *sinc* function $\sin(2\pi f \Delta T)/(\pi f)$, sharp features in the transform get spread out by the finite window and spurious *side-lobes* appear. There are many other ways to *window* data with weighting functions other than a rectangular step, in order to optimize desired attributes such as spectral resolution, sidelobe suppression, or phase uniformity. Finally, remember that the discrete sampling of the spectrum done by the DFT can miss important features that lie between the points of the transform.

The FFT is one of the most important algorithms in all of numerical mathematics. Beyond the many applications we've already seen for Fourier transforms it crops up in

places where you might not expect it, such as speeding up the multiplication of two long numbers (which is really just a convolution [Knuth, 1997]). When Cooley (then at IBM) first presented the FFT, IBM concluded that it was so significant it should be put in the public domain to prevent anyone from trying to patent it, and so it was published openly. Ironically, its very success has made this kind of behavior less common now.

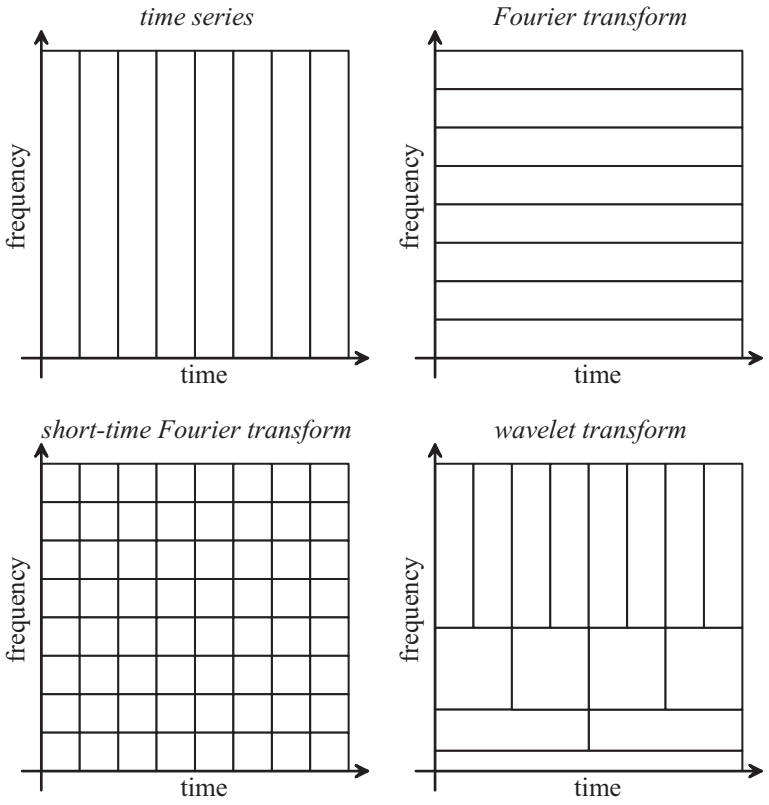

Figure 13.1. Division of time-frequency spaces by the coefficients of discrete transforms.

13.3 WAVELETS

Wavelets are families of orthogonal transformations that generalize Fourier transforms in a very important way by introducing locality (Figure 13.1). Trigonometric functions are defined everywhere. This makes them good at describing global properties, such as the frequency of a signal, but very bad at describing locally varying properties. On the other hand, a time series represents a signal as a series of local impulses, which have an infinite spectrum of Fourier coefficients. A sine wave is most conveniently expressed in the frequency domain, and a step function is much more naturally defined in the time domain. In between these extremes lie most signals of interest, for which neither a global nor a local representation is best. A *short-time Fourier transform* (*STFT*) tries to do this by transforming short windows of data. This has the problem that low-frequency

estimates need big windows to be meaningful, while high-frequency estimates need small windows to be relevant. This is exactly the happy compromise that wavelets provide, retaining a useful notion of both location and frequency.

Wavelets can be understood as a hierarchical *filter bank*, shown in Figure 13.2. A signal is applied to two filters, one passing the high-frequency part of the signal and the other passing the low-frequency part. Then, the low-frequency part goes through a pair of filters, separating it into a new high-frequency component and an even lower-frequency one. This procedure is continued until the signals at the bottom are left with a single point. Since we don't want the transform to throw away information unless we explicitly decide to, each of these steps is done invertibly.

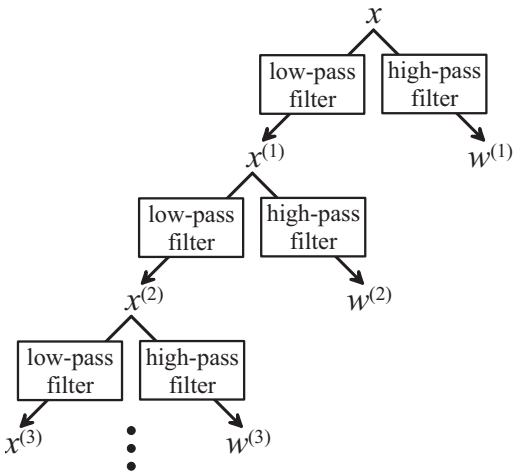

Figure 13.2. Interpretation of the wavelet transform as a hierarchical filter bank.

The earliest wavelets were based on expanding a function in terms of rectangular steps, the *Haar wavelets* [Haar, 1910]. This is usually a very poor approximation; we will instead start with the *Daubechies wavelets*, which are among the simplest but still most important families [Daubechies, 1988]. Given a record of N points x_n, the first step is to write down a linear filter

$$y_n = \sum_{i=0}^{M-1} b_i x_{n-i} \tag{13.11}$$

that is zero for "smooth" signals. To design it we certainly want it to vanish for a constant, so that (taking the order $M = 4$ for example)

$$b_0 \cdot 1 + b_1 \cdot 1 + b_2 \cdot 1 + b_3 \cdot 1 = 0 \quad . \tag{13.12}$$

The next thing that we could ask for is that it vanish for a linear ramp

$$b_0 \cdot 0 + b_1 \cdot 1 + b_2 \cdot 2 + b_3 \cdot 3 = 0 \quad . \tag{13.13}$$

Since this is a linear filter it will then vanish for any $x = \alpha n + \beta$. It will turn out that for a fourth-order wavelet this is all that we can do; given the other constraints to be

included six terms will be needed if we want it to vanish for a quadratic curve, and so forth. Next, we want to define another filter

$$z_n = \sum_{i=0}^{M-1} c_i x_{n-i} \tag{13.14}$$

that responds exactly oppositely, being large for smooth signals and small for nonsmooth signals. A linear filter is just a convolution of the signal with the filter's coefficients, so the series of the coefficients is the signal that the filter responds maximally to (Chapter 19). Therefore, if the output of our second filter vanishes when the coefficients of the first one are input to it, it will be as unlike the first one as two linear filters can be. This means that we want

$$\sum_{i=0}^{M-1} c_i b_i = 0 \tag{13.15}$$

(remember that because a linear filter is a convolution, the associated time series flips the order of the coefficients, and so both have the same index in this sum). A pair of filters with this property are called *quadrature mirror* filters. For $M = 4$ the equation to be solved is

$$c_0 b_0 + c_1 b_1 + c_2 b_2 + c_3 b_3 = 0 \quad . \tag{13.16}$$

By inspection, this can be enforced by flipping the order of the coefficients as well as the sign of every other one:

$$b_0 = c_3 \quad b_1 = -c_2 \quad b_2 = c_1 \quad b_3 = -c_0 \quad . \tag{13.17}$$

We now have two filters: one is large for the smooth parts of the signal, and the other for the nonsmooth parts. To apply them to an input vector we can write it as a matrix problem:

$$\begin{bmatrix}
c_0 & c_1 & c_2 & c_3 & & & & & & & & \\
c_3 & -c_2 & c_1 & -c_0 & & & & & & & & \\
 & & c_0 & c_1 & c_2 & c_3 & & & & & & \\
 & & c_3 & -c_2 & c_1 & -c_0 & & & & & & \\
 & & & & & & \ddots & & & & & \\
 & & & & & & & c_0 & c_1 & c_2 & c_3 & \\
 & & & & & & & c_3 & -c_2 & c_1 & -c_0 & \\
c_2 & c_3 & & & & & & & & c_0 & c_1 & \\
c_1 & -c_0 & & & & & & & & c_3 & -c_2 &
\end{bmatrix}
\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{N-1} \end{pmatrix}
=
\begin{pmatrix} x_0^{(1)} \\ w_0^{(1)} \\ x_1^{(1)} \\ w_1^{(1)} \\ \vdots \\ x_{N/2-2}^{(1)} \\ w_{N/2-2}^{(1)} \\ x_{N/2-1}^{(1)} \\ w_{N/2-1}^{(1)} \end{pmatrix} \tag{13.18}$$

(all empty matrix elements are 0). Such a representation of a moving filter is called a *circulant* matrix. Periodic boundary conditions were used to wrap around the coefficients, but it's also possible to define special coefficients for the boundaries to avoid that if necessary. I've called the output of the "smooth" filter $x^{(1)}$, and the output of the "nonsmooth" filter $w^{(1)}$. Each component has half as many points as the original series. The former is a lower resolution description of the signal, and the latter contains the fine structure that was lost in the smoothing.

It is convenient if the transformation is orthogonal so that the inverse is just the transpose. Requiring that the matrix times its transpose results in the identity matrix gives two nontrivial equations

$$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1$$

$$c_2 c_0 + c_3 c_1 = 0 \quad . \tag{13.19}$$

We also had two equations for the filter

$$c_3 - c_2 + c_1 - c_0 = 0$$

$$-c_2 + 2c_1 - 3c_0 = 0 \tag{13.20}$$

(written in terms of the c's instead of the b's). This is four equations in four unknowns, which can be solved to find

$$\begin{aligned} c_0 &= \frac{1+\sqrt{3}}{4\sqrt{2}} \qquad c_1 = \frac{3+\sqrt{3}}{4\sqrt{2}} \\ c_2 &= \frac{3-\sqrt{3}}{4\sqrt{2}} \qquad c_3 = \frac{1-\sqrt{3}}{4\sqrt{2}} \quad . \end{aligned} \tag{13.21}$$

Using these coefficients, the transformation can be inverted by using the transpose

$$\begin{bmatrix} c_0 & c_3 & & & & & & & c_2 & c_1 \\ c_1 & -c_2 & & & & & & & c_3 & -c_0 \\ c_2 & c_1 & c_0 & c_3 & & & & & & \\ c_3 & -c_0 & c_1 & -c_2 & & & & & & \\ & & c_2 & c_1 & c_0 & c_3 & & & & \\ & & c_3 & -c_0 & c_1 & -c_2 & & & & \\ & & & & \ddots & & & & & \\ & & & c_2 & c_1 & c_0 & c_3 & & & \\ & & & c_3 & -c_0 & c_1 & -c_2 & & & \\ & & & & & c_2 & c_1 & c_0 & c_3 & \\ & & & & & c_3 & -c_0 & c_1 & -c_2 \end{bmatrix} \begin{pmatrix} x_0^{(1)} \\ w_0^{(1)} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{N/2-1}^{(1)} \\ w_{N/2-1}^{(1)} \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{pmatrix}$$

Let's now multiply the output from the filters by another orthonormal matrix that splits the two types of results:

$$\begin{bmatrix} 1 & 0 & & & & & & & & \\ & 0 & 1 & & & & & & & \\ & & & & & 1 & 0 & & & \\ & & & & & & 0 & 1 & 0 & \\ & & & & \ddots & & & & & \\ 0 & 1 & 0 & & & & & & & \\ & & 0 & 1 & & & & & & \\ & & & & & & & 1 & 0 & \\ & & & & & & & & 0 & 1 \end{bmatrix} \begin{pmatrix} x_0^{(1)} \\ w_0^{(1)} \\ x_1^{(1)} \\ w_1^{(1)} \\ \vdots \\ x_{N/2-2}^{(1)} \\ w_{N/2-2}^{(1)} \\ x_{N/2-1}^{(1)} \\ w_{N/2-1}^{(1)} \end{pmatrix} = \begin{pmatrix} x_0^{(1)} \\ x_1^{(1)} \\ \vdots \\ x_{N/2-2}^{(1)} \\ x_{N/2-1}^{(1)} \\ w_0^{(1)} \\ w_1^{(1)} \\ \vdots \\ w_{N/2-1}^{(1)} \end{pmatrix}$$

The first half of the resulting vector is a smoothed version of the original signal at half the time resolution, and the second half contains the details lost in the smoothing. The original series can be recovered by multiplying by the transposes of the two matrices used. We can now go ahead and do the same sequence of operations on the new x's, to give a version at even lower resolution as well as some more "detail" coefficients. Repeating the filtering and shuffling operations until we're left with just two x values and so can go no further gives the following sequence of coefficient vectors:

$$\begin{array}{ccccccc}
x_0 & x_0^{(1)} & x_0^{(1)} & x_0^{(2)} & x_0^{(2)} & x_0^{(\log_2 N-1)} & x_0^{(\log_2 N-1)} \\
x_1 & w_0^{(1)} & x_1^{(1)} & w_0^{(2)} & x_1^{(2)} & w_0^{(\log_2 N-1)} & x_1^{(\log_2 N-1)} \\
\vdots & x_1^{(1)} & \vdots & x_1^{(2)} & \vdots & x_1^{(\log_2 N-1)} & w_0^{(\log_2 N-1)} \\
\vdots & w_1^{(1)} & \vdots & w_1^{(2)} & x_{N/4-1}^{(2)} & w_1^{(\log_2 N-1)} & w_1^{(\log_2 N-1)} \\
\vdots & \vdots & \vdots & \vdots & w_0^{(2)} & w_0^{(\log_2 N-2)} & w_0^{(\log_2 N-2)} \\
\vdots & \vdots & \vdots & \vdots & w_1^{(2)} & \vdots & \vdots \\
\vdots & \vdots & \vdots & x_{N/4-1}^{(2)} & \vdots & \cdots \quad \vdots & \vdots \\
\vdots & \vdots & x_{N/2-1}^{(1)} & w_{N/4-1}^{(2)} & w_{N/4-1}^{(2)} & \vdots & \vdots \\
\vdots & \vdots & w_0^{(1)} & w_0^{(1)} & w_0^{(1)} & \vdots & \vdots \\
\vdots & \vdots & w_1^{(1)} & w_1^{(1)} & w_1^{(1)} & \vdots & \vdots \\
\vdots & x_{N/2-1}^{(1)} & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_{N-1} & w_{N/2-1}^{(1)} & w_{N/2-1}^{(1)} & w_{N/2-1}^{(1)} & w_{N/2-1}^{(1)} & w_{N/2-1}^{(1)} & w_{N/2-1}^{(1)}
\end{array}$$

This defines the *Discrete Wavelet Transformation* (*DWT*), and the final w's are the wavelet coefficients. They represent structure at many scales as well as at many locations. If any of the wavelet coefficients are small they can be set to zero to approximate the original series with less information, but the beauty of this kind of compression is that it can find important regions in the time-frequency space rather than projecting all of the information onto the frequency axis (as done by an FFT) or the time axis (by impulses).

A sine wave looks like, well, a sine wave. What does a wavelet look like? We can find out by setting one of the wavelet coefficients to 1 and all the others to 0, and then running the inverse wavelet transform back to find the x series that produces it (just as inverting a Fourier transform of an impulse gives a sinusoidal function). Problem 12.3 shows that this results in quite a curious looking function. To understand it, consider that after one pass of the smoothing filter,

$$x_n^{(1)} = c_0 x_{2n} + c_1 x_{2n+1} + c_2 x_{2n+2} + c_3 x_{2n+3} \quad . \tag{13.22}$$

If a function exists that satisfies

$$X_n = c_0 X_{2n} + c_1 X_{2n+1} + c_2 X_{2n+2} + c_3 X_{2n+3} \tag{13.23}$$

then it will be unchanged by the smoothing (this is a *dilation equation*, instead of a difference or differential equation). The associated wavelet function is

$$W_n = c_3 X_{2n} - c_2 X_{2n+1} + c_1 X_{2n+2} - c_0 X_{2n+3} \quad . \tag{13.24}$$

In the limit of many iterations, so that n approaches a continuous variable, these are the basis functions that are invariant under the transformation. Remarkably, in the continuum limit these apparently innocent and certainly useful functions are very complicated, not even differentiable.

We've been looking at fourth-order wavelets; higher orders are similarly defined. For each two additional coefficients used it's possible to go to one higher derivative of the function that can be matched. Beyond order 6, the coefficients must be found numerically. Our wavelets also have had *compact support* (they are zero everywhere except for where they are defined); this is convenient numerically but can be relaxed in order to get other benefits such as the analytical form and simple spectrum of the *harmonic wavelets* [Newland, 1994].

Just as a high-dimensional Fourier transform can be done by transforming each axis in turn, wavelets can be extended to higher dimensions by transforming each axis separately [Press *et al*., 2007]. This restricts the wavelets to the axes of the space; it is also possible to define more general multi-dimensional wavelets. The state of the art in wavelets has advanced rapidly since their introduction; see for example [Chui *et al*., 1994]. Beyond wavelets there are other time-frequency transforms, such as *Wigner functions* [Hlawatsch & Boudreaux-Bartels, 1992], which first arose as a probabilistic representation of quantum mechanics for studying semi-classical systems [Balazs & Jennings, 1984].

13.4 PRINCIPAL COMPONENTS

Wavelets were constructed based on the assumption that time and frequency are the interesting axes against which a signal can be viewed. This certainly need not be true, and doesn't even apply to a set of measurements that have no particular temporal or spatial ordering. Rather than designing one transform to apply to all data we might hope to do better by customizing a transform to provide the best representation for a given data set (where "best" of course will reflect some combination of what we hope to achieve and what we know how to accomplish).

Let's once again let $\vec{x}$ be a measurement vector, and $\vec{y} = \mathbf{M} \cdot \vec{x}$ be a transformation to a new set of variables with more desirable properties. The *covariance matrix* of $\vec{y}$ is defined by

$$\mathbf{C}_y \equiv \langle (\vec{y} - \langle \vec{y} \rangle) \cdot (\vec{y} - \langle \vec{y} \rangle)^T \rangle \quad , \tag{13.25}$$

where the *outer product* of two column vectors $\vec{A}$ and $\vec{B}$ is

$$(\vec{A} \cdot \vec{B}^T)_{ij} = A_i B_j \quad , \tag{13.26}$$

and the average is taken over an ensemble of measurements. A reasonable definition of "best" is to ask that the covariance matrix of $\vec{y}$ be diagonal, so that each of its elements is uncorrelated.

To find the required transformation, the covariance matrix of $\vec{y}$ can be related to that of $\vec{x}$:

$$\begin{aligned} \mathbf{C}_y &= \langle (\vec{y} - \langle \vec{y} \rangle) \cdot (\vec{y} - \langle \vec{y} \rangle)^T \rangle \\ &= \langle [\mathbf{M} \cdot (\vec{x} - \langle \vec{x} \rangle)] \cdot [\mathbf{M} \cdot (\vec{x} - \langle \vec{x} \rangle)]^T \rangle \end{aligned}$$

$$
\begin{aligned}
&= \langle [\mathbf{M} \cdot (\vec{x} - \langle \vec{x} \rangle)] \cdot [(\vec{x} - \langle \vec{x} \rangle)^T \cdot \mathbf{M}^T] \rangle \\
&= \mathbf{M} \cdot \langle (\vec{x} - \langle \vec{x} \rangle) \cdot (\vec{x} - \langle \vec{x} \rangle)^T \rangle \cdot \mathbf{M}^T \\
&= \mathbf{M} \cdot \mathbf{C}_x \cdot \mathbf{M}^T \quad . \qquad (13.27)
\end{aligned}
$$

Because $\mathbf{C}_x$ is a real symmetric matrix it's possible to find an orthonormal set of eigenvectors [Golub & Loan, 1996]. Now consider what happens if the columns of $\mathbf{M}^T$ are taken to be these eigenvectors. After multiplication by $\mathbf{C}_x$ each eigenvector is returned multiplied by its corresponding eigenvalue. Then because of the orthonormality, the multiplication of this matrix by $\mathbf{M}$ gives zeros off-diagonal, and returns the values of the eigenvalues on the diagonal. Therefore $\mathbf{C}_y$ is a diagonal matrix as desired. If there are linear correlations among the elements of $\vec{x}$ then some of the eigenvalues will vanish; these components of $\vec{y}$ can be dropped from subsequent analysis. For real data sets the elements might not be exactly equal to zero, but the relative magnitudes of them let the important components be found and the less important ones be ignored. Such *variable subset selection* is frequently the key to successful modeling.

Use of the covariance matrix of a set of measurements to find a transformation to new variables that are uncorrelated is called *Principal Components Analysis* (*PCA*). It is such a useful idea that it led to many other related three-letter acronyms (*TLAs*). One is the *Karhunen–Loéve Transform* (*KLT*) [Fukunaga, 1990]. Here, a measurement vector $\vec{y}$ (such as a time series, or the values of the pixels in an image) is expanded in a sum over orthonormal basis vectors $\vec{\varphi}_i$ with expansion coefficients x_i,

$$\vec{y} = \sum_i x_i \vec{\varphi}_i \quad . \qquad (13.28)$$

Given an ensemble of measurements of $\vec{y}$, the goal is to choose a set of $\vec{\varphi}_i$ that make the x_i's as independent as possible. Defining $\mathbf{M}$ to be a matrix that has the $\vec{\varphi}_i$ as column vectors, the expansion of $\vec{y}$ can be written as $\vec{y} = \mathbf{M} \cdot \vec{x}$, where $\vec{x}$ is a column vector of the expansion coefficients. We've already seen that the covariance matrices of $\vec{y}$ and $\vec{x}$ are related by

$$\mathbf{C}_y = \mathbf{M} \cdot \mathbf{C}_x \cdot \mathbf{M}^T \qquad (13.29)$$

or

$$\mathbf{M}^T \cdot \mathbf{C}_y \cdot \mathbf{M} = \mathbf{C}_x \quad . \qquad (13.30)$$

Therefore if we choose the $\vec{\varphi}_i$ to be the eigenvectors of $\mathbf{C}_y$ then $\mathbf{C}_x$ will be diagonal. Since the $\vec{\varphi}_i$ are orthonormal, given a new measurement $\vec{y}$ the expansion coefficients can be found from

$$\vec{y} \cdot \vec{\varphi}_j = \sum_i x_i \vec{\varphi}_i \cdot \vec{\varphi}_j = \sum_i x_i \delta_{ij} = x_j \quad . \qquad (13.31)$$

This provides a convenient way to do *lossy compression* for storage or communications, by using only the significant coefficients to partially reconstruct a data vector from the bases.

13.5 INDEPENDENT COMPONENTS

PCA starts with the covariance matrix of all of the original variables and then throws out the insignificant components. *Factor Analysis* directly seeks a smaller set of variables that can explain the covariance structure of the observations [Hair *et al*., 1998]. *Independent Components Analysis* (*ICA*) goes further to search for a transformation that makes the new variables independent ($p(y_i, y_j) = p(y_i)p(y_j)$) rather than just uncorrelated [Comon, 1994; Bell & Sejnowski, 1995; Hyvärinen *et al*., 2004].

In the *blind source separation* problem, unknown sources $\vec{s}$ are mixed in observations $\vec{x}$ by an unknown matrix $\vec{x} = \mathbf{A} \cdot \vec{s}$. If weights $\mathbf{W} = \mathbf{A}^{-1}$ could be found then the sources could be separated by $\vec{s} = \mathbf{W} \cdot \vec{x}$, but how can this be done without information about either $\mathbf{A}$ or $\vec{s}$? The surprising answer is that this can be possible if the signals are interesting. In Section 6.1.2 we saw that a sum of random variables approaces a Gaussian distribution for almost any distribution of the variables. Conversely, as long as the distributions do not start out as Gaussians, then the departure from Gaussianity can be used as a signature for separating them. This is the basis for ICA.

Gaussianity can be tested with the *kurtosis* (4th cumulant), but because that raises variables to the fourth power it's sensitive to outliers. Entropy is another test (Problem 11.3 showed that the entropy of a Gaussian is maximal for continuous distributions with a given variance), but that requires an estimate of the probability distribution function. A simpler approximation is to use a *contrast function* [Hyvarinen, 1999]. For a vector of weights $\vec{w}$ corresponding to one component of $\vec{s}$ (i.e., one row of $\mathbf{W}$), this approach seeks to maximize the expected value of a function relative to its value for a Gaussian, $\max\langle f(\vec{w} \cdot \vec{x})\rangle$, where the observations are used to evaluate the expectation. A convenient example is $f(\vec{x}) = \log\cosh(\vec{x})$ [Hyvärinen & Oja, 2000].

Because ICA can't determine absolute scale factors in $\mathbf{W}$, it's convential in ICA to start with PCA, so that the data is zero mean and has a diagonal covariance matrix with unit variances (called *sphering* the data). The weight vectors are then taken to have $|\vec{w}|^2 = 1$ to preserve that undetermined norm. To find them we want to make extremal

$$F = \langle f(\vec{w} \cdot \vec{x})\rangle - \lambda \vec{w} \cdot \vec{w} \tag{13.32}$$

with the Lagrange multiplier for the normalization. Taking the gradient,

$$\frac{\partial F}{\partial \vec{w}} = \langle \vec{x} f'(\vec{w} \cdot \vec{x})\rangle - \lambda \vec{w} \tag{13.33}$$

The Jacobian can be approximated by:

$$\begin{aligned} \frac{\partial^2 F}{\partial \vec{w}^2} &= \langle \vec{x}\vec{x}^T f''(\vec{w} \cdot x)\rangle - \lambda \mathbf{I} \\ &\approx \langle \vec{x}\vec{x}^T\rangle \langle f''(\vec{w} \cdot x)\rangle - \lambda \mathbf{I} \\ &\approx \left(\langle f''(\vec{w} \cdot x)\rangle - \lambda\right) \mathbf{I} \end{aligned} \tag{13.34}$$

because of the initial diagonalization of the covariance matrix. A Newton root-finding step is performed by subtracting the gradient over the Jacobian:

$$\vec{w} \leftarrow \vec{w} - \left(\langle \vec{x} f'(\vec{w} \cdot \vec{x})\rangle - \lambda \vec{w}\right) / \left(\langle f''(\vec{w} \cdot \vec{x})\rangle - \lambda\right) \tag{13.35}$$

Multiplying both sides by $\left(\lambda - \langle f''(\vec{w} \cdot \vec{x})\rangle\right)$,

$$\vec{w}\left(\lambda - \langle f''(\vec{w} \cdot \vec{x})\rangle\right) \leftarrow \vec{w}\left(\lambda - \langle f''(\vec{w} \cdot \vec{x})\rangle\right) + \left(\langle \vec{x} f'(\vec{w} \cdot \vec{x})\rangle - \lambda \vec{w}\right) \tag{13.36}$$

Since we'll be normalizing the weights the multiplicative factor can be dropped on the left side, and the right side simplified:

$$\vec{w} \leftarrow \langle \vec{x} f'(\vec{w} \cdot \vec{x})\rangle - \vec{w}\langle f''(\vec{w} \cdot \vec{x})\rangle \tag{13.37}$$

After this step the normalization is preserved with

$$\vec{w} \leftarrow \frac{\vec{w}}{|\vec{w}|} \tag{13.38}$$

(this corresponds to choosing the Lagrange multiplier). Starting with random normalized weights, this iteration will seek the least-Gaussian mixture. Multiple weights can either be found by orthogonalizing serially to span the remaining subspace, or be calculated jointly [Hyvarinen, 1999]. This is a global linear ICA transformation; it can done both nonlinearly and locally [Hyvärinen *et al*., 2004].

In the next chapter we'll see another way to accomplish something similar, with a neural network *autoencoder*.

13.5.1 Compressed Sensing

usually measure, then compress

can instead compress, then measure

sparsity a kind of universal prior

example: total variation (TV)

random sampling vs periodic sampling artifacts

fitting error only is underconstrained

L2 norm: sum of squares

match samples and minimize L2 constraint: pseudo-inverse

L0 norm: number of nonzero elements

match samples and minimize L0 constraint: combinatorial, NP-hard

L1 norm: sum of absolute values

match samples and minimize L1 constraint: compressed sensing

DTMF problem

Fornasier, Massimo, and Rauhut, Holger. (2011). Compressive sensing. Pages 187–228 of: Handbook of Mathematical Methods in Imaging. Springer.

Donoho, David L. (2006). Compressed sensing. Information Theory, IEEE Transactions on, 52(4), 1289–1306.

Candes, Emmanuel J, and Tao, Terence. (2006). Near-optimal signal recovery from random projections: Universal encoding strategies? Information Theory, IEEE Transactions on, 52(12), 5406–5425.

Candes, Emmanuel J, Romberg, Justin, and Tao, Terence. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. Information Theory, IEEE Transactions on, 52(2), 489–509.

Candes, Emmanuel J, Romberg, Justin K, and Tao, Terence. (2006). Stable signal

recovery from incomplete and inaccurate measurements. Communications on pure and applied mathematics, 59(8), 1207–1223.

Kim, Seung-Jean, Koh, Kwangmoo, Lustig, Michael, and Boyd, Stephen. (2007). An efficient method for compressed sensing. Pages III–117 of: Image Processing, 2007. ICIP 2007. IEEE International Conference on, vol. 3. IEEE.

Yang, Allen Y, Sastry, Shankar S, Ganesh, Arvind, and Ma, Yi. (2010). Fast l1-minimization algorithms and an application in robust face recognition: A review. Pages 1849–1852 of: Image Processing (ICIP), 2010 17th IEEE International Conference on. IEEE.

Wakin, Michael, Stephen Becker, Eric Nakamura, Michael Grant, Emilio Sovero, Daniel Ching, Juhwan Yoo, Justin Romberg, Azita Emami-Neyestanak, and Emmanuel Candes. "A nonuniform sampler for wideband spectrally-sparse environments." IEEE Journal on Emerging and Selected Topics in Circuits and Systems 2, no. 3 (2012): 516-529.

Yoo, Juhwan, Stephen Becker, Matthew Loh, Manuel Monge, Emmanuel Candes, and Azita Emami-Neyestanak. "A 100MHz–2GHz 12.5 x sub-Nyquist rate receiver in 90nm CMOS." In 2012 IEEE Radio Frequency Integrated Circuits Symposium, pp. 31-34. IEEE, 2012.

Tropp, Joel A., Jason N. Laska, Marco F. Duarte, Justin K. Romberg, and Richard G. Baraniuk. "Beyond Nyquist: Efficient sampling of sparse bandlimited signals." arXiv preprint arXiv:0902.0026 (2009).

13.6 SELECTED REFERENCES

[Golub & Loan, 1996] Golub, Gene H., & Loan, Charles F. Van. (1996). *Matrix Computations*. 3rd edn. Baltimore, MD: Johns Hopkins University Press.

Everything you always wanted to know about transformations with matrices.

[Fukunaga, 1990] Fukunaga, Keinosuke (1990). *Introduction to Statistical Pattern Recognition*. 2nd edn. Boston, MA: Academic Press.

Much of the effort in pattern recognition goes into finding good representations.

13.7 PROBLEMS

(12.1) Prove that the DFT is unitary.

(12.2) Calculate the inverse wavelet transform, using Daubechies fourth-order coefficients, of a vector of length 2^{12}, with a 1 in the 5th and 30th places and zeros elsewhere.

(12.3) Consider a measurement of a three-component vector $\vec{x}$, with x_1 and x_2 being drawn independently from a Gaussian distribution with zero mean and unit variance, and $x_3 = x_1 + x_2$.

(*a*) Analytically calculate the covariance matrix of $\vec{x}$.

(*b*) What are the eigenvalues?

(*c*) Numerically verify these results by drawing a data set from the distribution and computing the covariance matrix and eigenvalues.

(*d*) Numerically find the eigenvectors of the covariance matrix, and use them to construct a transformation to a new set of variables $\vec{y}$ that have a diagonal covariance matrix with no zero eigenvalues. Verify this on the data set.

(12.4) Generate pairs of uniform random variables $\{s_1, s_2\}$ with each component contained in $[0, 1]$.

(*a*) Plot these data.

(*b*) Mix them ($\vec{x} = \mathbf{A} \cdot \vec{s}$) with a square matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$ and plot.

(*c*) Make $\vec{x}$ zero mean, diagonalize with unit variance, and plot.

(*d*) Find the independent components of $\vec{x}$ with the log cosh contrast function, and plot.

(12.5) (*a*) Generate and plot a time series $\{t_j\}$ for the sum of two sine waves at 697 and 1209 Hz (the *DTMF* tone for the number 1 key), sampling it at 10,000 samples per second for 0.25 second ($N = 2500$ points).

(*b*) Calculate and plot the *Discrete Cosine Transform (DCT)* coefficients $\{f_i\}$ for these data, defined by their multiplication by the matrix $f_i = \sum_{j=0}^{N-1} D_{ij} t_j$, where

$$D_{ij} = \begin{cases} \sqrt{\frac{1}{N}} & (i = 0) \\ \sqrt{\frac{2}{N}} \cos\left(\frac{\pi(2j+1)i}{2N}\right) & (1 \leq i \leq N-1) \end{cases} \tag{13.39}$$

(*c*) Plot the inverse transform of the $\{f_i\}$ by multiplying them by the inverse of the DCT matrix (which is equal to its transpose) and verify that it matches the time series.

(*d*) Randomly sample and plot a subset $\{t'_k\}$ of 5% of the $\{t_j\}$ ($M = 125$ points).

(*e*) Starting with a random guess for the DCT coefficients $\{f'_i\}$, use gradient descent to minimize the error at the sample points

$$\min_{\{f'_i\}} \sum_{k=0}^{M-1} \left(t'_k - \sum_{i=0}^{N-1} D_{ik} f'_i \right)^2 \tag{13.40}$$

and plot the resulting estimated coefficients.

(*f*) The preceding minimization is under-constrained; it becames well-posed if a norm of the DCT coefficients is minized subject to a constraint of agreeing with the sampled points. One of the simplest (but not best, see Chapter 17) ways to do this is by adding a *penalty term* to the minimization. Repeat the gradient descent minimization using the L2 norm:

$$\min_{\{f'_i\}} \sum_{k=0}^{M-1} \left(t'_k - \sum_{i=0}^{N-1} D_{ik} f'_i \right)^2 + \sum_{i=0}^{N-1} f'^2_i \tag{13.41}$$

and plot the resulting estimated coefficients.

(*g*) Repeat the gradient descent minimization using the L1 norm:

$$\min_{\{f'_i\}} \sum_{k=0}^{M-1} \left(t'_k - \sum_{i=0}^{N-1} D_{ik} f'_i \right)^2 + \sum_{i=0}^{N-1} |f'_i| \tag{13.42}$$

and plot the resulting estimated coefficients, compare to the L2 norm estimate, and compare M to the Nyquist sampling limit of twice the highest frequency.