Empirical Orthogonal Functions

Fourier transforms let us project a data record into a bunch of orthogonal modes, and two-dimensional Fourier transforms let us project a data array into a two-dimensional array of orthogonal Fourier modes. Fourier space modes are convenient when we find that a few modes explain a lot of the variability. But obviously there are cases where variability isn’t really sinusoidal, and Fourier modes do a poor job of representing the overall variability. However, clearly we can define other types of orthogonal basis functions that might represent our data.

Our goal here will be to find a set of functions that represent as much of the variability in our data record as possible using as small a set of orthogonal functions as possible.

Fourier modes are orthogonal, but they aren’t necessarily compact, unless we have a signal that can be expressed as a single frequency and wavenumber. In addition, for two dimensional arrays, Fourier modes are really good are representing propagating signals but not as natural for components of the data that are stationary in time or space and not oscillatory with a regular cycle.

As an example, we’ll start with a small data set, consisting of minimum temperature records from San Miguel and Alpine (in Figure 1). We’d like to find a single way to represent as much of the variability in the two records as possible. What’s the maximum fraction of the common variability that we can capture using a single time series?

Formally, we’ll set up the problem this way. Given a two-dimensional data set \( Y \) which is an \( N \times M \) matrix, where \( N \) might be the number of elements in our time series and \( M \) the number of geographic locations where we have observations. Thus for the Alpine and San Miguel data, \( N \) is 366, and \( M \) is 2. We’d like to represent as much of \( Y \) as possible as the product of a \( N \times 1 \) column vector that we’ll call \( g \) and a \( 1 \times M \) row vector that we’ll call \( h \). We’ll also require that \( h \) is normalized to one. Thus we want to minimize

\[
\mathcal{L} = \sum_{i=1}^{N} \sum_{j=1}^{M} (Y_{i,j} - g_i h_j)^2 + \Lambda \left( \sum_{j=1}^{M} h_j^2 - 1 \right). \tag{1}
\]

Here \( \Lambda \) is a Lagrange multiplier that scales the relative importance of these two constraints. For the moment we won’t worry about the value of \( \Lambda \).

Since we want to adjust \( g \) and \( h \) in order to minimize \( \mathcal{L} \), we’ll take the derivative of \( \mathcal{L} \) in terms of both

\[
g \rightarrow \frac{\partial \mathcal{L}}{\partial g_i} = 2 \sum_{j=1}^{M} (Y_{i,j} - g_i h_j) h_j = 0
\]

\[
h \rightarrow \frac{\partial \mathcal{L}}{\partial h_j} = 2 \sum_{i=1}^{N} (Y_{i,j} - g_i h_j) g_i = 0
\]

Solving these equations involves setting up a matrix equation, which we’ll do next.
of them:
\[
\frac{\partial L}{\partial g_i} = -2 \sum_{j=1}^{M} (Y_{i,j} - g_i h_j) h_j = 0 \\
\frac{\partial L}{\partial h_j} = -2 \sum_{i=1}^{N} (Y_{i,j} - g_i h_j) g_i + 2\lambda h_j = 0.
\]  

(2)  
(3)  

Since \( hh^T = 1 \), the first constraint is met when \( \sum_{j=1}^{M} (Y_{i,j} h_j) = g_i \) or in matrix form \( G = Y h^T \). The second constraint is met when \( h_j (\Lambda + \sum_{i=1}^{N} h_i^2) = \sum_{i=1}^{N} Y_{i,j} g_i \). Since \( \Lambda + \sum_{i=1}^{N} h_i^2 \) is a constant for any given \( i \), we’ll call it \( \lambda \). Thus the constraint is equivalent to \( \lambda h_j = \sum_{i=1}^{N} g_i Y_{i,j} \) or in matrix form \( \lambda h = g^T Y \).

Now we can substitute in our solution for \( g \) from before, to find \( \lambda h = h Y^T Y \), or equivalently, \( \lambda h^T = Y^T Y h^T \). In linear algebra, this is an eigenvector equation for the matrix \( Y^T Y \), and we can solve it for \( \lambda \) and \( h \) simultaneously. Conveniently, this means that we don’t need to worry about setting \( \Lambda \) to determine the relative weights of the two constraints in the cost function—the eigenvector solution will take care of this for us.

We can also multiply \( g = Y h^T \) by \( \lambda \) to get \( \lambda g = \lambda Y h^T = Y \lambda h^T = Y Y^T g \) which is an eigenvector equation for the matrix \( YY^T \). The vectors \( g \) and \( h \) then define the empirical modes that best represent our system.

The matrices \( YY^T \) and \( Y^T Y \) are covariance matrices for our data. One represents time averages of covariances between data at various spatial locations. The other is spatial averages of data at various points in time. They’re conceptually similar; their definitions depend only on how we’ve arranged the data in the matrix \( Y \).

Now that we’ve defined our empirical modes, how best can we solve for them? Here are three methods:

- **Find the covariance matrix formally, and then find its eigenvalues and eigenvectors.** If the matrix \( Y \) has no gaps, then its covariance matrix is just \( YY^T \) or \( Y^T Y \). In Matlab you can solve the eigenvalue equation with ‘\( \text{[V,lambda}=\text{eig}(Y^*Y)\)’. \( V \) gives us the empirical modes \( h \) in the space domain. To find the empirical modes \( g \) in the time domain, we could solve for \( YY^T \). It’s probably more efficient to project our data onto the time domain modes and find the appropriate amplitudes, keeping in mind that \( g = Y h^T \).

- **We can also find eigenvalues and eigenvectors using the singular value decomposition of \( YY^T \) or \( Y^T Y \).** We know that any matrix can be represented with its singular value decomposition \( USV^T \). Since \( YY^T \) is symmetric, its singular value decomposition will be symmetric as well: \( YY^T = V S V^T \), where \( S \) represents the eigenvalues and \( V \) the eigenvectors or empirical orthogonal modes. The nice thing about the singular value decomposition is that we’ve already thought about it for least-squares fitting, and we know that the vectors \( V \) are defined to be orthonormal and that the singular values are sorted by size. In this case we can define \( h = V \).

- **If we’re considering the SVD, then why not skip the covariance matrix.** We know that any matrix \( Y \) can be decomposed into its singular value decomposition. \( Y = USV^T \). That means that \( YY^T = USV^T V S V^T = US^2 U^T \) and \( Y^T Y = V S U^T U S V^T = V S^2 V^T \). Either way \( U \) and \( V \) for matrix \( Y \) are the same singular values that we’d find by computing the SVD of \( YY^T \) or \( Y^T Y \). The singular values \( S \) are a measure of amplitude, and we can interpret them as telling us how much variance is tied up in each mode. \( S^2 \) in the SVD of \( Y \) are equivalent to \( S \) computed above from \( YY^T \). Using the SVD is far and away the easiest strategy because we haven’t had to do anything about determining the covariance matrix of our data, and we’ve managed to represent it with both spatial and temporal modes.

Although people often distinguish between temporal and spatial modes by giving one of them an amplitude that is not one, the SVD format suggests that there’s really no difference between temporal and spatial
modes. Both are automatically normalized to have length 1. The singular values tell us how much variability is associated with each mode, and it often seems more sensible to keep the singular values separate, rather than multiplying them by one or the other of the modes. Thus, instead of saying that $h^T = V_i$ and $g = S_{j,j}U_j$, we can interpret $U$ and $V$ as spatial and temporal modes respectively, and the diagonal of $S$ as corresponding amplitudes.

Originally we said that our leading empirical mode would need to capture as much of the variability in $Y$ as possible as a product of $gh$. This corresponds to $U_1 S_{1,1} V_1^T$.

What if we want to capture a little more variability? Now we want to know how to represent as much of the variability as possible in $Y - U_1 S_{1,1} V_1^T$. In this case, we’ll take the second mode, $U_2 S_{2,2} V_2^T$. It is orthogonal to the first mode, so doesn’t duplicate any of the information in the first mode, and has the next largest singular value, so it captures the next largest fraction of variability in $Y$.

The temporal mode EOFs for the Alpine and San Miguel example are shown in Figure 2. These have been computed using the following:

```matlab
A=[alpine_min sanmiguel_min];
[u,s,v]=svd(A-ones(366,1)*mean(A));
```

Removing the mean temperatures from each record is important. If we didn’t do that, then the mode 1 EOF would tell us mostly about the mean temperature. The corresponding spatial modes are $V_1 = [0.8186 0.5744]$ and $V_2 = [-0.5744 0.8186]$. These tell us that mode 1 is weighted more heavily towards Alpine (the column 1 data set) and less heavily to San Miguel. The corresponding singular values are $s = [199.3673 45.8250]$.

We can determine the fraction of the overall variance represented by each empirical orthogonal mode. If the singular values are $s_i$, then the fraction of total variance explained by empirical mode $i$ is $s_i^2 / \sum_{j=1}^{N} s_j^2$. In the Alpine and San Miguel example, we find:

```matlab
diag(s.^2)/sum(diag(s.^2))
```

that mode 1 explains 95% of the variance in the data and mode 2 explains the remaining 5%. In other words, if we compute the variance of the first mode reconstruction of the data, and the variance of the data:

```matlab
a=sum(sum((u(:,1)*s(1,1)*v(:,1)').^2));
b=sum(sum((A-ones(366,1)*mean(A)).^2));
a/b
```
EOF Examples

Empirical Orthogonal Functions go by a variety of names. Often you’ll hear “principal component analysis”. (Confusingly, a related concept, called “canonical correlation” that relates two different data records, is also sometimes called “singular value decomposition”.) They appear all over climate research.

The Pacific Decadal Oscillation (PDO) is one example of an EOF. It’s the first mode of North Pacific monthly sea surface temperature variability (poleward of 20°N) from 1900 to 1993. Figure 3 shows the spatial mode, and Figure 4 shows the temporal mode. These data are really in three dimensions—time, latitude, and longitude, so you might wonder how to compute an EOF from a three dimensional array. The answer turns out to be simple: EOFs don’t care about the spatial organization of our information, so we can arbitrarily rearrange the spatial points into a one-dimensional vector at each point in time. That gives us a two-dimensional array. We compute EOFs, and then we rearrange the spatial mode EOF into a grid again. The Pacific Decadal Oscillation is thought to be important because it modulates the El Niño patterns in the Pacific Ocean. When the PDO phase is high, we might have a different frequency of El Niño events than when the phase is low.