Seeking Structure in Data

Contents

1.1 Linear Algebra Review

1.1.1 Matrix notation and multiplication

A matrix $\mathbf{A}$ with M rows and N columns (dimensions $M \times N$) is ordered in the following way:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

The notation a_{ij} will be used to denote the i^{th} row and the j^{th} column, where $1 \leqslant i \leqslant M$ and $1 \leqslant j \leqslant N$.

1.1.2 Transpose of a matrix

The transpose of a matrix $\mathbf{X}$ is written as $\mathbf{X}^T$, and is calculated by flipping the rows and columns of a the matrix. So, if the dimensions of $\mathbf{X}$ are $M \times N$, then the dimensions of $\mathbf{X}^T$ are $N \times M$.

Written in index notation

$$\mathbf{X} = \mathbf{Y}^T \tag{1}$$

Then,

$$\mathbf{X}_{ij} = \mathbf{Y}_{ji} \tag{2}$$

1.1.3 Inner product of two vectors

All vectors are matrices with their second dimension being 1. (Note that all matrices are not vectors however).

If $\mathbf{X}$ and $\mathbf{Y}$ are two vectors of length $M \times 1$, then their *inner product* is

$$\mathbf{X}^T\mathbf{Y} = \sum_{i=1}^{M} x_i y_i = \text{scalar} \tag{3}$$

$$[1 \times M] \cdot [M \times 1] = [1 \times 1] \tag{4}$$

If $\mathbf{X}$ and $\mathbf{Y}$ are standardized, then $1/\mathsf{M}$ times their inner product is the regression coefficient/correlation coefficient! Note that this is only true if they have mean of zero and standard deviation of 1. If the two vectors have mean zero only, then their inner product is the *covariance* of the two vectors.

1.1.4 Multiplying matrices

If $\mathbf{A}$ is $\mathsf{M} \times \mathsf{N}$ and $\mathbf{B}$ is $\mathsf{N} \times \mathsf{P}$, then, their product can be written as

$$\mathbf{AB} = \mathbf{C} \tag{5}$$

$$[\mathsf{M} \times \mathsf{N}] \cdot [\mathsf{N} \times \mathsf{P}] = [\mathsf{M} \times \mathsf{P}] \tag{6}$$

Note that unlike with regular multiplication, one cannot re-order matrix multiplication. The number of columns of the first matrix *must* be equal to the number of rows of the second matrix.

To multiply matrices together, you multiply the row of the first matrix with the column of the second matrix, sum the values, and this is the value that goes in the first row and first column of the solution (this should be review, but let's do an example anyway):

$$\begin{bmatrix} 1 & 2 & 3 \\ -1 & -1 & -3 \end{bmatrix} \times \begin{bmatrix} 2 & 3 \\ 4 & 2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 10 \\ -3 & -8 \end{bmatrix}$$

$$[2 \times 3] \times [3 \times 2] = [2 \times 2]$$

1.1.5 Covariance and Correlation matrices

Say you have a matrix $\mathbf{X}$ of dimensions $\mathsf{M} \times \mathsf{N}$. If the columns of $\mathbf{X}$ have mean zero, then

$$\mathbf{C} = \frac{1}{\mathsf{M}} \mathbf{X}^{\mathsf{T}} \mathbf{X} \tag{7}$$

where $\mathbf{C}$ is the *covariance matrix* of dimensions $\mathsf{N} \times \mathsf{N}$.

If the columns of $\mathbf{X}$ are standardized, then

$$\mathbf{C} = \frac{1}{\mathsf{M}} \mathbf{X}^{\mathsf{T}} \mathbf{X} \tag{8}$$

is actually the *correlation matrix*.

1.1.6 Inverse

The inverse of a square matrix is an incredibly important and useful concept. The inverse of square matrix $\mathbf{X}$ is written as $\mathbf{X}^{-1}$. In matrix notation, this does not mean raise the matrix

to the -1 power, but rather, it denotes the inverse specifically. The inverse has a very special property such that

$$\mathbf{X}\mathbf{X}^{-1} = \mathbf{X}^{-1}\mathbf{X} = \mathbf{I} \tag{9}$$

$\mathbf{I}$ is the *identity matrix*, which is a square with 1's along the diagonal and zeros elsewhere.

Important: only square matrices have inverses! There are things called "right inverses" and "left inverses" for non-square matrices, but we won't discuss them here.

In algebra, if you have a variable multiplied on side of an equation, you use division to get rid of it. With matrices, you use the matrix inverse!

1.1.7 Types of matrices

- $\mathbf{X}$ is diagonal if: $\mathbf{X}$ is square, with values along the diagonal and zeros everywhere else
- $\mathbf{X}$ is *symmetric* if: $\mathbf{X}^{\mathsf{T}} = \mathbf{X}$
- $\mathbf{X}$ is *orthonormal* if: $\mathbf{X}^{\mathsf{T}} = \mathbf{X}^{-1}$ or/and $\mathbf{X}^{\mathsf{T}}\mathbf{X} = \mathbf{X}\mathbf{X}^{\mathsf{T}} = \mathbf{I}$

i.e. if all of the off diagonal elements of $\mathbf{X}^{\mathsf{T}}\mathbf{X}$ are zero, then $\mathbf{X}^{\mathsf{T}} = \mathbf{X}^{-1}$, and all the columns in $\mathbf{X}$ are linearly independent (not correlated with one another).

1.1.8 Vector spaces

- $\mathbf{R}^0$: a single point (e.g. 0)
- $\mathbf{R}^1$: the real line
- $\mathbf{R}^2$: where all 2-element vectors exist
- $\mathbf{R}^3$: 3-D space

Vector spaces are spanned by a set of *basis vectors*. The basis for $\mathbf{R}^{\mathsf{N}}$ is a set of N vectors that through linear combination can describe any vector in $\mathbf{R}^{\mathsf{N}}$.

While there are an infinite number of basis vectors for any R^{N} ($\mathsf{N} > 0$), the typical vectors chosen are the unit vectors. Let's work through some examples.

For $\mathbf{R}^2$, there are 2 basis vectors, each of length 2×1. The unit vectors are,

$$\mathbf{e}_1 = (1, 0) \tag{10}$$
$$\mathbf{e}_2 = (0, 1) \tag{11}$$
$$\tag{12}$$

Any point in $\mathbf{R}^2$ can be written as a linear combination of $\mathbf{e}_1$ and $\mathbf{e}_2$. For example, the point (3,2) is actually written as

$$(3, 2) = 3\mathbf{e}_1 + 2\mathbf{e}_2 \tag{13}$$

Try it, and you will see you can move anywhere in $\mathbf{R}^2$ space!

Here's a question, do the following form a basis for $\mathbf{R}^2$?

$$\mathbf{e}_1 = (1, 0) \tag{14}$$
$$\mathbf{e}_2 = (2, 0) \tag{15}$$
$$\tag{16}$$

The answer is "no", since any point with a non-zero second index cannot be obtained. In this case, the space (0,1) and all linear combinations of this vector is called the *null space* since it cannot be reached by the vectors listed $(\mathbf{e}_1, \mathbf{e}_2)$. More on null spaces later.

How about for $\mathbf{R}^3$? The basis vectors in this case are 3 vectors each of length 3×1

$$\mathbf{e}_1 = (1, 0, 0) \tag{17}$$
$$\mathbf{e}_2 = (0, 1, 0) \tag{18}$$
$$\mathbf{e}_3 = (0, 0, 1) \tag{19}$$

By definition, basis vectors must be *linearly independent* - that is, they all provide unique information that the others do not.

To check if your vectors form a basis, you ask if there is an $\mathbf{x}$ such that

$$\mathbf{A}\mathbf{x} = 0 \tag{20}$$

where $\mathbf{A}$ is a matrix made of the vectors you are testing. If there is a solution to this equation (there is a unique $\mathbf{x}$), then the vectors do not span the entire space and/or are not all independent. If such a $\mathbf{x}$ exists then there is a non-trivial null-space of $\mathbf{A}$.

Instead of solving the above equation for specific $\mathbf{x}$'s, we really just want to know if a solution exists or not. Turns out, there is a shortcut. If

$$\det(\mathbf{A}) = |\mathbf{A}| = 0 \tag{21}$$

then there is a non-trivial solution $\mathbf{x}$ to

$$\mathbf{A}\mathbf{x} = 0 \tag{22}$$

Determinants can be tricky to calculate for large vectors, but for 2×2 matrices, they are easy:

$$\det(\mathbf{A}) = |\mathbf{A}| = a_{11}a_{22} - a_{12}a_{21} \tag{23}$$

If the matrix $\mathbf{A}$ is square, and there is a non-trivial solution to

$$\mathbf{A}\mathbf{x} = 0 \tag{24}$$

then $\mathbf{A}$ is termed *singular* or *degenerate* and does not have an inverse.

1.1.9 Rank of a matrix

The rank $\mathtt{r}$ of a matrix is the number of linearly independent columns. Returning to our previous example, if $\mathbf{A}$ holds the basis vectors for $\mathbf{R}^2$, then

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

We can check that the columns of $\mathbf{A}$ are linearly independent by confirming that

$$\det \mathbf{A} = 1 \neq 0 \tag{25}$$

Thus, the rank of $\mathbf{A}$ is 2. However, if

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}$$

then

$$\det \mathbf{A} = 0 - 0 = 0 \tag{26}$$

and we see the columns are not linearly independent. We could also see this by just looking at the matrix and seeing that 2 times the first column gives the second column. However, this is tricky to do with larger matrices. Anyway, in this instance the rank $\mathtt{r}$ of $\mathbf{A}$ is 1, even though $\mathbf{A}$ is 2×2.

The rank $\mathtt{r}$ is always less than or equal to the smallest dimension of a particular matrix, i.e.

$$\mathtt{r} \leqslant \mathtt{min}(\mathsf{M}, \mathsf{N}) \tag{27}$$

If the matrix $\mathbf{A}$ is square, say dimensions $\mathsf{N} \times \mathsf{N}$, and if $\mathtt{r} < \mathsf{N}$, then it tells us there is a nontrivial null-space for $\mathbf{A}$. The directions in $\mathbf{R}^{\mathsf{N}}$ not spanned by the columns of $\mathbf{A}$ define the *null space*.

If the columns of $\mathbf{A}$ are linearly independent, A is *full rank*.

If the columns of $\mathbf{A}$ are not linearly independent, A is *rank deficient*.

1.2 Eigenvalues and eigenvectors

Recall that the covariance matrix (also called the *dispersion matrix*) is calculated as

$$\mathbf{C} = \frac{1}{\mathsf{M}} \mathbf{X}^{\mathsf{T}} \mathbf{X} \tag{28}$$

$$\text{or} \tag{29}$$

$$\mathbf{C} = \frac{1}{\mathsf{N}} \mathbf{X} \mathbf{X}^{\mathsf{T}} \tag{30}$$

assuming the mean of the columns of $\mathbf{X}$ are zero and $\mathbf{X}$ is $\mathsf{N} \times \mathsf{M}$. Note that $\mathbf{C}$ will always be a square matrix that is also symmetric (i.e. $\mathbf{C} = \mathbf{C}^{\mathsf{T}}$).

Principle Component Analysis consists of an eigenanalysis of $\mathbf{C}$. That is, any symmetric matrix $\mathbf{C}$ can be decomposed in the following way:

$$\mathbf{C}\mathbf{e}_i = \lambda_i \mathbf{e}_i \tag{31}$$

$$\mathbf{C}\mathbf{E} = \mathbf{E}\mathbf{\Lambda} \tag{32}$$

where $\mathbf{E}$ is the matrix with eigenvectors $\mathbf{e}_i$ as its columns and $\mathbf{\Lambda}$ is the matrix with eigenvalues λ_i along its diagonal and zeros elsewhere.

The set of eigenvectors and associated eigenvalues represent a *coordinate transformation* into a coordinate space where $\mathbf{C}$ becomes diagonal.

> Because the covariance matrix is diagonal in this new coordinate space, the variations in these new directions are uncorrelated with each other. The eigenvectors define directions in the initial coordinate space along which the maximum possible variance can be explained, and in which variance in one direction is orthogonal to the variance explained by other directions defined by the other eigenvectors. The eigenvalues indicate how much variance is explained by each eigenvector. *[D. Hartmann notes, Chapter 4]*

For example, if

$$\mathbf{A} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

then, the covariance matrix

$$\mathbf{C} = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$$

and one can show that

$$\mathbf{e}_1 = \begin{bmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix} = \sqrt{2}/2 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Note that the norm of $\mathbf{e}_1 = 1$.

We can check this by plugging into the eigenvalue equation above, namely,

$$\mathbf{C}\mathbf{e}_i = \lambda_i \mathbf{e}_i \tag{33}$$

$$\mathbf{C}\mathbf{e}_1 = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \begin{bmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix} = 4 \cdot \begin{bmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$$

Note that $\lambda_1 = 4$ for this to be true. It turns out, $\sum \lambda_i$ = total variance of $\mathbf{A}$, that is, the sum of the diagonal of the covariance matrix $\mathbf{C}$ which is 4. In this problem, there is only one non-zero eigenvalue (that is, $\lambda_2 = 0$), so, the first eigenvector explains *all of the variance* of $\mathbf{A}$.

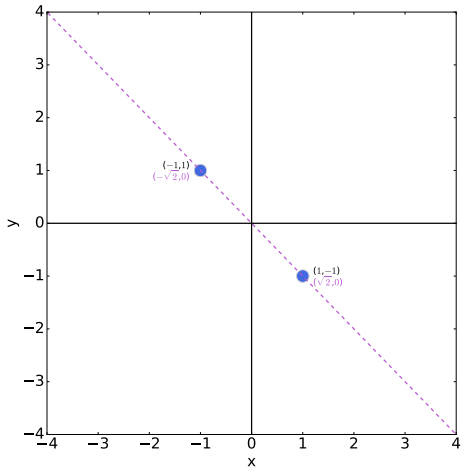

Figure 1: Dots denote the points in $\mathbf{A}$ *with means removed. Black coordinates denote coordinates with the original x-y axis, and purple coordinates denote coordinates along the new axis* $\mathbf{e}_1$.

In a picture, this looks like the following:

Note that

$$\mathbf{e}_1 = \begin{bmatrix} \sqrt{2}/2 \\ -\sqrt{2}/2 \end{bmatrix}$$

is an equally valid eigenvector. Thus, the absolute sign of the entire vector is arbitrary.

Back to the covariance matrix. The eigenvectors are solved for in the following way:

$$\begin{aligned} \mathbf{C}\mathbf{e}_i &= \lambda_i \mathbf{e}_i && (34) \\ (\mathbf{C}\mathbf{e}_i - \lambda_i \mathbf{e}_i) &= 0 && (35) \\ (\mathbf{C} - \lambda_i \mathbf{I})\mathbf{e}_i &= 0 && (36) \\ & && (37) \end{aligned}$$

Thus, the eigenvector $\mathbf{e}_i$ lies in the null-space of $(\mathbf{C} - \lambda_i \mathbf{I})$.

$(\mathbf{C} - \lambda_i \mathbf{I})$ only has a null-space if it is singular, that is, if $\det(\mathbf{C} - \lambda_i \mathbf{I}) = 0$. Solving this is known as an eigenvalue problem and show-up frequency in engineering and science.

The eigenvalues λ_i are positive for a real, symmetric matrix $\mathbf{C}$. In general, if $\mathbf{C}$ is dimensions $M \times M$, then there will be M eigenvalues and M eigenvectors. This is true unless $\mathbf{C}$ is degenerate, in which case, there will be only r eigenvalues and eigenvectors (where r is the rank of the matrix).

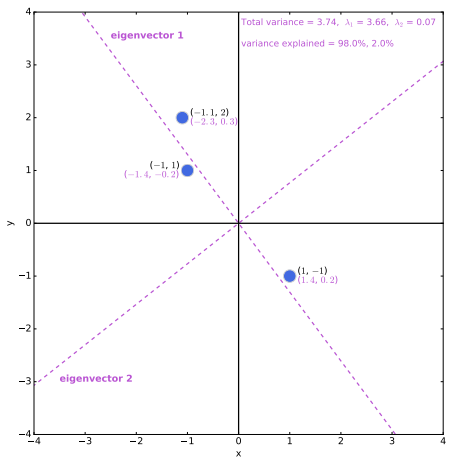

Figure 2: Dots denote another data set with means removed. Black coordinates denote coordinates with the original x-y axis, and purple coordinates denote coordinates along the new axes $\mathbf{e}_1$ *and* $\mathbf{e}_2$*.*

Another example is shown in Figure 2, where the three blue points no longer fall exactly along the same line. If we populate a data matrix $\mathbf{A}$ with these three points, and perform a similar eigenfunction analysis as before, we obtain the two eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$ which can act as a new coordinate axes. While both coordinates for this new axis are non-zero, we see that eigenvector 1 explains most of the variance in the points, while eigenvector 2 accounts for the deviations from eigenvector 1.

Properties of Eigenvectors

- the eigenvectors of a matrix form a linearly independent set
- If $\mathbf{C}$ $(M \times M)$ has full rank, the eigenvectors form a basis for $\mathbf{R}^M$
- If $\mathbf{C}$ is singular with rank $r = k$, the first k eigenvectors span the subspace of $\mathbf{C}$ ($\mathbf{R}^k$), the last $k+1 : M$ eigenvectors span the null-space of $\mathbf{C}$
- If $\mathbf{C}$ is real and symmetric, the eigenvectors are not only linearly independent, they are also orthogonal

Review of terminology

- *linearly independent:* you cannot reach one of the vectors using a linear combination of another (or multiple other) vectors
- *orthogonal:* the angle between the vectors is 90^0, the vector do not share any vector space
- *orthonormal:* the vectors are orthogonal, and have unit length (length $= \sqrt{x^2 + y^2}$).

Example: Python notebook eigenanalysis_2d.py

1.3 EOFs via Eigenanalysis

Example: SHOW INTRO EOF SLIDES 9_EOFINTRO.PDF

- EOF stands for *Empirical Orthogonal Functions*
- PCA stands for *Principle Component Analysis*
- EOF analysis and PCA are practically the same thing
- PCA is more generally used, and just implies splitting a matrix into sets of linearly uncorrelated variables, called principle components
- EOF analysis normally refers to finding the principle components that maximize the variance explained (so a subset of all PC's)

We will discuss *two* ways of calculating EOFs in this class. Keep in mind that both provide identical EOFs at the end. First, we will discuss EOFs via eigenanalysis of the covariance matrix.

1.3.1 Eigenanalysis

Say we have a data matrix $\mathbf{X}$ which is $M \times N$. For now, let the rows denote different times and the columns different spatial locations (although this is not required for the actual application).

The goal is to decompose $\mathbf{X}$ into orthogonal eigenvectors that explain the most variance of $\mathbf{X}$. That is, we want to *maximize the resemblance* of $\mathbf{e}_1$ (dimensions $[N \times 1]$; that is space) to the data, that is, find the $\mathbf{e}_i$ that explains the most variance.

1. the projection of the $\mathbf{e}_1$ onto the data is measured by the inner product of $\mathbf{e}_1$ with $\mathbf{X}$ (this is the "resemblance')
2. the projection is squared to ensure that a positive or negative resemblance are counted the same (as in the case of linear regression where we square the errors)
3. to ensure that the projection (measure of resemblance) is independent of the number of observations, we should divide by M
4. to make the projection (measure of resemblance) independent of the magnitude of $\mathbf{e}_1$, we will normalize it to have unit length, that is, $\mathbf{e}_1^T \mathbf{e}_1 = 1$. Otherwise the projection can be made arbitrarily large by increasing the length of $\mathbf{e}_1$
5. if one follows these rules, the measure of resemblance of $\mathbf{e}$ to the data is:

$$(\mathbf{X}\mathbf{e}_1)^2 \frac{1}{M} = \mathbf{e}_1^T \mathbf{X}^T \mathbf{X} \mathbf{e}_1 \frac{1}{M}. \tag{38}$$

This is equivalent to maximizing

$$\mathbf{e}_1^T \mathbf{C} \mathbf{e}_1 \tag{39}$$

subject to $\mathbf{e}_1^T \mathbf{e}_1 = 1$ and where,

$$\mathbf{C} = \frac{1}{M} \mathbf{X}^T \mathbf{X} \tag{40}$$

If we assume the maximum value of this squared projection is λ_1, then this is the variance explained by vector $\mathbf{e}_1$, that is,

$$\mathbf{e}_1^T \mathbf{C} \mathbf{e}_1 = \lambda_1 \tag{41}$$

This equation is the same form as the eigenvalue problem we discussed previously. That is, this equation becomes

$$\mathbf{C}\mathbf{e}_1 = \mathbf{e}_1 \lambda_1 \tag{42}$$

since the norm (length) of $\mathbf{e}_1$ is one. The full matrix notation (where the dimensions of $\mathbf{C}$, $\mathbf{\Lambda}$ and $\mathbf{E}$ are $N \times N$) is

$$\mathbf{E}^T \mathbf{C} \mathbf{E} = \mathbf{\Lambda} \Rightarrow \mathbf{E}\mathbf{E}^T \mathbf{C} \mathbf{E} = \mathbf{E}\mathbf{\Lambda} \Rightarrow \mathbf{C}\mathbf{E} = \mathbf{E}\mathbf{\Lambda} \tag{43}$$

using $\mathbf{E}^T\mathbf{E} = \mathbf{E}\mathbf{E}^T = \mathbf{I}$.

Thus, $\mathbf{e}_1$ must be an eigenvector of $\mathbf{C}$ with corresponding eigenvalue λ_1! Hence, we can find $\mathbf{e}_1$ by "eigenanalyzing" $\mathbf{C}$ and then choosing the eigenvector that corresponds to the largest eigenvalue.

Notes

- the 1st eigenvector corresponds to the vector that explains the most variance in $\mathbf{X}$ and has the largest eigenvalue
- the 2nd eigenvector corresponds to the vector that explains the second most variance in $\mathbf{X}$ and has the 2nd largest eigenvalue
- The eigenanalysis of the covariance matrix transforms $\mathbf{C}$ into a different coordinate system where the "new" dispersion matrix is diagonal ($\mathbf{\Lambda}$).
- In this new coordinate space, all of the variance is along the diagonal since the different vectors are orthogonal. Thus, the fraction of variance explained by the j^{th} eigenvector is the corresponding eigenvalue λ_j divided by the sum of all of the eigenvalues, that is $\lambda_j / \sum_i \lambda_i$.
- Typically, EOFs are ordered by their corresponding λ, such that EOF 1 explains the largest variance (largest λ) and the last EOF (typically the N^{th} EOF) has the smallest.
- *No other linear combination of k predictors can explain a larger fraction of the variance than the first k EOF/PC's!*

1.3.2 Eigenvectors are orthogonal

Finally, to confirm that different eigenvectors are orthogonal, we suppose we have two eigenvectors $\mathbf{e}_j$ and $\mathbf{e}_k$. Thus, we know that

$$\mathbf{C}\mathbf{e}_j = \mathbf{e}_j\lambda_j \tag{44}$$

and

$$\mathbf{C}\mathbf{e}_k = \mathbf{e}_k\lambda_k \tag{45}$$

Multiplying the j^{th} equation by $\mathbf{e}_k^T$ and taking the transpose, then multiplying the k^{th} by $\mathbf{e}_j^T$ and subtracting the two equations leads to

$$\mathbf{e}_j^T\mathbf{C}^T\mathbf{e}_k - \mathbf{e}_j^T\mathbf{C}\mathbf{e}_k = (\lambda_j - \lambda_k)\mathbf{e}_j^T\mathbf{e}_k \tag{46}$$

Since $\mathbf{C}$ is symmetric, $\mathbf{C} = \mathbf{C}^T$ and so the left-hand side is zero. Thus,

$$\mathbf{e}_j^T\mathbf{e}_k = 0 \tag{47}$$

$$\text{unless} \tag{48}$$

$$\lambda_j = \lambda_k \tag{49}$$

Therefore, the eigenvectors are orthogonal if the eigenvalues are distinct.

1.3.3 Getting back your original data

We wish to maintain all of the original data in the new "coordinate transformed" data set, and thus, we need to know the *loading vectors* to express a particular state in the new EOF-basis.

We define

$$\mathbf{Z} = \mathbf{X}\mathbf{E} \tag{50}$$

Then it follows that

$$\mathbf{X} = \mathbf{Z}\mathbf{E}^T \tag{51}$$

$$\text{since} \tag{52}$$

$$\mathbf{E}\mathbf{E}^T = \mathbf{I} \tag{53}$$

Thus, we can get our original $\mathbf{X}$ data back by multiplying $\mathbf{Z}\mathbf{E}^T$.

$\mathbf{Z}$ are known as the principle components (PC's), and are length M. These values tell you how much a given sample looks like a particular EOF structure.

One reconstructs the original $\mathbf{X}$ data from this new coordinate system by combining the eigenvectors in linear combinations with amplitudes of the PCs. That is, a given observation at time t and location i, written as $\mathbf{x}_i(t)$ can be found using a linear combination of the EOFs multiplied by their PC time series value

$$x_i(t) = \sum_{j=1}^{N} e_{ij}z_j(t) \tag{54}$$

As an example, suppose you have a set of orthogonal and normalized eigenvectors. The first one might looks like this:

$$\mathbf{e}_1 = \begin{bmatrix} e_{11} \\ e_{21} \\ e_{31} \\ e_{41} \\ \dots \end{bmatrix}$$

Next, *project* $\mathbf{e}_1$ onto the original data to get the amplitude of this eigenvector at each time step. This is done in the following way:

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mn} \end{bmatrix} \begin{bmatrix} e_{11} \\ e_{21} \\ e_{31} \\ e_{41} \\ \dots \\ e_{n1} \end{bmatrix} = \begin{bmatrix} z_{11} \\ z_{21} \\ z_{31} \\ z_{41} \\ \dots \\ z_{m1} \end{bmatrix}$$

That is

$$\mathbf{XE} = \mathbf{Z} \tag{55}$$

Note, you do not want to project $\mathbf{e}$ onto the covariance matrix $\mathbf{C}$, but rather, you want to project it onto the original data.

1.3.4 Practical considerations

Recall that the covariance matrix of $\mathbf{X}$ can be computed in one of two ways, either,

$$\mathbf{C} = \frac{1}{M}\mathbf{X}^T\mathbf{X} \rightarrow [N \times N] \tag{56}$$

$$\text{or} \tag{57}$$

$$\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T \rightarrow [M \times M] \tag{58}$$

So far, we have used the following covariance (due to the sampling dimension being M):

$$\mathbf{C} = \frac{1}{M}\mathbf{X}^T\mathbf{X} \rightarrow [N \times N] \tag{59}$$

If you have $M = 20000$ days, and $N = 400$ grid points then, performing eigenanalysis on

$$\mathbf{C} = \frac{1}{M}\mathbf{X}^T\mathbf{X} \rightarrow [N \times N] \tag{60}$$

is the way to go.

However, if you have $M = 100$ days and $N = 40000$ grid points in your data. Your covariance matrix would then be 40000 by 40000! In this case, you have two options

1. interpolate your data to a courser mesh

2. perform the eigenanalysis on the spatial dispersion relation ($\mathbf{C}$ is $M \times M$). Your general answer will not change, although the eigenvalues, eigenvectors and PCs will all be scaled differently if $M \neq N$. After you standardize your PC and calculate $\mathbf{Z}$, however, you will get identical answers. The important thing is that you must remember that the EOFs are actually the PC's and the PC's are actually the EOFs.

In this case, you want to apply the eigenanalysis to the smaller covariance matrix, that is

$$\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T \rightarrow [M \times M] \tag{61}$$

Then, the eigenvectors will actually be your "PCs" in the sampling dimension, and calculating $\mathbf{Z}$ will lead to the eigenvectors (spatial maps) (although they will differ by a scaling factor).

1.3.5 General summary of finding EOFs by eigenanalyzing the covariance matrix for $\mathbf{X} = [M \times N]$

1. subtract the mean values along the sampling dimension of $\mathbf{X}$

2. unless there are issues with data size and computation time, define $\mathbf{C}$ along the sampling dimension so you are left with a matrix that is the space dimensions, i.e.,

$$\mathbf{C} = \frac{1}{M}\mathbf{X}^T\mathbf{X} \rightarrow [N \times N], \text{ if } M \text{ is the sampling dimension} \tag{62}$$

$$\text{or} \tag{63}$$

$$\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T \rightarrow [M \times M], \text{ if } N \text{ is the sampling dimension} \tag{64}$$

3. eigenanalyze $\mathbf{C}$ by diagonalizing the matrix (e.g. "eig" in Matlab, "EIGENQL" in IDL)

4. the first EOF is the eigenvector corresponding to the largest eigenvalue

5. to find the PC's, project the data onto the set of eigenvectors (PC's should be the length of your sampling dimension)

6. the fraction of variance explained by the i^{th} EOF pair is $\frac{\lambda_i}{\sum_{j=1}\lambda_j}$

Example: SHOW SLIDES OF 8_EOF_PHYSICALVARS.PDF

Example: SHOW WEBPAGE EIGENSTYLE: HTTP://BLOG.THEHACKERATI.COM/POST/126701202241/EIGENSTYLE

1.4 EOFs via Singular Value Decomposition (SVD)

So far, we have discussed how to calculate EOFs $\mathbf{E}$ and their corresponding PC-timeseries $\mathbf{Z}$ using the covariance matrix and eigenanalysis. However, there is another method that can be used. This is called "Singular Value Decomposition" or SVD. This method is very general and can be used in other applications as well.

It can be shown that any $M \times N$ matrix can be decomposed into the product of 3 matrices each with special qualities:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}} \tag{65}$$

where

- $\mathbf{U}$ is $M \times M$
- $\mathbf{\Sigma}$ is $M \times N$
- $\mathbf{V}$ is $N \times N$
- $\mathbf{U}$ and $\mathbf{V}$ are orthogonal
- $\mathbf{\Sigma}$ is a diagonal matrix

If we assume the M dimension is space and the N dimension is time (this is to stay consistent with Prof. Hartmann's notes), something even more exciting follows:

- the columns of $\mathbf{U}$ are the *eigenvectors of* $\mathbf{X}\mathbf{X}^{\mathsf{T}}$
- the columns of $\mathbf{V}$ (rows of $\mathbf{V}^{\mathsf{T}}$) are the *eigenvectors of* $\mathbf{X}^{\mathsf{T}}\mathbf{X}$, or, in other words, are the corresponding PCs
- the r *singular values* along the diagonal of $\mathbf{\Sigma}$ are proportional to the square roots of the nonzero eigenvalues of both $\mathbf{X}\mathbf{X}^{\mathsf{T}}$ and $\mathbf{X}^{\mathsf{T}}\mathbf{X}$, in other words, $s_i = \sqrt{\lambda_i}$
- to obtain the variance explained by each EOF/PC pair, just square the elements of $\mathbf{\Sigma}$

Note! If you have time along the M dimension and space along the N dimension, this just means that the columns of $\mathbf{V}$ are the EOFs and the columns of $\mathbf{U}$ are the PCs. Everything else remains identical.

The matrix $\mathbf{\Sigma}$ is a bit odd, since it is not square, and yet, we are interested in the values along its diagonals (all other values are zero). The matrix can really be viewed in one of two ways.

1. if $M > N$, then

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{D} \\ 0 \end{bmatrix}$$

where $\mathbf{D}$ is an $N \times N$ diagonal matrix and 0 is an $(M - N) \times N$ matrix of zeros. The first r elements of $\mathbf{D}$ are the singular values.

2. In the case where $\mathbf{X}$ is full rank and $M = N$, then $\mathbf{\Sigma}$ is truly a square, diagonal matrix.

If the singular values are rearranged such that σ_1 is the largest value, then

- the first column of $\mathbf{U}$ is the leading EOF of the data
- the first column of $\mathbf{V}$ is the leading PC

1.4.1 Equivalence of SVD to diagonalizing the covariance matrix

To convince you that the two methods for calculating EOFs are similar, we start with the SVD decomposition of our matrix $\mathbf{X}$

$$\mathbf{X} = \mathbf{U\Sigma V}^{\mathsf{T}} \tag{66}$$

Square both sides,

$$\begin{aligned} \mathbf{XX}^{\mathsf{T}} &= (\mathbf{U\Sigma V}^{\mathsf{T}})(\mathbf{U\Sigma V}^{\mathsf{T}})^{\mathsf{T}} && (67)\\ &= (\mathbf{U\Sigma V}^{\mathsf{T}})(\mathbf{V\Sigma}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}}) && (68) \end{aligned}$$

Since $\mathbf{V}^{\mathsf{T}}\mathbf{V} = \mathbf{I}$ (typically the columns of $\mathbf{U}$ and $\mathbf{V}$ are made to be orthonormal),

$$\mathbf{XX}^{\mathsf{T}} = \mathbf{U\Sigma}^2\mathbf{U}^{\mathsf{T}} \tag{69}$$

or, written another way by multiplying both sides by $\mathbf{U}$ leads to

$$N \cdot \mathbf{CU} = \mathbf{U\Sigma}^2 \tag{70}$$

Comparing with the eigenanalysis of the covariance matrix,

$$\mathbf{CU} = \mathbf{U\Lambda} \tag{71}$$

So,

$$\frac{1}{N}\mathbf{\Sigma}^2 = \mathbf{\Lambda} \tag{72}$$

and $\mathbf{U}$ are the eigenvectors of $\mathbf{C} = \frac{1}{N}\mathbf{XX}^{\mathsf{T}}$. Note that if $\mathbf{C} = \frac{1}{M}\mathbf{XX}^{\mathsf{T}}$ for your eigenanalysis then

$$\frac{1}{M}\mathbf{\Sigma}^2 = \mathbf{\Lambda}. \tag{73}$$

Similarly, one can find a relationship between the PCs $\mathbf{Z}$ and $\mathbf{V}$, and it is easy to show that

$$\mathbf{Z} = \mathbf{\Sigma V}^{\mathsf{T}} \tag{74}$$

Note that this is completely consistent with $\mathbf{X} = \mathbf{EZ}$ from the eigenanalaysis, since, starting with the SVD decomposition,

$$\begin{aligned} \mathbf{X} &= \mathbf{U\Sigma V}^{\mathsf{T}} && (75)\\ \mathbf{X} &= \mathbf{UZ} && (76) \end{aligned}$$

and we know that $\mathbf{U} = \mathbf{E}$.

1.4.2 General summary of finding EOFs with SVD for $\mathbf{X} = [M \times N]$

1. subtract the mean values along the sampling dimension of $\mathbf{X}$
2. decompose $\mathbf{X}$ using built-in SVD routine
3. if M is the sample space (time), the EOFs (spatial patterns) correspond to the columns of $\mathbf{V}$ and the PCs (time series) to the columns of $\mathbf{U}$
4. if N is the sample space (time), the EOFs (spatial patterns) correspond to $\mathbf{U}$ and the PCs (time series) to $\mathbf{V}$
5. the fraction of variance explained by the i^{th} EOF pair is $\frac{\sigma_i^2}{\sum_{j=1} \sigma_j^2}$

1.5 EOFs with actual data

1.5.1 Preparing the data

Removing the sampling mean

- it is typical to remove the sampling dimension mean.
- this will make your PC's have a mean of zero
- you may want to remove the seasonal cycle if this is not of interest to you (or diurnal cycle)

Removing the spatial mean

- it is less common to remove the spatial mean since you often want to allow the EOFs to have the same pattern everywhere
- some exceptions: you are interested in the gradients in temperature, not the full temperature field itself
- you are interested in the eddy component of the field, so you might remove the zonal-mean

Standardizing the data

- sometimes, it is desirable to take the amplitude out of the data before doing EOF analysis, for example, when
  - your data is made of a combination of parameters with different units, and you don't want the parameter with the biggest units to dominate the variance

  - if the variance of the data varies greatly from space to space (and this is not of interest to you)

- the way to do this is after subtracting the mean along the sampling dimension, divide by the standard deviation as well (now you have units of σ)

- note that you will be eigenanalyzing the *correlation matrix* (not the covariance matrix) if you use the eigenanalysis method

Weighting

- if the data is gridded, you need to weight according to grid area:

  - if you are calculating EOFs using gridded maps where the area of each grid box is different, you weight your data anomaly matrix by $\sqrt{\cos\theta}$ (since when you multiple the grid points by each other you will get your full $\cos\theta$ in your covariance matrix. See Chung and Nigam (1999). This works for SVD too!

  - note that the resulting EOF will look unphysical and have unphysical values, this will be remedied when we discuss how to present EOFs

Missing data

- most programs will not perform eigenanalysis or SVD with missing values or NaNs

- the best option is to only use locations with $\alpha\%$ of the data present, and calculate the EOFs via the covariance matrix so that $\mathbf{C}$ has no missing data when input into the eigenanalysis function

1.5.2 Presentation of EOFs

Let $\mathbf{X}$ have dimensions $[M \times N]$ such that M is the number of samples and N is the spatial locations.

From now on, to make notation a bit easier, we will only write equations using EOF 1 and PC 1. The steps can be repeated for EOF 2 and PC 2, EOF 3 and PC 3, etc.

In the literature, one sees both the EOFs and the PCs plotted, depending on what is of interest.

Generally, the spatial domain is more interesting since the PC (sampling domain) is usually noisy.

Problem: the EOFs don't have physical units, so we need to figure out how to plot the patterns in a meaningful way

Here is how it is done:

1. calculate the leading PC Z (in a variety of ways, either you get it directly from SVD, or using eigenanalysis you project the data and EOF onto each other)
2. standardize Z (the mean is already 0 if you subtracted the mean from the data, so just divide by σ_Z)
3. project Z back onto your *original* anomaly data and divide by the length of Z to get D, the EOF 1 structure in the original units i.e. the unweighted data X. If you standardized X before the computation, you want to project onto the non-standardized data.

If the original data is $\mathbf{X}$, the weighted data is $\mathbf{X}_w$, following eigenanalysis

$$z_1 = \mathbf{X}_w e_1 \tag{77}$$

Then,

$$\tilde{z}_1 = \frac{z_1 - \mu_{z_1}}{\sigma_{z_1}} \tag{78}$$

Then,

$$d_1 = \frac{1}{M}\tilde{z}_1^T \mathbf{X} = \frac{1}{\text{length}(z_1)}\tilde{z}_1^T \mathbf{X} \tag{79}$$

and now d_1 is in physical units of your data, and denotes the anomaly associated with 1 standard deviation variation of z_1.

Note that this procedure works wether you used the covariance method or SVD. For SVD, there is one less step since the raw PC 1 is already given to you.

1.5.3 How many EOFs should be retained?

EOFs will always give you an answer. For example, EOFs of red noise will give you sines and cosines that look physical. The rate at which the eigenvalues drop off will be a function of the redness of the data.

The orthogonality of EOFs impose a constraint on their shape! Higher order EOFs are often trying to represent the noise while still being orthogonal to the other EOFs.

North et al. (1982) argues that the significance of an EOF is a function of the degree of separation between eigenvalues, where the 95% confidence bounds on the eigenvalues is

$$\Delta\lambda = \lambda\sqrt{\frac{2}{N^*}} \tag{80}$$

In this case, the N^* is our friend the "effective degrees of freedom". This is not obvious how to decide what N^* is - you need to come up with a value that is representative of the entire data set!

So, when in doubt, it is a good idea to test other methods of robustness, e.g.

- subdividing your sample and comparing EOF structures, are they the same?

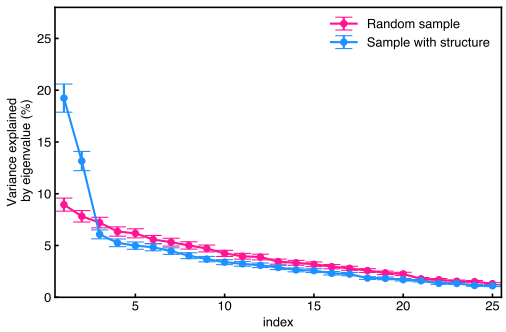

Figure 3: Example figure for displaying eigenvalues and their 95% confidence bounds based on North et al. (1982).

- are your results explainable through theory?
- are results sensitive to the size of the domain? (if highly sensitive, this could be sines and cosines being fit inside the boundary)

An example of a plot showing the eigenvalues and their 95% confidence bounds following North et al. (1982) is given in Figure 3. The pink bars are the eigenvalues of a data set that is pure white noise. The blue bars denote the eigenvalues from a data set that has some structure, specifically, the first eigenvector explains substantially more variance than the others.

Wave-like phenomena

Note, if you have a traveling wave in your data, it will appear as *two* eigenvalues that are close together that are spaced away from the others. These are the paired sine and cosine portions of the wave.

Example: PYTHON NOTEBOOK EOF_FROM_SCRATCH.PY

Example: PYTHON NOTEBOOK EOF_NOISE_REDUCTION.PY

1.5.4 Recipe for calculating EOFs: temporal covariance matrix

Assume that $\mathbf{X} = [M \times N] = [\text{sampling} \times \text{space}]$
$\mathbf{e}_1, \mathbf{d}_1 = [N \times 1]$ and $\mathbf{z}_1 = [M \times 1]$

1. **Prepare the data**
   - calculate anomalies along the sampling dimension, and denote this matrix as $\mathbf{X}$ - this may only require subtracting the sample mean, but could also include removing the seasonal cycle, etc.
   - decide if you want to standardize your data (divide by the standard deviation in the sampling dimension) - call this modified matrix $\mathbf{X}_w$
   - remember to save your original *anomaly* data matrix $\mathbf{X}$, you will need it later
2. **Calculate the temporal covariance matrix**
   - $\mathbf{C} = \frac{1}{M}\mathbf{X}_w^T\mathbf{X}_w$
   - if you need to area-weight the data (e.g. gridded global data sets when the area of a grid-point decreases toward the pole), you will want to do this now.
3. **Perform eigenanalysis**
   - use built-in functions to do this - the code should output your eigenvectors $\mathbf{E}$ and eigenvalues $\mathbf{\Lambda}$ (eigenvalues λ are the values along the diagonal of $\mathbf{\Lambda}$)
   - $[\mathbf{E}\ \mathbf{\Lambda}] = eig(C)$; (in Matlab)
4. **Calculate & plot variance explained by each EOF/PC pair**
   - variance explained is the eigenvalue divided by the sum of all of the eigenvalues:
   - fraction variance $= \lambda_i / \sum_j \lambda_j$
   - plot the variance explained and determine which of the first few are distinct - one method is to use the North et al. (1982) 95% confidence bounds and look where they don't overlap: $\Delta\lambda = \lambda\sqrt{\frac{2}{N^*}}$
5. **Retain EOFs you are interested in**
   - keep the EOFs that look interesting according to their variance explained, this may only be the first EOF (first column of $\mathbf{E}$), denoted $\mathbf{e}_1$
6. **Calculate the PCs**
   - calculate PC 1, denoted $\mathbf{z}_1$, by regressing the weighted/standardized anomaly matrix $\mathbf{X}_w$ onto $\mathbf{e}_1$ (repeat for other EOF pairs)
   - that is, $\mathbf{z}_1 = \mathbf{X}_w\mathbf{e}_1$ and $\mathbf{z}_2 = \mathbf{X}_w\mathbf{e}_2$, and so on ...
7. **Standardize the PCs**
   - standardize $\mathbf{z}_1$ by subtracting its mean and dividing by its standard deviation, denoted as $\tilde{\mathbf{z}}_1$
   - $\tilde{\mathbf{z}}_1 = \frac{z_1 - \mu_{z_1}}{\sigma_{z_1}}$
   - the values of $\tilde{\mathbf{z}}_1$ are now in units of σ (this is the PC you want to use from now on)
8. **Calculate EOFs in physical units**
   - project $\tilde{\mathbf{z}}_1$ onto the *original* anomaly data $\mathbf{X}$ to get EOF 1 in physical units, denoted $\mathbf{d}_1$
   - $\mathbf{d}_1 = \frac{1}{M}\tilde{\mathbf{z}}_1^T\mathbf{X}$
   - $\mathbf{d}_1$ is in units of the original data, and the field has the meaning that it is *"the anomaly associated with 1 standard deviation variation of the PC time series"*
   - note that if $\mathbf{X} \equiv \mathbf{X_w}$, $\mathbf{d}_1$ will have the same pattern as $\mathbf{e}_1$, just different values. However, if you area-weighted or standardized your data, the spatial patterns will be different

1.5.5 Recipe for calculating EOFs: spatial dispersion matrix

Assume that $\mathbf{X} = [M \times N] = [\text{sampling} \times \text{space}]$
$\mathbf{e}_1, \mathbf{d}_1 = [N \times 1]$ and $\mathbf{z}_1 = [M \times 1]$

1. **Prepare the data**
   - calculate anomalies along the sampling dimension, and denote this matrix as $\mathbf{X}$ - this may only require subtracting the sample mean, but could also include removing the seasonal cycle, etc.
   - decide if you want to standardize your data (divide by the standard deviation in the sampling dimension) - call this modified matrix $\mathbf{X}_w$
   - remember to save your original *anomaly* data matrix $\mathbf{X}$, you will need it later
2. **Calculate the spatial dispersion matrix**
   - $\mathbf{C} = \frac{1}{N}\mathbf{X}_w\mathbf{X}_w^T$
   - if you need to area-weight the data (e.g. gridded global data sets when the area of a grid-point decreases toward the pole), you will want to do this now.
3. **Perform eigenanalysis**
   - use built-in functions to do this - the code should output your PC's in $\mathbf{E}$ and eigenvalues in $\mathbf{\Lambda}$ (eigenvalues λ are the values along the diagonal of $\mathbf{\Lambda}$)
   - $[\mathbf{E}\ \mathbf{\Lambda}] = eig(C)$; (in Matlab)
4. **Calculate & plot variance explained by each EOF/PC pair**
   - variance explained is the eigenvalue divided by the sum of all of the eigenvalues:
   - fraction variance $= \lambda_i / \sum_j \lambda_j$
   - plot the variance explained and determine which of the first few are distinct - one method is to use the North et al. (1982) 95% confidence bounds and look where they don't overlap: $\Delta\lambda = \lambda\sqrt{\frac{2}{N^*}}$
5. **Retain PC's you are interested in**
   - keep the PCs that look interesting according to their variance explained, this may only be the first column of $\mathbf{E}$, denoted $\mathbf{z}_1$ (since this is actually the first PC because you are analyzing the dispersion matrix)
6. **Standardize the PCs**
   - standardize $\mathbf{z}_1$ by subtracting its mean and dividing by its standard deviation, denoted as $\tilde{\mathbf{z}}_1$
   - $\tilde{\mathbf{z}}_1 = \frac{z_1 - \mu_{z_1}}{\sigma_{z_1}}$
   - the values of $\tilde{\mathbf{z}}_1$ are now in units of σ (this is the PC you want to use from now on)
7. **Calculate EOFs in physical units**
   - project $\tilde{\mathbf{z}}_1$ onto the *original* anomaly data $\mathbf{X}$ to get EOF 1 in physical units, denoted $\mathbf{d}_1$
   - $\mathbf{d}_1 = \frac{1}{M}\tilde{\mathbf{z}}_1^T\mathbf{X}$, $\mathbf{d}_2 = \frac{1}{M}\tilde{\mathbf{z}}_2^T\mathbf{X}$, and so on...
   - $\mathbf{d}_1$ is in units of the original data, and the field has the meaning that it is *"the anomaly associated with 1 standard deviation variation of the PC time series"*

1.5.6 Recipe for calculating EOFs: SVD

Assume that $\mathbf{X} = [N \times M] = [\text{space} \times \text{sampling}]$
$\mathbf{e}_1, \mathbf{d}_1 = [N \times 1]$
$\mathbf{z}_1 = [M \times 1]$

1. **Prepare the data**
   - calculate anomalies along the sampling dimension, and denote this matrix as $\mathbf{X}$ - this may only require subtracting the sample mean, but could also include removing the seasonal cycle, diurnal cycle, etc.
   - decide if you want to standardize your data (divide by the standard deviation in the sampling dimension) - call this modified matrix $\mathbf{X}_w$
   - if you need to area-weight the data (e.g. gridded global data sets when the area of a grid-point decreases toward the pole), you will want to do this now.
   - remember to save your original *anomaly* data matrix $\mathbf{X}$, you will need it later
2. **Perform SVD**
   - use built-in functions to do this - the code should output $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{\Sigma}$ (singular values = s are along the diagonal of $\mathbf{\Sigma}$)
   - $[\mathbf{U}\ \mathbf{\Sigma}\ \mathbf{V}] = svd(\mathbf{X}_w)$; (in Matlab)
   - the eigenvectors are vectors of $\mathbf{U}$, the PCs are the vectors of $\mathbf{V}$, and the singular values are along the diagonal of $\mathbf{\Sigma}$
3. **Calculate & plot variance explained by each EOF/PC pair**
   - variance explained is the singular value *squared* (s^2) divided by the sum of all of the *squared* singular values: fraction variance $= s_i^2 / \sum_j s_j^2$
   - plot the variance explained and determine which of the first few are distinct - one method is to use the North et al. (1982) 95% confidence bounds and look where they don't overlap: $\Delta\lambda = \lambda\sqrt{\frac{2}{N^*}}$
4. **Retain EOFs you are interested in**
   - keep the EOFs that look interesting according to their variance explained, this may only be the first EOF (first column of $\mathbf{U}$), denoted $\mathbf{e}_1$
5. **Calculate PCs**
   - the PCs having already been calculated for you, so PC 1 is the first column of $\mathbf{V}$, denote the first PC as $\mathbf{z}_1$
6. **Standardize PCs**
   - standardize $\mathbf{z}_1$ by subtracting its mean and dividing by its standard deviation, denoted as $\tilde{\mathbf{z}_1}$
   - $\tilde{\mathbf{z}}_1 = \frac{z_1 - \mu_{z_1}}{\sigma_{z_1}}$
   - the values of $\tilde{\mathbf{z}}_1$ are now in units of σ
7. **Calculate EOFs in physical units**
   - project $\tilde{\mathbf{z}}_1$ onto the original anomaly data $\mathbf{X}$ to get the EOF 1 in physical units, denoted $\mathbf{d}_1$
   - $\mathbf{d}_1 = \frac{1}{M}\mathbf{X}\tilde{\mathbf{z}}_1$, $\mathbf{d}_2 = \frac{1}{M}\mathbf{X}\tilde{\mathbf{z}}_2$, and so on...
   - $\mathbf{d}_1$ is in units of the original data, and the field has the meaning that it is *"the anomaly associated with 1 standard deviation variation of the PC time series"*
   - note that if $\mathbf{X} \equiv \mathbf{X}_\mathbf{w}$, $\mathbf{d}_1$ will have the same pattern as $\mathbf{e}_1$, just different values - however, if you area-weighted or standardized your data, the spatial patterns will actually be different

2 Cluster Analysis

Cluster analysis is a popular technique in data mining and its goal is to group a set of observations into a number of distinct clusters. The end result is that each observation belongs to only one cluster, and this clustering is determined based on a particular cost function. Determining the optimal set of clusters is often an iterative process that begins with an initial guess.

Recall that the principal components in EOF analysis tell you how much a given observation looks like a particular EOF pattern. While, an observation may look predominately like one particular EOF (thus, the principal component for this EOF is large), it likely still has non-zero principal component values for the other EOFs. In cluster analysis, this is not allowed, as each observation is tied to only one cluster.

2.1 k-means Clustering

k-means clustering is one of the more popular clustering techniques in Earth Science due to its relative simplicity. The aim is to group N observations into k clusters in which each observation belongs to the cluster with the nearest center. This results in the data being partitioned into *Voronoi cells*. The optimal distribution of the cluster centers is the distribution that *minimizes the within cluster sum of squares* (i.e. the sum of the distance functions of each point to the cluster center). Mathematically, this is written as:

$$\underset{\mathbf{S}}{\operatorname{argmin}} = \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathbf{S}_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \tag{81}$$

where $\mathbf{S} = S_1, S_2, ..., S_k$ denotes the set of k clusters, $\mathbf{x}$ denotes the set of observations, $\boldsymbol{\mu}_i$ denotes the center of cluster S_i defined as the mean of all points in S_i, and argmin specifies that we are looking for the minimum over varying options of $\mathbf{S}$.

While the idea behind k-means is straightforward, actually computing the most optimal solution is very difficult and time intensive, especially for large data sets. Thus, most applications of k-means clustering employ heuristic algorithms that converge quickly to a local optimal solution that may or may not be the global optimal solution. Thus, in practice, one typically runs the k-means clustering algorithm many times (with each instance starting with a unique initial guess) and takes the iteration that produces the smallest minimum within cluster sum of squares (i.e. (81)).

The standard algorithm for k-means clustering is Lloyd's algorithm, which begins with an initial guess of the cluster centers $\boldsymbol{\mu}_i$ and then iteratively refines this guess until the it converges. Fig. 4 shows an example of this iterative process.

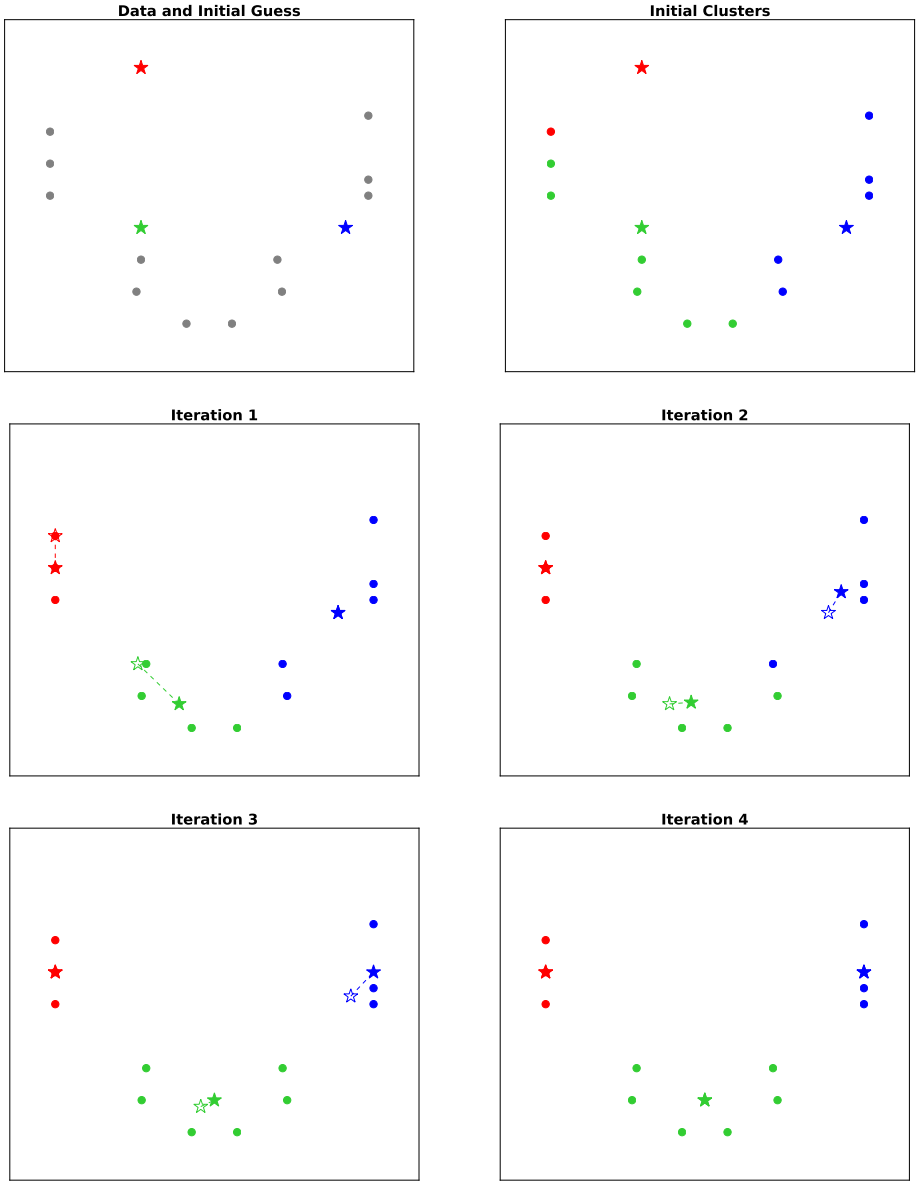

Figure 4: An example of the iterative refinement of k*-means clustering using Lloyd's algorithm. Cluster centers are shows as stars, and empty stars denote the previous cluster center. The algorithm has converged after the third iteration.*

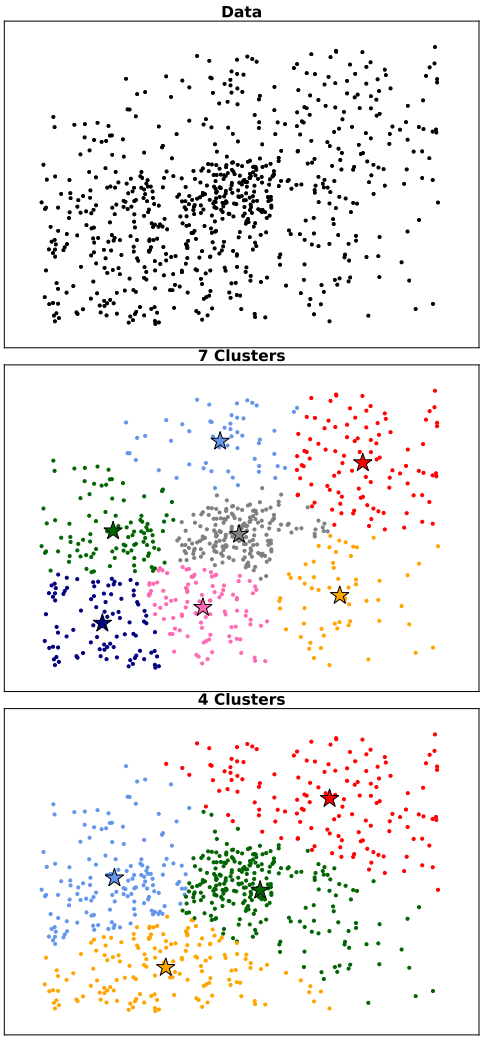

Figure 5: Using k-means clustering to identify 4 and 7 clusters in the data. The left panel shows the raw data, and the middle and right panels show the same points but now colored by the cluster which within they resides. The stars denote the centroids of each of the seven clusters. The k-means algorithm was run 20 different times and the iteration with the optimal cost function was retained.

To apply k-means clustering, two decisions that must be made:

1. **How many clusters do you want?** This is a decision that must be made by the user, and is not determined by the clustering algorithm. One will obtain different results based on the choice of k as seen in Fig. 5. There are methods such as the *gap statistic* that can be used to help determine which k to use (Tibshirani et al. 2001).

2. **How should you seed your initial guess?** There are many options available for how to best make your initial guess at the cluster centers. In addition, running k-means clustering many times, and choosing the optimal solution out of these is a way to avoid your final solution being too dependent on the initial guess.

2.2 Self-organizing maps (SOMs)

A self-organizing map, also known as a *Kohonen map*, is yet another form of cluster analysis that uses unsupervised machine learning to train an artificial neural network. Under certain conditions it is identical to the k-means clustering method, however, the distinguishing feature of SOMs is that when one cluster center is updated, the neighboring cluster centers can (although not required) also be updated in a similar way. This results in a set of cluster centers (or *nodes*) that are organized such that nodes that are most similar are located near one another and nodes that are least similar are further apart when the are organized into an $m \times p$ grid.

Like most artificial neural networks, the SOMs algorithm can be summarized as a two step process: training and mapping.

1. **Training:** The training phase is when the SOM algorithm does the heavy thinking. During training, the SOM nodes are updated by comparing them to input examples (i.e. observations) that are ingested one at a time. Often, the same training data is iterated through multiple times. The result is a set of trained SOM nodes.

2. **Mapping:** After training is complete, the mapping phase involves automatically classifying a new observation as belonging to one of the nodes. In some cases, the training data is distinct from the data you are interested in mapping, while in other cases they may be one and the same.

As with k-means clustering, the user must first determine how many SOM nodes they wish to create to represent their data. Unlike k-means, however, this information is provided as an $m \times p$ grid, and thus, the user must choose both m and p. In the example below we will choose a SOM size of 20×20 giving a total of 400 nodes, however, it is important to note that the resulting SOM nodes are highly dependent on the dimensions chosen (just as in k-means).

SOM training begins by first initializing the $m \cdot p$ nodes. This can be done either with random data, or using the principal components of the observations. Then, the algorithm proceeds by ingesting the training data either one observation at a time (*online training*) or as a single group (*batch training*). The observations are compared to each of the $m \cdot p$ nodes and the *best match unit* (BMU) is identified as the node with the smallest Euclidean distance to the observation in question. We will denote this BMU as n_i to signify the i^{th} node, where $1 \leqslant i \leqslant m \cdot p$. Finally, the BMU and neighboring nodes are updated in the following manner:

$$n_i(s+1) = n_i(s) + \alpha(s)\Theta(s)[\mathbf{x}(t) - n_i(s)] \tag{82}$$

where s is the step index, α is the learning rate parameter, Θ is the neighborhood function, t is an index into the training data set, and $\mathbf{x}(t)$ is the observation (input vector). Specific descriptions of these inputs are given below.

- **step index (s):** current iteration number
- **training time (t):** index into the training sample

- **learning rate parameter (α):** how strongly to update the nodes given the observation; typically decreases with training time
- **neighborhood function (Θ):** a function describing how many surrounding nodes to update besides the BMU; typically decreases with training time

The learning rate parameter and the neighborhood function must all be chosen by the user. Often, the learning rate parameter starts large, and then decreases linearly with training time. There are many neighborhood functions to choose from, but some common ones are a 2D Guassian or the Epanechikov function. It is typically considered good practice to initially set your neighborhood function parameters to include the entire SOM-space to ensure that all nodes are updated. This is then modified at later training times. Typically, the SOM algorithm is implemented in multiple stages, with varying combinations of the above parameters.

Since SOM analysis has a large number of free parameters chosen by the user, there are many possible SOMs that can be computed from a single set of observations. One question is, how do I choose which one to use? Two key metrics should be considered:

1. **quantization error:** the average distance between each observation and its BMU
2. **topographic error:** the fraction of all observations who's first and second BMUs are *not* adjacent units

One wants to minimize both the quantization error and the topographic error, and typically, the topographic error is kept near zero while the quantization error is minimized. Thus, it is good practice to perform SOM analysis multiple times on the same data set, using different paramater values, and then choose the best based on the quantization and topographic errors.

To demonstrate SOM analysis and how the results can be visualized, we use hourly observations from Christman Field in Fort Collins, Colorado from January 1 - December 31, 2016. The data input into the algorithm is a 2D matrix X with dimensions [8784, 6]. That is, 8784 hourly observation periods and 6 variables: temperature (degrees F), relative humidity (%), wind speed (mph), pressure (mb), solar radiation (W/m^2) and precipitation (mm).

SOM training was performed in two phases: *rough* and *finetune*, each of which had different training lengths and initial and final radii of influence (for use in the neighborhood function). The SOM nodes were initialized with random data and a 20$\times$ 20 grid was chosen. The resulting SOM nodes are plotted in Fig. 6, where each panel denotes a different variable and each grid box denotes a different SOM. Fig. 7 displays the frequency of occurrence of each of the SOM nodes (i.e. how many observations had that particular node as their BMU).

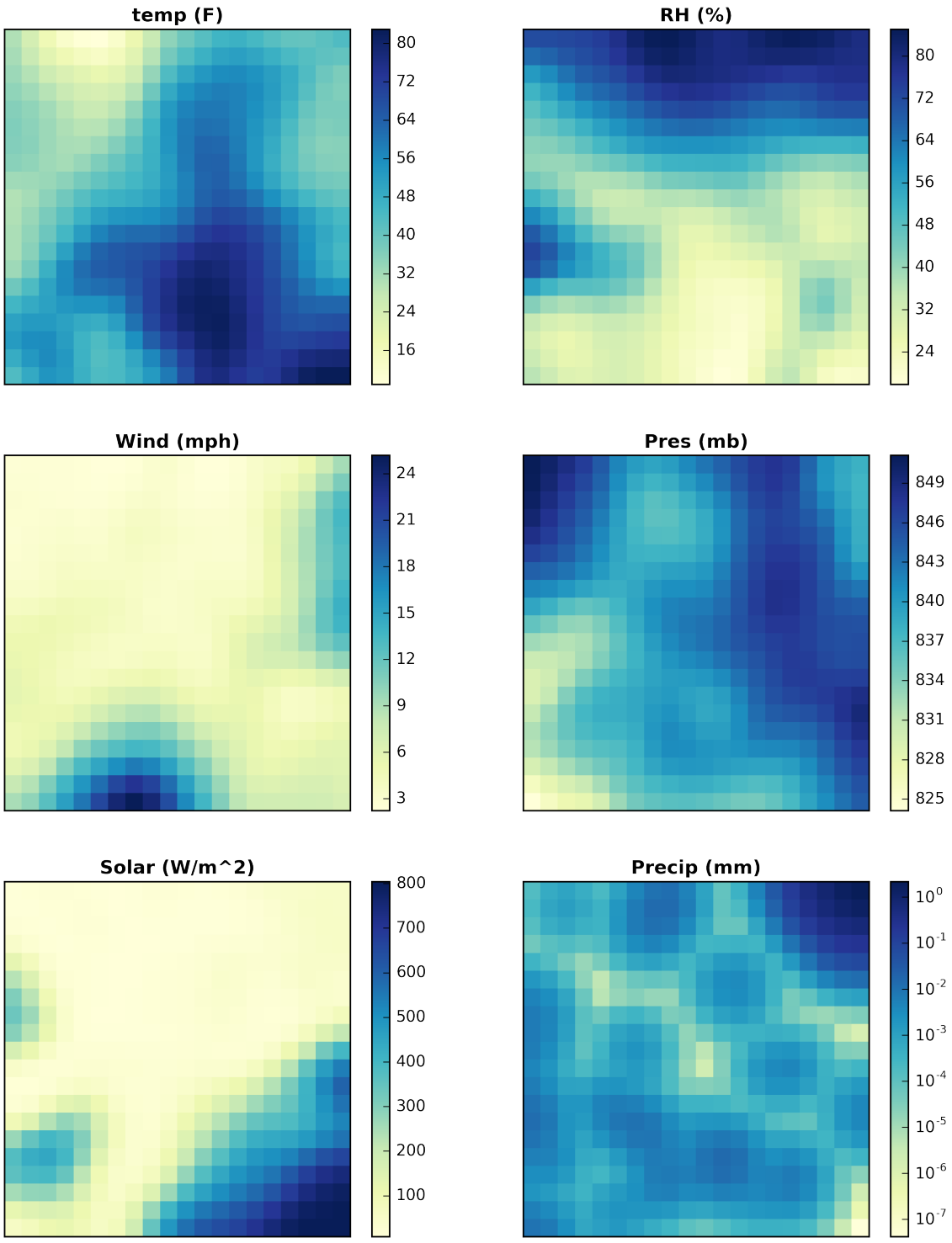

Figure 6: SOM weights for each variable displayed in a 20×20 *grid.*

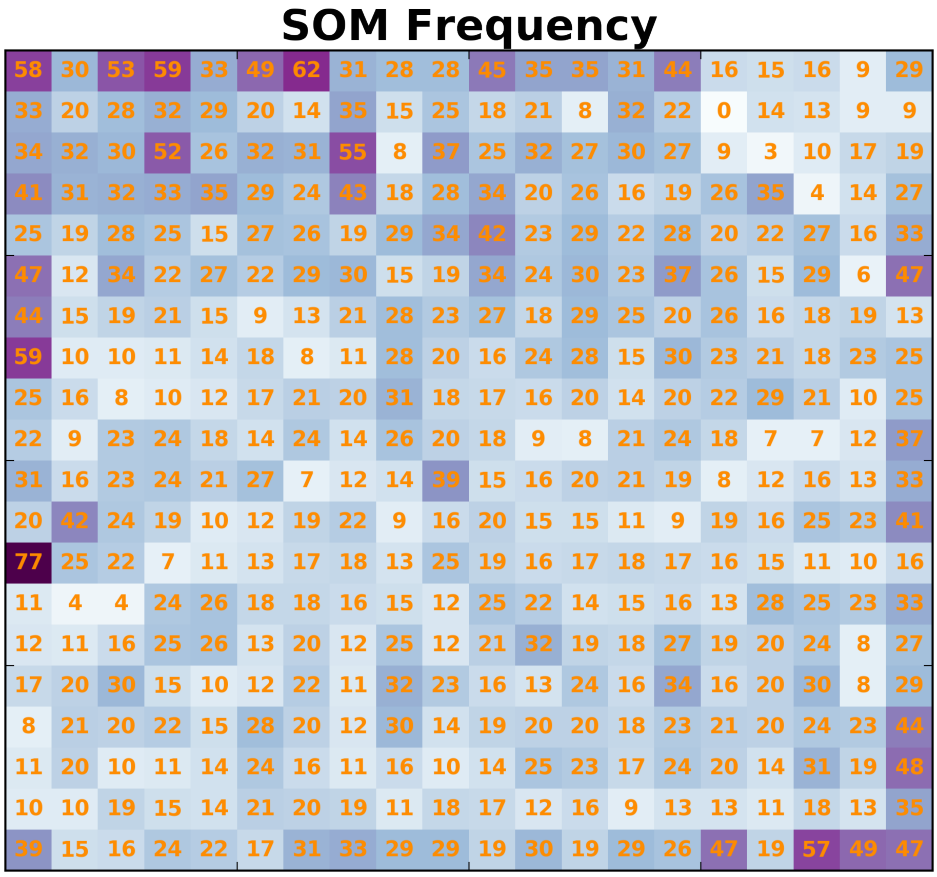

Figure 7: The frequency of occurrence of each of the 400 SOM patterns. Darker colors denote more frequent, and the number of occurrences out of the 8784 observations is written in orange.

Example resources that use SOMs. The Gervais et al. (2016) has a wonderful detailed explanation of SOM analysis. Highly recommended reading!

- Gervais, M., Atallah, E., Gyakum, J. R., & Bruno Tremblay, L. (2016). Arctic air masses in a warming world. Journal of Climate, 29(7), 2359-2373. http://doi.org/10.1175/JCLI-D-15-0499.1
- Lee, S., & Feldstein, S. B. (2013). Detecting ozone- and greenhouse gas-driven wind trends with observational data. Science (New York, N.Y.), 339(6119), 563-7. http://doi.org/10.1126/science.1225154
- Skific, N., Francis, J. A., & Cassano, J. J. (2009). Attribution of projected changes in atmospheric moisture transport in the Arctic: A self-organizing map perspective. Journal of Climate, 22(15), 4135-4153. http://doi.org/10.1175/2009JCLI2645.1
- python SOM code and example: `https://github.com/sevamoo/SOMPY/blob/master/sompy/examples/California\%20Housing.ipynb`