5 Matrices for Climate Data

Matrices appear everywhere in climate science. For examples, climate data may be written as a matrix – a 2-dimensional rectangular array of numbers or symbols – and most data analyses and multivariate statistical studies require the use of matrices. The study of matrices is often included in a course known as linear algebra. This chapter is limited to (i) describing the basic matrix methods needed for this book, such as the inverse of a matrix and the eigenvector decomposition of a matrix, and (ii) presenting matrix application examples of real climate data, such as the sea level pressure data of Darwin and Tahiti. From climate data matrices, we wish to extract helpful information, such as the spatial patterns of climate dynamics (e.g., El Niño Southern Oscillation), and temporal occurrence of the patterns. These are related to eigenvectors and eigenvalues of matrices. This chapter features the space-time data arrangement, which uses rows of a matrix for spatial locations, and columns for temporal steps. The singular value decomposition (SVD) helps reveal the spatial and temporal features of climate dynamics as singular vectors and the strength of their variability as singular values.

To better focus on matrix theory, some application examples of linear algebra, such as the balance of chemical reaction equations, are not included in the main text, but are arranged as exercise problems. We have also designed exercise problems for the matrix analysis of real climate data from both observations and models.

5.1 Matrix Definitions

Matrices have appeared earlier in this book, e.g., the matrix formulation of a multilinear regression in Chapter 4. A matrix is a rectangular array of numbers (or even expressions), often denoted by an uppercase letter in either boldface or plain, $\mathbf{A}$ or A:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ a_{21} & a_{22} & \cdots & a_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{np} \end{bmatrix} \tag{5.1}$$

A Sample Space-Time Dataset

Lat	Lon	1934-3	1934-4	1934-5	1934-6	1934-7	1934-8	1934-9	1934-10	1934-11	1934-12	1935-1	1935-2	1935-3	1935-4	1935-5
32.5	**242.5**	1.86	1.14	1.03	−0.65	0.12	−0.27	−0.30	−0.30	0.13	0.34	−0.48	−0.18	−1.43	−0.50	−0.84
32.5	**247.5**	3.14	2.42	2.29	−2.08	0.93	−0.09	−0.49	0.46	0.21	0.79	0.07	−0.18	−2.04	−0.44	−2.32
32.5	**252.5**	1.42	1.94	2.47	−0.24	1.18	1.04	0.11	1.30	0.46	0.73	0.93	−1.07	−0.09	0.18	−2.32
32.5	**257.5**	−0.97	0.96	0.95	1.89	1.44	1.49	0.50	2.68	1.55	0.87	2.65	−0.36	1.95	0.64	−2.18
32.5	**262.5**	−1.89	1.09	0.51	2.64	2.12	2.36	0.07	2.67	1.90	0.47	2.40	0.14	2.50	0.04	−1.66
32.5	**267.5**	−1.36	0.82	-0.06	1.36	0.89	1.46	-0.55	2.28	1.19	−0.17	2.14	0.54	3.27	0.21	−0.35
32.5	**272.5**	−0.98	0.61	0.26	0.82	0.36	0.38	-0.15	1.37	0.93	−0.66	1.31	0.23	2.45	0.76	0.85
32.5	**277.5**	−1.26	0.51	-0.24	0.75	0.65	0.39	0.71	0.91	0.36	−0.85	0.99	−0.25	2.12	0.68	0.90
32.5	**282.5**	0.54	0.88	0.09	0.25	0.32	0.13	0.20	1.08	0.51	0.55	0.81	0.87	1.31	0.94	0.77
32.5	**287.5**	0.72	0.99	0.29	0.39	0.36	0.12	0.50	1.04	0.32	0.38	0.23	0.46	0.62	0.58	0.39
32.5	**292.5**	0.79	0.93	0.27	0.48	0.23	0.18	0.90	1.01	0.48	0.22	−0.23	0.11	0.21	0.30	−0.04
32.5	**297.5**	0.68	0.59	0.26	0.33	0.17	0.17	0.82	0.69	0.50	0.26	−0.42	−0.15	−0.11	0.07	−0.37
32.5	**302.5**	0.63	0.42	0.33	0.35	0.46	0.21	0.65	0.48	0.34	0.27	−0.64	−0.27	−0.40	−0.13	−0.59
32.5	**307.5**	0.69	0.43	0.48	0.54	0.69	0.20	0.46	0.33	0.25	0.26	−0.63	−0.15	−0.37	−0.12	−0.54
32.5	**312.5**	0.80	0.51	0.44	0.44	0.68	0.26	0.45	0.41	0.26	0.28	−0.28	0.08	−0.16	0.05	−0.21
32.5	**317.5**	0.83	0.47	0.16	0.26	0.61	0.36	0.47	0.49	0.14	0.10	−0.01	0.21	0.04	0.23	0.24
32.5	**322.5**	0.62	0.16	−0.19	0.10	0.44	0.39	0.41	0.43	−0.04	−0.09	−0.10	0.10	0.09	0.33	0.45
32.5	**327.5**	0.24	−0.29	−0.54	0.05	0.27	0.29	0.21	0.23	−0.35	−0.19	−0.21	−0.03	0.09	0.40	0.52

Figure 5.1 A subset of the monthly surface air temperature anomalies from the NOAAGlobalTemp Version 4.0 dataset.

This is an $n \times p$ matrix, in which a_{ij} are called elements or entries of the matrix $\mathbf{A}$, i is the row index from 1 to n, and j is the column index from 1 to p. The dimensions of a matrix are denoted by subscript, e.g., $\mathbf{A}_{n\times p}$ indicating that the matrix $\mathbf{A}$ has n rows and p columns. The matrix may also be indicated by square brackets around a typical element $\mathbf{A} = [a_{ij}]$ or sometimes $\{A\}_{ij}$, maybe even A_{ij}. If $n = p$, then the array is a square matrix.

Figure 5.1 is an example of a space-time climate data matrix. It is a subset of the $5^\circ \times 5^\circ$ gridded monthly surface air temperature anomalies from the NOAA Merged Land Ocean Global Surface Temperature Analysis (NOAAGlobalTemp) (Version 4.0). The rows are indexed according to the spatial locations prescribed by the latitude and longitude of the centroid of a $5^\circ \times 5^\circ$ grid box (see the entries of the first two columns in boldface). The columns are indexed according to time (see the first-row entries in boldface). The other entries are the temperature anomalies with respect to the 1971–2000 monthly climatology. The anomalies are arranged according to the locations by rows and the time by columns. The units for the anomaly data are °C.

For a given month, the spatial temperature data on the Earth's surface is itself a 2-dimensional array. To make a space-time matrix, we assign each grid box a unique index s from 1 to n if the spatial region has n grid boxes. The index assignment is subjective, depending on the application needs. The commonly used way is to fix a latitude and increase the index number as the longitude increases, as indicated by the first two columns of the data matrix shown in Figure 5.1. When the longitude is finished at this latitude band, go to the next latitude band until the completion of the latitude range. This can go from south to north, or from north to south. Of course, one can fix the longitude first, and increase the index according to the ascending or descending order of latitudes.

Following this spatial index as the row number, the climate data for a given month is a column vector. If the dataset has data for p months, then the space-time data matrix has p columns. If the dataset has n grid boxes, then the data forms an $n \times p$ space-time data matrix. You can conveniently use row or column operations of a computer language to calculate statistics of the dataset, such as spatial average, temporal mean, temporal variance, etc.

For more explicit indication of space and time, you may use s for the row index and t for the column index in a space-time data matrix. Thus, $[A_{st}]$ indicates a space-time data matrix (s for space and t for time).

This space-time indexing can be extended to the data in 3D space and 1D time, as long as we can assign a unique ID s from 1 to n for a 3D grid box and a unique ID t for time. The presently popular netCDF (Network Common Data Form) data format in climate science, denoted by `.nc`, uses this index procedure for a 4D dataset. For example, to express the output of a 3D climate model, you can start your index longitude first for a given latitude and altitude. When longitude exhausts, count the next latitude. When the latitude exhausts, count the next altitude until the last layer of the atmosphere or ocean. Eventually, a space-time data matrix is formed $[A_{st}]$.

To visualize the row data of a space-time data matrix $[A_{st}]$, just plot a line graph of the row data against time. To visualize the column data of a space-time data matrix $[A_{st}]$, you need to convert the column vector into a 2D pixel format for a 2D domain (e.g., the contiguous United States (CONUS) region), or a 3D data array format for a 3D domain (e.g., the CONUS atmosphere domain from the 1,000 mb surface level to the 10 mb height level). This means that the climate data are represented in another matrix format, such as the surface air temperature anomaly data on a 5-degree latitude–longitude grid for the entire world for December 2015 visualized by Figure 1.8. The data behind the figure is a 36×72 data matrix on the grid whose rows are for latitude and columns for longitude. This data matrix is in space-space pixel format like the data for a photo. Each time corresponds to a new space-space pixel data matrix. Thus, the latitude–longitude-time forms a 3D data array. With elevation, then latitude–longitude-altitude-time forms a 4D data array, which is often written in the netCDF file in climate science. You can use the 4DVD data visualization tool `www.4dvd.org`, described in Chapter 1, to visualize the 4D Reanalysis data array as an example to understand the space-time data plot, and netCDF data structure.

The coordinates (32.5, 262.5) in the sixth row of Figure 5.1 indicate a $5^\circ \times 5^\circ$ grid box centered at (32.5° N, 97.5° W). This box covers part of Texas, USA. The large temperature anomalies for the summer of 1934 (2.64°C for June, 2.12°C for July, and 2.36°C for August) were in the 1930s Dust Bowl period. The hot summer of 1934 was a wave of severe drought. The disastrous dust storms in the 1930s over the American and Canadian prairies destroyed many farms and greatly damaged the ecology.

5.2 Fundamental Properties and Basic Operations of Matrices

This section provides a concise list summarizing fundamental properties and commonly used operations of matrices. We limit our material to the basics that are sufficient for this book.

(i) Zero matrix: A zero matrix has every entry equal to zero: $\mathbf{0} = [0]$, or $0 = [0]$, or explicitly

$$\mathbf{0} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{5.2}$$

(ii) Identity matrix: An identity matrix is a square matrix whose diagonal entries are all equal to one and whose off-diagonal entries are all equal to zero, and is denoted by I or $\mathbf{I}$. See an expression of an identity matrix below:

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{5.3}$$

People also use the following notation

$$\mathbf{I} = [\delta_{ij}], \tag{5.4}$$

where

$$\delta_{ij} = \begin{cases} 1 & \text{when } i = j \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

is called the Kronecker delta.

An identity matrix may be regarded as a special case of a *diagonal matrix*, which refers to any square matrix whose off-diagonal elements are zero. Hence, a diagonal matrix has the following general expression:

$$\mathbf{D} = [d_i \delta_{ij}], \tag{5.6}$$

where $d_1, d_2, \ldots, d_n$ are the n-diagonal elements. If $d_i = 1, i = 1, 2, \ldots, n$, then the diagonal matrix becomes an identity matrix.

(iii) A transpose matrix: The *transpose* of a matrix A is obtained by interchanging the rows and columns. The new matrix is denoted by $\mathbf{A}^t$. Computing the matrix transpose is very easy: simply rotate each horizontal row clockwise to a vertical column, one

row at a time. If $\mathbf{A}$ has dimension $n \times p$, its transpose has dimension $p \times n$. The elements of the transposed matrix are related to the originals by

$$(A^t)_{ij} = A_{ji} \tag{5.7}$$

For example, if

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \tag{5.8}$$

then

$$\mathbf{A}^t = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \tag{5.9}$$

If $\mathbf{A}^t = \mathbf{A}$ or $a_{ij} = a_{ji}$, then the matrix $\mathbf{A}$ is said to be *symmetric*. Of course, a symmetric matrix must be a square matrix.

(iv) Equal matrices: Two matrices $\mathbf{A}$ and $\mathbf{B}$ are equal if every pair of corresponding entries is equal, i.e., the equation $\mathbf{A} = \mathbf{B}$ means $a_{ij} = b_{ij}$ for all i and j.

(v) Matrix addition: The sum of two matrices $\mathbf{A}$ and $\mathbf{B}$ is defined by the sum of corresponding entries, i.e., $\mathbf{A} + \mathbf{B} = [a_{ij} + b_{ij}]$.

(vi) Matrix subtraction: The difference of two matrices $\mathbf{A}$ and $\mathbf{B}$ is defined by the difference of corresponding entries, i.e., $\mathbf{A} - \mathbf{B} = [a_{ij} - b_{ij}]$. The equation of $\mathbf{A} = \mathbf{B}$ is equivalent to $\mathbf{A} - \mathbf{B} = 0$.

(vii) Row vector: A row vector is of dimension p is a $1 \times p$ matrix:

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & \cdots & u_p \end{bmatrix}. \tag{5.10}$$

Matrix $\mathbf{A}_{n \times p}$ may be regarded as a stack of n row vectors $\mathbf{a}_{i:}, i = 1, 2, \ldots, n$, each of which is a p-dimensional row vector. Hence,

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_{1:} \\ \mathbf{a}_{2:} \\ \cdots \\ \mathbf{a}_{n:} \end{bmatrix}. \tag{5.11}$$

Here, : in the second position of the double-index subscript means the inclusion of all the columns.

(viii) Column vector: A column vector of dimension n is an $n \times 1$ matrix

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \cdots \\ v_n \end{bmatrix}. \tag{5.12}$$

The transpose of a column vector becomes a row vector, and vice versa.

Matrix $\mathbf{A}_{n \times p}$ may be regarded as an array of p column vectors $\mathbf{a}_{:j}, j = 1, 2, \ldots, p$, each of which is an n-dimensional column vector. Hence,

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_{:1} & \mathbf{a}_{:2} & \cdots & \mathbf{a}_{:p} \end{bmatrix}. \tag{5.13}$$

Here, : in the first position of the double-index subscript means the inclusion of all the rows.

(ix) Dot product of two vectors: Two vectors of the same dimension can form a *dot product* that is equal to the sum of the products of the corresponding entries:

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n. \tag{5.14}$$

The dot product is also called an *inner product*.

For example, if

$$\mathbf{u} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}, \tag{5.15}$$

then

$$\mathbf{u} \cdot \mathbf{v} = 1 \times 4 + 2 \times 5 + 3 \times 6 = 32. \tag{5.16}$$

The *amplitude of vector* $\mathbf{u}$ of dimension n is defined as

$$|\mathbf{u}| = \sqrt{u_1^2 + u_2^2 + \cdots + u_n^2}. \tag{5.17}$$

Sometimes, the amplitude is also called length, or Euclidean length, or magnitude. Please do not mix the concept of Euclidean length of a vector with the dimensional length of a vector. The latter means the number of entries of a vector, i.e., n.

If the Euclidean length of $\mathbf{u}$ is equal to one, we say that the $\mathbf{u}$ is a *unit vector*. If every element of $\mathbf{u}$ is zero, then we say that $\mathbf{u}$ is a *zero vector*.

By the definition of dot product, we have

$$|\mathbf{u}|^2 = \mathbf{u} \cdot \mathbf{u}. \tag{5.18}$$

If $\mathbf{u} \cdot \mathbf{v} = 0$, we say that $\mathbf{u}$ and $\mathbf{v}$ are *orthogonal*. Further, if $\mathbf{u} \cdot \mathbf{v} = 0$ and $|\mathbf{u}| = |\mathbf{v}| = 1$, then we say that $\mathbf{u}$ and $\mathbf{v}$ are *orthonormal*.

(x) Matrix multiplication: The product of matrix $\mathbf{A}_{n\times p}$ and matrix $\mathbf{B}_{p\times m}$ is an $n \times m$ matrix $\mathbf{C}_{n\times m}$ whose element c_{ij} is the dot product of the ith row vector of A and jth column vector of $\mathbf{B}$:

$$c_{ij} = \mathbf{a}_{i:} \cdot \mathbf{b}_{:j}. \tag{5.19}$$

We denote

$$\mathbf{C}_{n\times m} = \mathbf{A}_{n\times p}\mathbf{B}_{p\times m}, \tag{5.20}$$

or simply

$$\mathbf{C} = \mathbf{AB}. \tag{5.21}$$

Note that the number of columns of $\mathbf{A}$ and the number of rows of $\mathbf{B}$ must be the same before the multiplication $\mathbf{AB}$ can be made, because the dot product $\mathbf{a}_{i:} \cdot \mathbf{b}_{:j}$ requires this condition. This is referred to as the dimension-matching condition for matrix multiplication. If this condition is violated, the two matrices cannot be multiplied. For example, for the following two matrices

$$\mathbf{A}_{3\times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \\ 3 & 2 \end{bmatrix} \quad \mathbf{B}_{2\times 2} = \begin{bmatrix} 0 & -1 \\ 1 & 2 \end{bmatrix}, \tag{5.22}$$

we can compute

$$\mathbf{A}_{3\times 2}\mathbf{B}_{2\times 2} = \begin{bmatrix} 0 & -1 \\ 4 & 8 \\ 2 & 1 \end{bmatrix}. \tag{5.23}$$

However, the expression

$$\mathbf{B}_{2\times 2}\mathbf{A}_{3\times 2} \tag{5.24}$$

is not defined, because the dimensions do not match. Thus, for matrix multiplication of two matrices, their order is important. The product **BA** may not be equal to **AB** even when both are defined. That is, the commutative law does not hold for matrix multiplication.

The dot product of two vectors can be written as the product of two matrices. If both **u** and **v** are n-dimensional column vectors, then

$$\mathbf{u}\cdot\mathbf{v} = \mathbf{u}^t\mathbf{v}. \tag{5.25}$$

The right-hand side is a $1\times n$ matrix times an $n\times 1$ matrix, and the product is a 1×1 matrix, whose element is the result of the dot product. Computer programs usually calculate a dot product using this process of matrix multiplication.

A scaler can always multiply a matrix, which is defined as follows. Given a scalar c and a matrix **A**, their product is

$$c\mathbf{A} = [ca_{ij}] = \mathbf{A}c. \tag{5.26}$$

The scaler multiplication can be extended to multiple vectors

$$(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_p)$$

or matrices to form a *linear combination*:

$$\mathbf{u} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \cdots + c_p\mathbf{u}_p, \tag{5.27}$$

where $c_1, c_2, \ldots, c_p$ are coefficients of the linear combination and at least one of the coefficients is nonzero. Multivariate linear regression discussed at the end of previous chapter is a linear combination. This is a very useful mathematical expression in data science.

(xi) Matrix inversion: For a given square matrix **A**, if there is a matrix **B** such that

$$\mathbf{BA} = \mathbf{AB} = \mathbf{I}, \tag{5.28}$$

then **B** is called the *inverse matrix* of **A**, denoted by $\mathbf{A}^{-1}$, i.e.,

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}. \tag{5.29}$$

Not all the matrices have an inverse. If a matrix has an inverse, then the matrix is said to be *invertible*. Equivalently, $\mathbf{A}^{-1}$ exists.

As an example of the matrix inversion, given

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}, \tag{5.30}$$

we have

$$\mathbf{A}^{-1} = \begin{bmatrix} 2/3 & 1/3 \\ -1/3 & 1/3 \end{bmatrix}. \tag{5.31}$$

Hand calculation for the inverse of a small matrix is already very difficult, and that for the inverse of a large matrix is almost impossible. Computers can do the calculations for us, as will be shown in examples later.

According to the definition of inverse, we have the following formula for the inverse of the product of two matrices:

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}, \tag{5.32}$$

if both $\mathbf{A}$ and $\mathbf{B}$ are invertible matrices. Please note the order switch of the matrices.

With the definition of an inverse matrix, we can define the *matrix division* by

$$\mathbf{A}/\mathbf{B} = \mathbf{AB}^{-1} \tag{5.33}$$

when $\mathbf{B}^{-1}$ exists. In matrix operations, we usually do not use the concept of matrix division, but always use the matrix inverse and matrix multiplication.

(xii) More properties of the matrix transpose:

$$(\mathbf{A}^t)^t = \mathbf{A} \tag{5.34}$$

$$(\mathbf{A}+\mathbf{B})^t = \mathbf{A}^t + \mathbf{B}^t \tag{5.35}$$

$$(\mathbf{AB})^t = \mathbf{B}^t\mathbf{A}^t \tag{5.36}$$

$$(\mathbf{A}^{-1})^t = (\mathbf{A}^t)^{-1}. \tag{5.37}$$

(xiii) Orthogonal matrices: An *orthogonal matrix*[1] is one whose row vectors are orthonormal. In this case, the inverse matrix can be easily found: it is its transpose. That is, if $\mathbf{A}$ is an orthogonal matrix, then

$$\mathbf{A}^{-1} = \mathbf{A}^t. \tag{5.38}$$

The proof of this claim is very simple. The orthonormal property of the row vectors of $\mathbf{A}$ implies that

$$\mathbf{AA}^t = \mathbf{I}. \tag{5.39}$$

By the definition of matrix inverse, $\mathbf{A}^t$ is the inverse matrix of $\mathbf{A}$.

If $\mathbf{A}$ is an orthogonal matrix, its row vectors are also orthonormal. This can be proved by multiplying both sides of the above by $\mathbf{A}^t$ from the left:

$$\mathbf{A}^t\mathbf{AA}^t = \mathbf{A}^t\mathbf{I}. \tag{5.40}$$

Then multiply both sides of this equation by $(\mathbf{A}^t)^{-1}$ from the right:

$$\mathbf{A}^t\mathbf{A}(\mathbf{A}^t(\mathbf{A}^t)^{-1}) = \mathbf{A}^t\mathbf{I}(\mathbf{A}^t)^{-1}, \tag{5.41}$$

which yields

$$\mathbf{A}^t\mathbf{A} = \mathbf{I}. \tag{5.42}$$

[1] Although *orthogonal matrix* is a standard mathematical terminology, it is acceptable if you call it *orthonormal matrix*.

This implies that the column vectors of $\mathbf{A}$ are orthonormal.

As an example, the following matrix

$$\mathbf{T} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{5.43}$$

is an orthogonal matrix for any given real number θ. You can easily verify this using the trigonometrical identity $\sin^2\theta + \cos^2\theta = 1$. We thus have

$$\mathbf{T}^{-1} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \tag{5.44}$$

You can easily verify that $\mathbf{T}^{-1}\mathbf{T} = \mathbf{I}$ by hand calculation of the product of the two matrices in this equation.

5.3 Some Basic Concepts and Theories of Linear Algebra

According to `Encyclopedia.com`, "Linear algebra originated as the study of linear equations." Linear algebra deals with vectors, matrices, and vector spaces. Before the 1950s, it was part of Abstract Algebra (Tucker 1993). In 1965, the Committee on the Undergraduate Program in Mathematics, Mathematical Association of America, outlined the following topics for a stand-alone linear algebra course: linear systems, matrices, vectors, linear transformations, unitary geometry with characteristic values. The vectors and matrices have been dealt with in the previous two sections of this chapter. This section deals with linear systems of equations and linear transformations, and next with characteristic values.

5.3.1 Linear Equations

A meteorologist needs to make a decision on what instruments to order under the following constraint. She is given a budget of \$1,000 to purchase 30 instruments for her observational sites. Her supplier has two products for the instrument: the first is \$30 per set, and the second \$40 per set. She would like to buy as many of the second type of instrument as possible under the budget constraint. Then, the question is how many instruments of the second kind she can buy? This problem leads to the following linear system of two equations:

$$30x_1 + 40x_2 = 1000, \tag{5.45}$$

$$x_1 + x_2 = 30. \tag{5.46}$$

The solution to these linear equations is $x_1 = 20$ and $x_2 = 10$.

This system of linear equations can be expressed in a matrix and two vectors as follows:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.47}$$

where

$$\mathbf{A} = \begin{bmatrix} 30 & 40 \\ 1 & 1 \end{bmatrix} \tag{5.48}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{5.49}$$

$$\mathbf{b} = \begin{bmatrix} 1000 \\ 30 \end{bmatrix} \tag{5.50}$$

$$\tag{5.51}$$

Then, the solution of this system may be tightly expressed in the following way:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}. \tag{5.52}$$

This expression is convenient for mathematical proofs, but is rarely used for solving a linear system, because finding an inverse matrix is computationally costly. A way to solve a linear system is to use Gauss elimination. The corresponding computing procedure is called the row operation on a matrix. There are numerous ways of solving a linear system. Some are particularly efficient for a certain system, such as a sparse matrix or a matrix of a diagonal band of width equal 3 or 5. Efficient algorithms for a linear system, particularly an extremely large system, are forever a research topic. In this book, we use a computer to solve a linear system without studying the algorithm details. The R and Python commands are as follows.

```
solve(A, b) #This is the R code for finding x
numpy.linalg.solve(A, b) #This is the Python code
```

5.3.2 Linear Transformations

A *linear transformation* is to convert vector $\mathbf{x}_{n\times 1}$ into $\mathbf{y}_{m\times 1}$ using the multiplication of a matrix $\mathbf{T}_{m\times n}$:

$$\mathbf{y}_{m\times 1} = \mathbf{T}_{m\times n}\mathbf{x}_{n\times 1}. \tag{5.53}$$

For example, the matrix

$$\mathbf{T} = \begin{bmatrix} -0.1 & 4 \\ 0.1 & -3 \end{bmatrix} \tag{5.54}$$

transforms the vector

$$\begin{bmatrix} 1000 \\ 30 \end{bmatrix} \tag{5.55}$$

into

$$\begin{bmatrix} 20 \\ 10 \end{bmatrix}. \tag{5.56}$$

This is the solution of the linear system in the previous subsection.

Usually, the linear transformation $\mathbf{Tx}$ changes both direction and magnitude of $\mathbf{x}$. However, if $\mathbf{T}$ is an orthogonal matrix, then $\mathbf{Tx}$ does not change the magnitude of $\mathbf{x}$, and changes only the direction. This claim can be simply proved by the following formula:

$$|\mathbf{Tx}|^2 = (\mathbf{Tx})^t\mathbf{Tx} = \mathbf{x}^t\mathbf{T}^t\mathbf{Tx} = \mathbf{x}^t(\mathbf{T}^t\mathbf{T})\mathbf{x} = \mathbf{x}^t\mathbf{Ix} = |\mathbf{x}|^2. \tag{5.57}$$

Thus, if $\mathbf{T}$ is an orthogonal matrix, then $\mathbf{Tx}$ is a rotation of the vector $\mathbf{x}$.

5.3.3 Linear Independence

The vectors $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_p}$ are *linearly dependent* if no vector can be represented by a linear combination of other $p-1$ vectors in this group. Otherwise, the group of vectors are linearly dependent.

If it is not linearly independent, then there must be a vector which can be represented by the other vectors through a linear combination. Suppose this vector is $\mathbf{x_1}$, then

$$\mathbf{x_1} = d_2\mathbf{x_2} + \cdots + d_p\mathbf{x_p}, \tag{5.58}$$

where at least one of the coefficients $d_2, d_3, \cdots, d_p$ is non-zero. Thus, the linear system of equations for $c_1, c_2, c_3, \cdots, c_p$

$$c_1\mathbf{x_1} + c_2\mathbf{x_2} + \cdots + c_p\mathbf{x_p} = 0 \tag{5.59}$$

has a non-zero solution. This system can be written as a matrix form

$$\mathbf{Xc} = \mathbf{0}, \tag{5.60}$$

where column vectors $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_p}$ form the matrix $\mathbf{X}$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_p} \end{bmatrix}, \tag{5.61}$$

the unknown vector is $\mathbf{c}$

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_p \end{bmatrix}, \tag{5.62}$$

and $\mathbf{0}$ is the p-dimensional zero column vector. The solution of this matrix equation is

$$\mathbf{c} = \mathbf{X}^{-1}\mathbf{0} = \mathbf{0}. \tag{5.63}$$

However, $\mathbf{c}$ must not be zero. This contradiction implies that $\mathbf{X}^{-1}$ does not exist if the column vectors are linearly dependent. In other words, if $\mathbf{X}^{-1}$ exists, then its column vectors are linearly independent.

Consider vectors in a 3-dimensional space. Any two column vectors $\mathbf{x}_2$ and $\mathbf{x}_3$ define a plane. If $\mathbf{x}_1$ can be written as a linear combination of $\mathbf{x}_2$ and $\mathbf{x}_3$, then it must lie in the same plane. So, $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are linearly dependent. The matrix $[\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3]_{3\times 3}$ is not invertible.

5.3.4 Determinants

For a square matrix $\mathbf{A}$, a convenient notation and concept is its *determinant*. It is a scaler and is denoted by $\det[\mathbf{A}]$ or $|\mathbf{A}|$. For a 2×2 matrix

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \tag{5.64}$$

its determinant is

$$\det[\mathbf{A}] = ad - cb. \tag{5.65}$$

For a high-dimensional matrix, the determinant computation is quite complex and is computationally expensive. We usually do not need to calculate the determinant of a large matrix, say $\mathbf{A}_{172\times172}$. The computer command for computing the determinant of a small square matrix is as follows:

```
det(A) #R command for determinant
np.linalg.det(a) #Python command for determinant
```

Two 2-dimensional column vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ can span a parallelogram, whose area S is equal to the absolute value of the determinant of the matrix consisting of the two vectors $\mathbf{A} = [\mathbf{x}_1\ \mathbf{x}_2]$:

$$S = \left|\det[\mathbf{x}_1\ \mathbf{x}_2]\right|. \tag{5.66}$$

Three 3-dimensional column vectors $\mathbf{x}_1, \mathbf{x}_2$, and $\mathbf{x}_3$ can span a parallelepiped, whose volume V is equal to the absolute value of the determinant of the matrix consisting of the three vectors $\mathbf{A} = [\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3]$:

$$V = \left|\det[\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3]\right|. \tag{5.67}$$

A few commonly used properties of determinant are listed below

(a) The determinant of a diagonal matrix is the product of its diagonal elements.
(b) If a determinant has a zero row or column, the determinant is zero.
(c) The determinant does not change after a matrix transpose, i.e., $\det[\mathbf{A}^t] = \det[\mathbf{A}]$.
(d) The determinant of the product of two matrices: $\det[\mathbf{AB}] = \det[\mathbf{A}]\det[\mathbf{B}]$.
(e) The determinant of the product of a matrix with a scaler: $\det[c\mathbf{B}] = c^n \det[\mathbf{A}]$, if $\mathbf{A}$ is an $n \times n$ matrix.
(f) The determinant of an orthogonal matrix is equal to 1 or -1.

5.3.5 Rank of a Matrix

The *rank* of $\mathbf{A}$ is the greatest number of columns of the matrix that are linearly independent, and is denoted by $\mathrm{r}[\mathbf{A}]$.

For a 3×3 matrix $\mathbf{A}$, we treat each column as a 3-dimensional vector. If all three lie along a line (i.e., colinear), the rank of $\mathbf{A}$ is one. If all three of the vectors lie in a plane, but are not collinear, the rank is two. If none of the three are collinear or lie in a plane, the rank is three.

If the rank of a square matrix is less than its dimension, then at least one column can be a linear combination of other columns, which implies that the determinant vanishes. If $\mathbf{A}$ has rank r, it is possible to find r linearly independent columns, and all the other columns are linear combinations of these r independent columns.

Some properties about the matrix rank are listed below:

(a) If $\det[\mathbf{A}_{n\times n}] \neq 0$, then $\mathrm{r}[\mathbf{A}] = n$, and the matrix $\mathbf{A}_{n\times n}$ is invertible and is said to be *nonsingular*.
(b) If $\det[\mathbf{A}] = 0$, then the rank of $\mathbf{A}$ is less than n, and $\mathbf{A}$ is not invertible and is said to be *singular*.
(c) If $\mathbf{B}$ is multiplied by a nonsingular matrix $\mathbf{A}$, the product has the same rank as $\mathbf{B}$.
(d) $0 \leq \mathrm{r}[\mathbf{A}_{n\times p}] \leq \min(n, p)$.
(e) $\mathrm{r}[\mathbf{A}] = \mathrm{r}[\mathbf{A}^t]$.
(f) $\mathrm{r}[\mathbf{A}\ \mathbf{B}] \leq \min(\mathrm{r}[\mathbf{A}], \mathrm{r}[\mathbf{B}])$.
(g) $\mathrm{r}[\mathbf{A}\ \mathbf{A}^t] = \mathrm{r}[\mathbf{A}^t\mathbf{A}] = \mathrm{r}[\mathbf{A}]$.
(h) $\mathrm{r}[\mathbf{A}+\mathbf{B}] \leq \mathrm{r}[\mathbf{A}]+\mathrm{r}[\mathbf{B}]$.

Computers can easily demonstrate the matrix computations following the theories presented in this chapter so far. The computer code is below.

```
#R code: Computational examples of matrices
A = matrix(c(1,0,0,4,3, 2), nrow = 3, byrow = TRUE)
B = matrix(c(0,1,-1,2), nrow = 2) #form a matrix by columns
C = A%*%B #matrix multiplication
C
#[1,]    0   -1
#[2,]    4    8
#[3,]    2    1
t(C) # transpose matrix of C
#[1,]    0    4    2
#[2,]   -1    8    1

A = matrix(c(1, -1, 1, 2), nrow =2, byrow = TRUE)
solve(A) #compute the inverse of A
#[1,]  0.6666667 0.3333333
#[2,] -0.3333333 0.3333333
A%*%solve(A) #verify the inverse of A
#[1,] 1.000000e+00    0
#[2,] 1.110223e-16    1

#Solve linear equations
A = matrix(c(30, 40, 1, 1), nrow =2, byrow = TRUE)
b = c(1000, 30)
solve(A,b)
#[1] 20 10
solve(A)%*%b #Another way to solve the equations
det(A) #compute the determinant
#[1] -10

library(Matrix)
rankMatrix(A) #Find the rank of a matrix
#[1] 2 #rank(A) = 2
```

```
#Orthogonal matrices
p = sqrt(2)/2
Q = matrix(c(p,-p,p,p), nrow=2)
Q #is an orthogonal matrix
#           [,1]      [,2]
#[1,]  0.7071068 0.7071068
#[2,] -0.7071068 0.7071068
Q%*%t(Q) #verify Q as an orthogonal matrix
#     [,1] [,2]
#[1,]    1    0
#[2,]    0    1
det(Q) #The determinant of an orthogonal matrix is 1 or -1
#[1] 1
```

```
#Matrix multiplication
A = [[1, 0],  [0, 4],[3, 2]]
B = [[0,-1],[1,2]]
C = np.matmul(A,B) #Or C = np.dot(A,B)
print('C=', C)
#C= [[ 0 -1]
# [ 4  8]
# [ 2  1]]
print('Transpose matrix of C =', C.transpose())
#Transpose matrix of C = [[ 0  4  2]
# [-1  8  1]]

#matrix inversion
A = [[1,-1],[1,2]]
np.linalg.inv(A)# compute the inverse of A
#array([[ 0.66666667,  0.33333333],
#       [-0.33333333,  0.33333333]])

#Solve a system of linear equations
A = [[30, 40],[1, 1]]
b = [[1000],[30]]
x = np.linalg.solve(A,b)
print('x=',x)
#x= [[20.]
# [10.]]

#Compute determinant of the previous matrix A
print('Determinant det(A)=', np.linalg.det(A))
#Determinant det(A)= -10.000000000000002

#An orthogonal matrix
p = np.sqrt(2)/2
Q = [[p,p],[-p,p]]
print('Orthogonal matrix Q=', np.round(Q,2))
T = np.transpose(Q)
print('Q times transpose of Q =', np.matmul(Q,T))
print('Determinant of Q =', np.linalg.det(Q))
#Orthogonal matrix Q= [[ 0.71  0.71]
# [-0.71  0.71]]
```

```
#Q times transpose of Q =  [[1. 0.]
# [0. 1.]]
#Determinant of Q = 1.0
```

5.4 Eigenvectors and Eigenvalues

5.4.1 Definition of Eigenvectors and Eigenvalues

The linear transform $\mathbf{y} = \mathbf{A}\mathbf{u}$ usually results in $\mathbf{y}$ not parallel to $\mathbf{u}$. For example,

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{5.68}$$

The vectors $\mathbf{u} = (1,0)$ and $\mathbf{A}\mathbf{u} = (1,2)$ are not parallel in the 2-dimensional space. See the blue vectors in Figure 5.2.

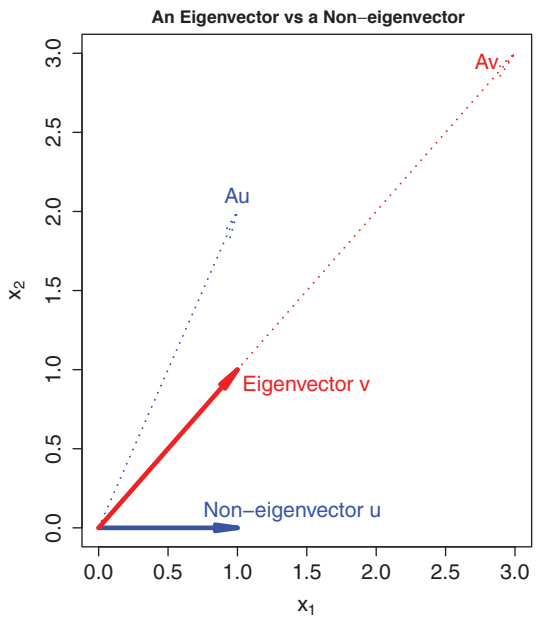

Figure 5.2 An eigenvector **v**, a non-eigenvector **u**, and their linear transforms by matrix **A**: **Av** and **Au**. Here, **Av** and **v** are parallel, and **Au** and **u** are not parallel.

However, there exist some special vectors $\mathbf{v}$ such that $\mathbf{A}\mathbf{v}$ is parallel to $\mathbf{v}$. For example, $\mathbf{v} = (1,1)$ is such a vector, since $\mathbf{A}\mathbf{v} = (3,3)$ is in the same direction as $\mathbf{v} = (1,1)$. See the red vectors in Figure 5.2. If two vectors are parallel, then one vector is a scalar multiplication of the other, e.g., $(3,3) = 3(1,1)$. We denote this scalar by λ. Thus,

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}. \tag{5.69}$$

These vectors v are special to A, maintain their own orientation when multiplied by A, and are called *eigenvectors*. Here, "eigen" is from German, meaning "self," "own," "particular," or "special."[2] The corresponding scalars λ are called *eigenvalues*. The formula (5.69) is a mathematical definition of the eigenvalue problem for matrix $\mathbf{A}$.

If $\mathbf{v}$ is an eigenvector, then its multiplication to a scalar c is also an eigenvector, since

$$\mathbf{A}(c\mathbf{v}) = c\mathbf{A}\mathbf{v} = c\lambda\mathbf{v} = \lambda(c\mathbf{v}).$$

Namely, all the vectors in the same direction $\mathbf{v}$ are also eigenvectors. The eigenvectors of length 1 are called unit eigenvectors, or unitary eigenvectors, and are unique up to a positive or negative sign. Most computer programs output unit eigenvectors. Thus, eigenvector describes a direction or an orientation. Each square matrix has its own special orientations.

The aforementioned vector

$$\mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5.70}$$

is an eigenvector that maintains its own direction after being multiplied by $\mathbf{A}$:

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{5.71}$$

Thus,

$$\mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

is an eigenvector of $\mathbf{A}$ and $\lambda = 3$ is an eigenvalue of $\mathbf{A}$. The corresponding unit eigenvector is

$$\mathbf{e} = \mathbf{v}/|\mathbf{v}| = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}.$$

Another eigenvector for the above matrix $\mathbf{A}$ is $\mathbf{v}_2 = (1, -1)$:

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -1 \times \begin{bmatrix} 1 \\ -1 \end{bmatrix}. \tag{5.72}$$

The second eigenvalue is $\lambda_2 = -1$.

The computer code for the eigenvalues and eigenvectors of the above matrix $\mathbf{A}$ is as follows.

```
#R code for eigenvectors and eigenvalues
A = matrix(c(1, 2, 2, 1), nrow=2)
eigen(A)
#$values
#[1]  3 -1
#$vectors
#[,1]       [,2]
#[1,] 0.7071068 -0.7071068
#[2,] 0.7071068  0.7071068
```

[2] The "eigen" part in the word "eigenvector" is from German or Dutch and means "self" or "own," as in "one's own." Thus, an eigenvector v is A's "own" vector. In English books, the word "eigenvector" is the standard translation of the German word "eigenvektor." The word "eigenvalue" is translated from the German word "eigenwert," as "wert" means "value." Instead of eigenvector, some English publications use "characteristic vector," which indicates "characteristics of a matrix," or "its own property of a matrix." German mathematician David Hilbert (1862–1943) was the first to use "eigenvektor," and "eigenwert" in his 1904 article about a general theory of linear integral equations.

```
#Python code for eigenvectors and eigenvalues
A = [[1,2], [2,1]]
np.linalg.eig(A)
#(array([ 3., -1.]), array([[ 0.70710678, -0.70710678],
#        [ 0.70710678,  0.70710678]]))
```

If $\mathbf{A}$ is an $N \times N$ matrix, then it has N eigenvalues and eigenvectors $(\lambda_n, v_n), n = 1, 2, \ldots, N$ for the following reason. The eigenvector $\mathbf{v}$ satisfies

$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{v} = 0. \tag{5.73}$$

The nonzero solution $\mathbf{v}$ of this equation requires that

$$\det(\mathbf{A} - \lambda \mathbf{I}) = 0. \tag{5.74}$$

Expanding the determinant out leads to an Nth degree polynomial in λ, which has exactly N roots. Some roots may be repeated and hence counted multiple times toward N. Some roots may be complex numbers, which are not discussed in this book. Each root is an eigenvalue and corresponds to an eigenvector.

This concise determinant expression is tidy and useful for mathematical proofs, but is not used as a computer algorithm for calculating eigenvalues or eigenvectors because it is computationally costly or even impossible for a large matrix. Nonetheless, some old textbooks of linear algebra defined eigenvalues using Eq. (5.74).

Figure 5.2 may be generated by the following computer code.

```
#R plot Fig. 5.2: An eigenvector v vs a non-eigenvector u
setwd('/Users/sshen/climstats')
setEPS() #Plot the figure and save the file
postscript("fig0502.eps", width = 6)
par(mar=c(4.5,4.5,2.0,0.5))
plot(9,9,
     main = 'An␣eigenvector␣vs␣a␣non-eigenvector',
     cex.axis = 1.4, cex.lab = 1.4,
     xlim = c(0,3), ylim=c(0,3),
     xlab = bquote(x[1]), ylab = bquote(x[2]))
arrows(0,0, 1,0, length = 0.25,
       angle = 8, lwd = 5, col = 'blue')
arrows(0,0, 1,2, length = 0.3,
       angle = 8, lwd = 2, col = 'blue',  lty = 3)
arrows(0,0, 1,1, length = 0.25,
       angle = 8, lwd = 5, col='red')
arrows(0,0, 3,3, length = 0.3,
       angle = 8, lwd = 2, col='red', lty = 3)
text(1.4,0.1, 'Non-eigenvector␣u', cex =1.4, col = 'blue')
text(1.0,2.1, 'Au', cex =1.4, col = 'blue')
text(1.5,0.9, 'Eigenvector␣v', cex =1.4, col = 'red')
text(2.8, 2.95, 'Av', cex =1.4, col = 'red')
dev.off()
```

```
#Python plot Fig. 5.2: An eigenvector vs a non-eigenvector
import matplotlib.patches as patches
fig = plt.figure(figsize = (12,12))

plt.axes().set_xlim(-0.1,3.1)
plt.axes().set_ylim(-0.1,3.1)
plt.axes().set_aspect(1)
style = "Simple, tail_width=0.5,head_width=8,head_length=18"
kw1 = dict(arrowstyle=style, color="blue")
kw2 = dict(arrowstyle=style, color="red")

a1 = patches.FancyArrowPatch((0,0), (1,0), **kw1,
                             linewidth =5)
a2 = patches.FancyArrowPatch((0, 0), (1,2),**kw1)
a3 = patches.FancyArrowPatch((0, 0), (1,1), **kw2,
                             linewidth = 5)
a4 = patches.FancyArrowPatch((0, 0), (3,3),**kw2)
for a in [a1, a2, a3, a4]:
    plt.gca().add_patch(a)
plt.title('An eigenvector v vs a non-eigenvector u')
plt.xlabel(r'$x_1$', fontsize = 25)
plt.ylabel(r'$x_2$', fontsize = 25)
plt.text(0.6, 0.1, 'Non-eigenvector u',
         color = 'blue', fontsize =25)
plt.text(0.8, 2.0, 'Au',
         color = 'blue', fontsize =25)
plt.text(1.03,0.85, 'Eigenvector v',
         color = 'red', fontsize =25)
plt.text(2.7, 2.9, 'Av',
         color = 'red', fontsize =25)
plt.show()
```

5.4.2 Properties of Eigenvectors and Eigenvalues for a Symmetric Matrix

A covariance matrix or a correlation matrix is a symmetric matrix and is often used in climate science. For a symmetric matrix $\mathbf{A}$, its eigenvalues and eigenvectors have the following properties:

(a) Eigenvalues of a symmetric matrix are real numbers.

(b) The n different unit eigenvectors of a symmetric matrix $\mathbf{A}_{n\times n}$ are independent and form an orthonormal set, i.e., $\mathbf{e}^{(\ell')} \cdot \mathbf{e}^{(\ell)} = \delta_{\ell\ell'}$, where $\mathbf{e}^{(\ell')}$ and $\mathbf{e}^{(\ell)}$ are any two different unit eigenvectors. We can use these unit vectors to express any n-dimensional vector using a linear combination:

$$\mathbf{x} = c_1\mathbf{e}^{(1)} + c_2\mathbf{e}^{(2)} + \cdots + c_n\mathbf{e}^{(n)}, \tag{5.75}$$

where $c_1, c_2, \ldots, c_n$ are scaler coefficients of the linear combination.

(c) If all the eigenvalues of $\mathbf{A}_{n\times n}$ are positive, then the *quadratic form*

$$Q(\mathbf{x}) = \mathbf{x}^t\mathbf{A}\mathbf{x} = \sum_{i,j=1}^{n} a_{ij}x_i x_j \tag{5.76}$$

is also a positive scaler for any nonzero vector $\mathbf{x}$. A quadratic form may be used to express kinetic energy or total variance in climate science. The kinetic energy in climate science is always positive. For all unit vectors $\mathbf{x}$, the maximum quadratic form is equal to the largest eigenvalue of $\mathbf{A}$. The maximum is achieved when $\mathbf{x}$ is the corresponding eigenvector of $\mathbf{A}$.

(d) The rank of matrix $\mathbf{A}$ is equal to the number of nonzero eigenvalues, where the multiplicity of repeated eigenvalues is counted.

(e) Eigenvalues of a diagonal matrix are equal to the diagonal elements.

(f) For a symmetric matrix $\mathbf{A}_{n\times n}$, its n unit column eigenvectors form an orthogonal matrix $\mathbf{Q}_{n\times n}$ such that $\mathbf{A}_{n\times n}$ can be diagonalized by $\mathbf{Q}_{n\times n}$ in the following way:

$$\mathbf{Q}^t_{n\times n}\mathbf{A}_{n\times n}\mathbf{Q}_{n\times n} = \mathbf{D}, \tag{5.77}$$

where $\mathbf{D}$ is a diagonal matrix whose diagonal elements are eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of $\mathbf{A}$. Further, the column vectors of $\mathbf{Q}$ are the unit eigenvectors of $\mathbf{A}$. This is a matrix diagonalization process.

For the symmetric matrix $\mathbf{A}$ in Eq. (5.68),

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \tag{5.78}$$

its eigenvalues and eigenvectors are

$$\lambda_1 = 3, \qquad \lambda_2 = -1, \tag{5.79}$$

$$\mathbf{v}_1 = \begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}. \tag{5.80}$$

Thus, the orthogonal matrix $\mathbf{Q}$ and the diagonal matrix $\mathbf{D}$ are as follows:

$$\mathbf{Q} = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} 3 & 0 \\ 0 & -1 \end{bmatrix}. \tag{5.81}$$

With $\mathbf{Q}$, $\mathbf{A}$, and $\mathbf{D}$, you can easily verify Eq. (5.77) by hand calculation or by computer coding.

Equation (5.77) can be written in the following way:

$$\mathbf{A}_{n\times n} = \mathbf{Q}_{n\times n}\mathbf{D}_{n\times n}\mathbf{Q}^t_{n\times n} \tag{5.82}$$

or

$$\mathbf{A}_{n\times n} = \sum_{k=1}^{n} \lambda_k \left(\mathbf{q}^{(k)}\right)_{n\times 1} \left((\mathbf{q}^{(k)})^t\right)_{1\times n}. \tag{5.83}$$

This is a process of matrix decomposition by orthogonal matrices or by eigenvectors. Further, if all the eigenvalues are nonnegative, then the matrix is said to be *positive semi-definite*, and the formula can be written as

$$\mathbf{A}_{n\times n} = \sum_{k=1}^{n} \left(\mathbf{v}^{(k)}\right)_{n\times 1} \left((\mathbf{v}^{(k)})^t\right)_{1\times n}, \tag{5.84}$$

where $\mathbf{v}^{(k)} = \sqrt{\lambda_k}\mathbf{q}^{(k)} (k = 1, 2, \ldots, n)$. The sample covariance matrix in climate science satisfies the positive eigenvalue assumption, and will be discussed in more details in the next chapter. Equation (5.84) means that a positive semi-definite symmetric matrix $\mathbf{A}_{n\times n}$ can be decomposed into a sum of n outer products of eigenvectors.

Matrix $\mathbf{A}$,

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \tag{5.85}$$

is not positive semi-definite since its second eigenvalue is -1, but matrix $\mathbf{C}$,

$$\mathbf{C} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \tag{5.86}$$

is positive semi-definite since its eigenvalues are 3 and 1, which are positive. The matrix $\mathbf{C}$ is actually positive definite. The eigenvectors are the same those of $\mathbf{A}$. Thus,

$$\mathbf{C} = \mathbf{Q}\mathbf{D}_c\mathbf{Q}^t, \tag{5.87}$$

where $\mathbf{Q}$ is given by Eq. (5.81), and $\mathbf{D}_c$ is

$$\mathbf{D}_c = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.88}$$

This can be verified by the following computer code.

```
#Verify diagonalization and decomposition: R code
C = matrix(c(2,1,1,2), nrow = 2)
eigen(C)
#$values
#[1] 3 1
#$vectors
#     [,1]       [,2]
#[1,] 0.7071068 -0.7071068
#[2,] 0.7071068  0.7071068
Q = eigen(C)$vectors
D = t(Q)%*%C%*%Q #Matrix diagonalization
D
#[1,]    3    0
#[2,]    0    1
Q%*%D%*%t(Q) #Matrix decomposition
#[1,]    2    1
#[2,]    1    2
D[1,1]*Q[,1]%*%t(Q[,1]) + D[2,2]*Q[,2]%*%t(Q[,2])
#[1,]    2    1
#[2,]    1    2
```

```
#Verify diagonalization and decomposition: Python code
C = [[2,1],[1,2]]
valC, Q = np.linalg.eig(C)
print('eigenvalues␣of␣C␣=', valC)
#eigenvalues of C = [3. 1.]
print('eigenvectors␣of␣C␣=', Q)
#eigenvectors of C = [[ 0.70710678 -0.70710678]
# [ 0.70710678  0.70710678]]
D = Q.transpose(1,0).dot(C).dot(Q)
print('D␣=␣', D)
#D =  [[3. 0.]
# [0. 1.]]
# Matrix C is decomposed into three matrices: C = Q D Q'
Q.dot(D).dot(Q.transpose(1,0))
#array([[2., 1.],
#       [1., 2.]])
D[0][0]*np.outer(Q[:][0],Q.transpose()[:][0]) + \
D[1][1]*np.outer(Q[:][1],Q.transpose()[:][1])
#array([[1., 2.],
#       [2., 1.]]) #matrix C is recovered from vectors
```

5.5 Singular Value Decomposition

The previous section shows an eigenvector-eigenvalue decomposition of a symmetric square matrix. A similar decomposition can be made for a rectangular matrix. The decomposition using unit eigenvectors and eigenvalues for a general rectangular matrix is called the *singular value decomposition* (SVD). Singular value is another name for eigenvalue. Although the basic mathematical theory of SVD was developed almost 200 years ago (Stewart 1993), the modern algorithm of efficient SVD computing was only developed by Gene H. Golub (1932–2007) and his colleagues in the 1970s. Now, SVD has become an important data analysis tool for every field: climate science is not an exception. A space-time climate data matrix is often a rectangular matrix, since the number of sites is not likely to be equal to the number of temporal observations at those sites. We may denote a space-time climate data matrix by $\mathbf{X}_{n\times m}$, where n is the number of sites, and m is the total number of temporal observations. For $\mathbf{X}_{n\times m}$, we can also interpret n as the number of grid boxes of a climate model output, and m as the number of time steps in the output.

5.5.1 SVD Formula and a Simple SVD Example

If $m \leq n$, then matrix $\mathbf{A}_{n\times m}$ has the following SVD decomposition:

$$\mathbf{A}_{n\times m} = \mathbf{U}_{n\times m}\mathbf{D}_{m\times m}(\mathbf{V}^t)_{m\times m}. \tag{5.89}$$

Here, $\mathbf{U}$ may be interpreted as a spatial matrix, consisting of m spatial orthonormal column vectors that are unit eigenvectors of $\mathbf{A}\mathbf{A}^t$; $\mathbf{V}$ may be interpreted as a temporal matrix, consisting of m temporal orthonormal column vectors that are unit eigenvectors of $\mathbf{A}^t\mathbf{A}$; and $\mathbf{D}$ is a diagonal matrix whose elements are the square root of the eigenvalues of $\mathbf{A}\mathbf{A}^t$ and may be interpreted as standard deviations. Figure 5.3 may help you understand this formula. The diagonal elements of $\mathbf{D}$ are called singular values, and the column vectors of $\mathbf{U}$ and $\mathbf{V}$ are called singular vectors. Here, the word "singular" may be understood as special (or opposite to "general"), or distinguished or out of ordinary.

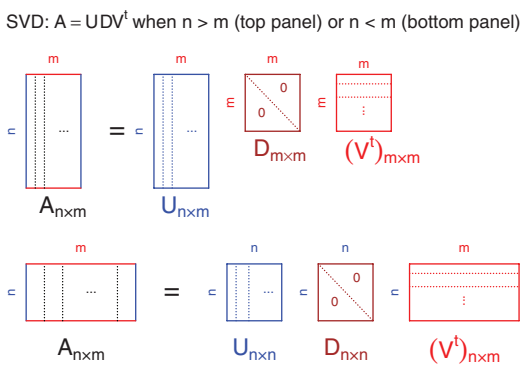

Figure 5.3 Schematic diagrams of SVD: $A = UDV^t$. Case 1: $n > m$ (top panel). Case 2: $n < m$ (bottom panel).

If $m \geq n$, then matrix $\mathbf{A}_{n\times m}$ has the following SVD decomposition:

$$\mathbf{A}_{n\times m} = \mathbf{U}_{n\times n}\mathbf{D}_{n\times n}(\mathbf{V}^t)_{n\times m}. \tag{5.90}$$

The following computer code shows a simple SVD example of a 2×3 matrix.

```
#SVD example for a 2-by-3 matrix: R code
A=matrix(c(-1,1,0,2,-2,3),nrow=2)
A #Show the 2-by-3 matrix
#      [,1] [,2] [,3]
#[1,]   -1    0   -2
#[2,]    1    2    3
svdA=svd(A) #Compute the SVD of A and put the results in svdA
svdA #Show SVD results: d, U, and V
round(svdA$d, digits=2) #Show only the singular values
#[1] 4.22 1.09
round(svdA$u, digits=2) #Show only matrix U
#      [,1] [,2]
#[1,] -0.48 0.88
#[2,]  0.88 0.48
round(svdA$v, digits=2)#Show only matrix V
#      [,1]  [,2]
#[1,] 0.32 -0.37
#[2,] 0.42  0.88
#[3,] 0.85 -0.29
sqrt(eigen(A%*%t(A))$values)
#[1] 4.221571 1.085514
```

```
#SVD example for a 2-by-3 matrix: Python code
A = [[-1,0, -2],[1,2,3]]
UsvdA, DsvdA, VsvdA = np.linalg.svd(A)
print('Singular values=', np.round(DsvdA,2))
#Singular values=  [4.22 1.09]
print('Spatial singular vectors=', np.round(UsvdA,2))
#Spatial singular vectors=  [[-0.48  0.88]
# [ 0.88  0.48]]
print('Temporal singular vectors=', np.round(VsvdA,2))
#Temporal singular vectors=  [[ 0.32  0.42  0.85]
# [-0.37  0.88 -0.29]
# [-0.87 -0.22  0.44]]

B = np.array(A)
C = np.matmul(B, B.T) #B times B transpose
valC, vecC = np.linalg.eig(C)
np.sqrt(valC)
#array([1.0855144 , 4.22157062])
```

The computer code shows the following SVD results expressed in following mathematical formulas:
(a) The vector form:

$$\begin{aligned} A &= \begin{bmatrix} -1 & 0 & -2 \\ 1 & 2 & 3 \end{bmatrix} \\ &= 4.22 \begin{bmatrix} -0.48 \\ 0.88 \end{bmatrix} \times \begin{bmatrix} 0.32 & 0.42 & 0.85 \end{bmatrix} + \\ &\quad 1.09 \begin{bmatrix} 0.88 \\ -0.48 \end{bmatrix} \times \begin{bmatrix} -0.37 & 0.88 & -0.29 \end{bmatrix} \end{aligned} \tag{5.91}$$

and
(b) The matrix form:

$$A = \begin{bmatrix} -1 & 0 & -2 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} -0.48 & 0.88 \\ 0.88 & 0.48 \end{bmatrix} \begin{bmatrix} 4.22 & 0 \\ 0 & 1.09 \end{bmatrix} \begin{bmatrix} 0.32 & 0.42 & 0.85 \\ -0.37 & 0.88 & -0.29 \end{bmatrix} \tag{5.92}$$

If we use only the first singular vectors to approximate A from the two triplets of singular vectors and singular values, the result is as follows.

```
#Data reconstruction by singular vectors: R code
round(svdA$d[1]*svdA$u[,1]%*%t(svdA$v[,1]),
      digits=1)
#      [,1] [,2] [,3]
#[1,] -0.7 -0.8 -1.7
#[2,]  1.2  1.5  3.2
```

It is quite close to A.

```
#Data reconstruction by singular vectors: Python code
np.round(DsvdA[0]*np.outer(UsvdA[:][0], VsvdA[:][0]),1)
#array([[-0.7, -0.8, -1.7],
#       [ 1.2,  1.5,  3.2]])
```

If we use both singular vectors to reconstruct A, then the reconstruction is exact without errors, as expected.

```
round(svdA$d[1]*svdA$u[,1]%*%t(svdA$v[,1]) +
  svdA$d[2]*svdA$u[,2]%*%t(svdA$v[,2]),
  digits =2)
#     [,1] [,2] [,3]
#[1,]   -1    0   -2
#[2,]    1    2    3
```

```
A1 = DsvdA[0]*np.outer(UsvdA[:][0], VsvdA[:][0])
A2 = DsvdA[1]*np.outer(UsvdA[:][1], VsvdA[:][1])
np.round(A1 + A2, 2)
#array([[-1.,  0., -2.],
#       [ 1.,  2.,  3.]])
```

Figure 5.3 for the schematic diagram of SVD may be plotted by the following computer code.

```
#R plot Fig. 5.3:  Schematic diagram of SVD
setwd('/Users/sshen/climstats')
setEPS() #Plot the figure and save the file
postscript("fig0503.eps", width = 11)
par(mar=c(0,0,0,0))
plot(200, axes = FALSE,
     xlab = "", ylab = "",
     xlim = c(-3,28), ylim = c(-3,16))
text(13,15.5, cex=2.2,
     bquote("SVD:" ~ A==UDV^t~ "when␣n␣>␣m␣or␣n␣<␣m"))
#Space-time data matrix A when n>m
segments(x0 = c(0,0,3,3),
         y0 = c(6,12,12,6) +1,
         x1 = c(0,3,3,0),
         y1 = c(12,12,6,6) +1,
         col = c('blue','red','blue','red'),lwd =3)
segments(x0 = c(0.5,1.0),
         y0 = c(6,6)+1,
         x1 = c(0.5,1.0),
         y1 = c(12,12)+1,
         lwd =1.3, lty = 3)
text(-.8, 9+1, 'n', srt=90, col ='blue', cex = 1.4)
text(1.5, 12.8+1, 'm', col = 'red',  cex = 1.4)
text(2.0, 9+1, '...',  cex = 1.4)
text(2, 5+1, bquote(A[n%*%m]),  cex = 2.5)
text(5, 9+1, '=',  cex = 3)
```

```
#Spatial matrix U
segments(x0 = c(7,7,10,10),
         y0 = c(6,12,12,6)+1,
         x1 = c(7,10,10,7),
         y1 = c(12,12,6,6)+1,
         col = c('blue','blue','blue','blue'), lwd =3)
segments(x0 = c(7.5,8),
         y0 = c(6,6)+1,
         x1 = c(7.5,8),
         y1 = c(12,12)+1,
         lwd =1.3, lty = 3, col = 'blue')
text(6.2, 9+1, 'n', srt=90, col ='blue', cex = 1.4)
text(8.5, 12.8+1, 'm', col = 'red',  cex = 1.4)
text(9, 9+1, '...',  cex = 1.4, col='blue')
text(8.7, 5.0+1, bquote(U[n%*%m]),  cex = 2.5, col= 'blue')
#Singular value diagonal matrix D
segments(x0 = c(12,12,15,15),
         y0 = c(9,12,12,9)+1,
         x1 = c(12,15,15,12),
         y1 = c(12,12,9,9)+1,
         col = c('brown','brown','brown','brown'), lwd =3)
segments(x0 = 12, y0 = 12+1, x1 = 15, y1 = 9+1, lty=3,
         col = c('brown'), lwd =1.3)#diagonal line
text(11.2, 10.5+1, 'm', srt=90, col ='red', cex = 1.4)
text(13.5, 12.8+1, 'm', col = 'red',  cex = 1.4)
text(14.1, 11.3+1, '0', col = 'brown',  cex = 1.4)
text(12.9, 10.0+1, '0', col = 'brown',  cex = 1.4)
text(13.9, 8.0+1, bquote(D[m%*%m]),  cex = 2.5, col='brown')
#Temporal matrix V
segments(x0 = c(17,17,20,20),
         y0 = c(9,12,12,9)+1,
         x1 = c(17,20,20,17),
         y1 = c(12,12,9,9)+1,
         col = c('red','red','red','red'), lwd =3)
segments(x0 = c(17,17),
         y0 = c(11.5,10.8)+1,
         x1 = c(20,20),
         y1 = c(11.5,10.8)+1,
         col = c('red','red'), lty=3, lwd =1.3)
text(16.2, 10.5+1, 'm', srt=90, col ='red', cex = 1.4)
text(18.5, 12.5+1, 'm', col = 'red',  cex = 1.4)
text(19.5, 8+1, bquote((V^t)[m%*%m]),  cex = 2.5, col='red')
text(18.5, 10+1, '...',  col='red', srt=90, cex =1.4)
#Space-time data matrix B when n < m
segments(x0 = c(0,0,6,6),
         y0 = c(0,3,3,0),
         x1 = c(0,6,6,0),
         y1 = c(3,3,0,0),
         col = c('blue','red','blue','red'), lwd =3)

segments(x0 = c(1,2,5),
         y0 = c(0,0,0),
         x1 = c(1,2,5),
         y1 = c(3,3,3),
         lwd =1.3, lty = 3)
text(-0.8, 1.5, 'n', srt=90, col ='blue', cex = 1.4)
text(3, 3.8, 'm', col = 'red',  cex = 1.4)
```

```
text(3.5, 1.5, '...',  cex = 1.4)
text(3, -1.5, bquote(A[n%*%m]),  cex = 2.5)
text(8, 1.5, '=',  cex = 3)
#Spatial matrix U
segments(x0 = c(11,11,14,14),
         y0 = c(0,3,3,0),
         x1 = c(11,14,14,11),
         y1 = c(3,3,0,0),
         col = c('blue','blue','blue','blue'), lwd =3)
segments(x0 = c(11.5,12.2),
         y0 = c(0,0),
         x1 = c(11.5,12.2),
         y1 = c(3,3),
         lwd =1.3, lty = 3, col = 'blue')
text(10.2, 1.5, 'n', srt=90, col ='blue', cex = 1.4)
text(12.5, 3.8, 'n', col = 'blue',  cex = 1.4)
text(13.2, 1.5, '...',  cex = 1.4, col='blue')
text(12.5, -1.5, bquote(U[n%*%n]),  cex = 2.5, col= 'blue')
#Singular value diagonal matrix D
segments(x0 = c(16,16,19,19),
         y0 = c(0,3,3,0),
         x1 = c(16,19,19,16),
         y1 = c(3,3,0,0),
         col = c('brown','brown','brown','brown'), lwd =3)
segments(x0 = 16, y0 = 3, x1 = 19, y1 = 0, lty=3,
         col = c('brown'), lwd =1.3)#diagonal line
text(15.2, 1.5, 'n', srt=90, col ='blue', cex = 1.4)
text(17.5, 3.8, 'n', col = 'blue',  cex = 1.4)
text(18.1, 2.3, '0', col = 'brown',  cex = 1.4)
text(16.9, 1.0, '0', col = 'brown',  cex = 1.4)
text(17.5, -1.5, bquote(D[n%*%n]),  cex = 2.5, col='brown')
#Temporal matrix V
segments(x0 = c(21,21,27,27),
         y0 = c(0,3,3,0),
         x1 = c(21,27,27,21),
         y1 = c(3,3,0,0),
         col = c('red','red','red','red'),
         lwd =3)
segments(x0 = c(21,21),
         y0 = c(2.5,1.8),
         x1 = c(27,27),
         y1 = c(2.5,1.8),
         col = c('red','red'), lty=3, lwd =1.3)
text(20.2, 1.5, 'n', srt=90, col ='blue', cex = 1.4)
text(24, 3.8, 'm', col = 'red',  cex = 1.4)
text(24, -1.5, bquote((V^t)[n%*%m]),  cex = 2.5, col='red')
text(24, 1, '...',  col='red', srt=90, cex =1.4)
dev.off()
```

```
#Python plot Fig. 5.3: Schematic diagram of SVD
import matplotlib.patches as patches
import numpy as np
import pylab as pl
```

```
from matplotlib import collections  as mc
lines = [[(0, 7), (0, 13)], [(0, 13), (3, 13)],
         [(3, 13), (3, 7)], [(3, 7), (0,7)]]
c = np.array(['b', 'r', 'b', 'r'])
lc = mc.LineCollection(lines, colors=c, linewidths=3)
fig, ax = pl.subplots()
ax.set_xlim([-3, 28])
ax.set_ylim([-3, 16])
ax.add_collection(lc)
ax.margins(0.1)
plt.plot([0.5,0.5], [7, 13],
         linestyle='dotted', color = 'k')
plt.plot([1, 1], [7, 13],
        linestyle='dotted', color = 'k')
plt.text(13, 15.5,
         r'SVD:␣␣$A==UDV^t$␣when␣n␣>␣m␣or␣n␣<␣m',
        fontsize = 30)
plt.text(-1.2, 10, 'n', color = 'blue',
           fontsize =25, rotation=90)
plt.text(1.1, 13.3, 'm', color = 'red',
           fontsize =25, rotation=0)
plt.text(0.0, 5.5, r'$A_{n\times␣m}$',
           fontsize =35, rotation=0)
plt.text(1.5, 10, '...',
           fontsize =25, rotation=0)
plt.axis('off')
plt.show()
```

This Python code generates the top-left rectangular box in Figure 5.3. The remaining code for other boxes is highly repetitive and can be found from the book website.

5.6 SVD for the Standardized Sea Level Pressure Data of Tahiti and Darwin

The Southern Oscillation Index (SOI) is an indicator for El Niño or La Niña. It is computed as the difference of sea level pressure (SLP) of Tahiti (17.75° S, 149.42° W) minus that of Darwin (12.46° S, 130.84° E). An SVD analysis of the SLP data can substantiate this calculation formula.

The following shows the data matrix of the standardized SLP anomalies of Tahiti and Darwin from 2009 to 2015, and its SVD.

```
#R SVD analysis for the weighted SOI from SLP data
setwd("/Users/sshen/climmath")
Pda<-read.table("data/PSTANDdarwin.txt", header=F)
dim(Pda)
#[1] 65 13 #Monthly Darwin data from 1951-2015
pdaDec<-Pda[,13] #Darwin Dec standardized SLP anomalies data
Pta<-read.table("data/PSTANDtahiti.txt", header=F)
ptaDec=Pta[,13] #Tahiti Dec standardized SLP anomalies
```

```
ptada1 = cbind(pdaDec, ptaDec) #space-time data matrix

#Space-time data format
ptada = t(ptada1[59:65,]) #2009-2015 data
colnames(ptada)<-2009:2015
rownames(ptada)<-c("Darwin", "Tahiti")
ptada #6 year of data for two stations
#        2009 2010 2011 2012 2013 2014 2015
#Darwin   0.5 -2.3 -2.2  0.3  0.3  0.1 -0.4
#Tahiti -0.7  2.5  1.9 -0.7  0.4 -0.8 -1.3
svdptd = svd(ptada) #SVD for the 2-by-6 matrix
U=round(svdptd$u, digits=2)
U
#[1,] -0.66 0.75
#[2,]  0.75 0.66
D=round(diag(svdptd$d), digits=2)
D
#[1,]  4.7 0.00
#[2,]  0.0 1.42
V =round(svdptd$v, digits=2)
t(V)
#[1,] -0.18  0.72  0.61 -0.15 0.02 -0.14 -0.15
#[2,] -0.06 -0.06 -0.28 -0.17 0.34 -0.32 -0.82
```

```
#Python SVD analysis of the Darwin and Tahiti SLP data
import os
os.chdir("/Users/sshen/climstats")
PDA = np.array(read_table("data/PSTANDdarwin.txt", \
                          header = None, delimiter = "\s+"))
PTA = np.array(read_table("data/PSTANDtahiti.txt", \
                          header = None, delimiter = "\s+"))
pdata = np.stack([PDA[58:65,12], PTA[58:65,12]], axis=0)
print('The␣Darwin␣and␣Tahiti␣SLP␣data␣2009-2015␣=', pdata)
#The Darwin and Tahiti Standardized SLP anomalies =
#[[ 0.5 -2.3 -2.2  0.3  0.3  0.1 -0.4]
# [-0.7  2.5  1.9 -0.7  0.4 -0.8 -1.3]]
u, d, v = np.linalg.svd(pdata)
print('Spatial␣singular␣vectors␣EOFs␣U␣=', np.round(u,2))
#Spatial singular vectors EOFs U = [[-0.66  0.75]
# [ 0.75  0.66]]
print('Diagonal␣matrix␣D␣=', np.round(np.diag(d),2))
#Diagonal matrix D = [[4.7  0.  ]
# [0.   1.42]]
print('Temporal␣singular␣vectors␣PCs␣V=', np.round(v,2))
#Temporal singular vectors PCs V=
#[[-0.18  0.72  0.61 -0.15  0.02 -0.14 -0.15]
# [-0.06 -0.06 -0.28 -0.17  0.34 -0.32 -0.82] ...]
```

One can verify that

$$\mathbf{UDV}^t \tag{5.93}$$

approximately recovers the original data matrix.

The first column vector $(-0.66, 0.75)$ of the spatial pattern matrix $\mathbf{U}$ may be interpreted to be associated with the SOI, which puts a negative weight -0.66 on Darwin, and a positive weight 0.75 on Tahiti. The weighted sum is approximately equal to the difference of Tahiti's SLP minus that of Darwin, which is the definition of SOI. The index measures large-scale ENSO dynamics of the tropical Pacific (Trenberth 2020). The magnitude of the vector $(-0.66, 0.75)$ is approximately 1, because U is a unitary matrix. The corresponding first temporal singular vector has a distinctly large value 0.72 in December 2010, which was a strong La Niña month. In this month, the Darwin had a strong negative SLP anomaly, while Tahiti had a strong positive SLP anomaly. This situation enhanced the easterly trade winds in the tropical Pacific and caused abnormally high precipitation in Australia in the 2010–2011 La Niña period.

The second column vector $(0.75, 0.66)$ of the spatial pattern matrix $\mathbf{U}$ also has climate implications. The weighted sum with two positive equal weights measures the small scales tropical Pacific dynamics (Trenberth 2020).

5.7 Chapter Summary

A matrix can be regarded as a 2-dimensional $n \times m$ rectangular array of numbers or symbols, or as m column vectors, or as n row vectors. This may be interpreted as climate data at n locations with m time steps. Many mathematical properties of a matrix of climate data have climate interpretations. For example, SVD decomposes a space-time climate data matrix into an orthogonal spatial pattern matrix, an orthogonal temporal pattern matrix, and a diagonal "energy-level" matrix that measures the standard deviation of the temporal pattern:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^t. \tag{5.94}$$

The column vectors of the spatial matrix $\mathbf{U}$ are spatial singular vectors, also called EOFs, while those of the temporal matrix $\mathbf{V}$ are temporal singular vectors, also called PCs. The first few EOFs often have climate dynamic interpretations, such as El Niño Southern Oscillation (ENSO). If EOF1 corresponds to El Niño and shows some typical ENSO properties, such as the opposite signs of SLP anomalies of Darwin and Tahiti, then PC1 shows a temporal pattern, e.g., the extreme values of PC1 indicating both the occurrence time and the strength of El Niño. The diagonal elements of matrix $\mathbf{D}$ are singular values, also known as eigenvalues.

An eigenvector $\mathbf{v}$ of a square matrix $\mathbf{C}$ is a special vector such that $\mathbf{C}$'s action on $\mathbf{v}$ does not change its orientation, i.e., $\mathbf{C}\mathbf{v}$ is parallel to $\mathbf{v}$. This statement implies the existence of a scaler λ, called eigenvalue (also known as singular value or characteristic value), such that

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v}. \tag{5.95}$$

We have also discussed the matrix method of solving a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.96}$$

linear independence of vectors, linear transform, and other basic matrix methods. These methods are useful for the chapters on covariance, EOFs, spectral analysis, regression analysis, and machine learning. You may focus on the computing methods and computer code of the relevant methods. The mathematical proofs of this chapter, although helpful for exploring new mathematical methods, are not necessarily needed to read the other chapters of this book. If you are interested in an in-depth mathematical exploration of matrix theory, you may wish read the books by Horn and Johnson (1985) and Strang (2016).

References and Further Reading

[1] G. H. Golub and C. Reinsch, 1970: Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14, 403–420.

This seminal paper established an important method, known as the Golub–Reinsch algorithm, to compute the eigenvalues of a covariance matrix from a space-time matrix A without actually first computing the covariance matrix AA^t. This algorithm makes the SVD computation very efficient, which helps scientists consider SVD as a genuine linear algebra method, not a traditionally regarded statistical method based on a covariance matrix.

[2] R. A. Horn and C. R. Johnson, 1985: *Matrix Analysis*. Cambridge University Press.

This is a comprehensive book on matrix theory and is a good reference for a researcher in climate statistics. It assumes knowledge of a first course of linear algebra.

[3] G. Strang, 2016: *Introduction to Linear Algebra*. 5th ed., Wellesley-Cambridge Press.

Gilbert Strang (1934–) is an American mathematician and educator. His textbooks and pedagogy have been internationally influential. This text is one of the very few basic linear algebra books that includes excellent materials on SVD, probability, and statistics.

[4] G. W. Stewart, 1993: On the early history of the singular value decomposition. *SIAM Review*, 35, 551–566.

This paper describes the contributions from five mathematicians in the period of 1814–1955 to the development of the basic SVD theory.

[5] K. Trenberth, and National Center for Atmospheric Research Staff (eds.), 2020: The Climate Data Guide: Southern Oscillation Indices: Signal, Noise and Tahiti/Darwin SLP (SOI).

https://climatedataguide.ucar.edu/climate-data/southern-oscillation-indices-signal-noise-and-tahitidarwin-slp-soi

This site describes the optimal indices for large- and small-scale dynamics.

[6] A. Tucker, 1993: The growing importance of linear algebra in undergraduate mathematics. *College Mathematics Journal*, 24, 3–9.

This paper describes the historical development of linear algebra, such as the term "matrix" being coined by J. J. Sylvester in 1848, and pointed out that "tools of linear algebra find use in almost all academic fields and throughout modern society." The use of linear algebra in the big data era is now even more popular.

Exercises

5.1 Write a computer code to

(a) Read the NOAAGlobalTemp data file, and

(b) Generate a 4×8 space-time data matrix for the December mean surface air temperature anomaly data of four grid boxes and eight years. *Hint: You may find the NOAA Global Surface Temperature (NOAAGlobalTemp) dataset online. You can use either netCDF format or CSV format.*

5.2 Write a computer code to find the inverse of the following matrix.

```
#      [,1] [,2] [,3]
#[1,]  1.7 -0.7  1.3
#[2,] -1.6 -1.4  0.4
#[3,] -1.5 -0.3  0.6
```

5.3 Write a computer code to solve the following linear system of equations:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.97}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}. \tag{5.98}$$

5.4 The following equation

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.99}$$

has infinitely many solutions, and cannot be directly solved by a simple computer command, such as `solve(A, b)`.

(a) Show that the three row vectors of the coefficient matrix are not linearly independent.
(b) Because of the dependence, the linear system has only two independent equations. Thus, reduce the linear system to two equations by treating x_3 as an arbitrary value while treating x_1 and x_2 as variables.
(c) Solve the two equations for x_1 and x_2 and express them in terms of x_3. The infinite possibilities of x_3 imply infinitely many solutions of the original system.

5.5 Ethane is a gas similar to the greenhouse gas methane and can burn with oxygen to form carbon dioxide and water:

$$C_2H_6 + O_2 \longrightarrow CO_2 + H_2O. \tag{5.100}$$

Given two ethane molecules, how many molecules of oxygen, carbon dioxide and water are required for this chemical reaction equation to be balanced? *Hint: Assume x, y, and z molecules of oxygen, carbon dioxide, and water, respectively, and use the balance of the number of atoms of carbon, hydrogen, and oxygen to form linear equations. Solve the system of linear equations.*

5.6 Carry out the same procedure as the previous problem but for the burning of methane CH_4.

5.7 (a) Use matrix multiplication to show that the vector

$$\mathbf{u} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5.101}$$

is not an eigenvector of the following matrix:

$$\mathbf{A} = \begin{bmatrix} 0 & 4 \\ -2 & -7 \end{bmatrix} \tag{5.102}$$

(b) Find all the unit eigenvectors of matrix $\mathbf{A}$ in (a).

5.8 Use hand calculation to compute the matrix multiplication of $\mathbf{UDV}^t$ where the data of relevant matrices are given by the following R output.

```
A=matrix(c(1,-1,1,1),nrow=2)
A
#[1,]    1    1
#[2,]   -1    1
svd(A)
#$d
#[1] 1.414214 1.414214
#$u
#[1,] -0.7071068 0.7071068
#[2,]  0.7071068 0.7071068
#$v
#[1,]   -1    0
#[2,]    0    1
```

5.9 Use a computer to find the matrices $\mathbf{U}, \mathbf{D}$ and $\mathbf{V}$ of the SVD for the following data matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{bmatrix}. \tag{5.103}$$

5.10 Use the first singular vectors from the result of Exercise 5.9 to approximately reconstruct the data matrix $\mathbf{A}$ using

$$B = d_1 \mathbf{u}_1 \mathbf{v}_1^t. \tag{5.104}$$

Describe the goodness of the approximation using text, limited to 20 to 100 words.

5.11 For the following data matrix

$$\mathbf{A} = \begin{bmatrix} 1.2 & -0.5 & 0.9 & -0.6 \\ 1.0 & -0.7 & -0.4 & 0.9 \\ -0.2 & 1.1 & 1.6 & -0.4 \end{bmatrix}, \tag{5.105}$$

(a) Use a computer to find the eigenvectors and eigenvalues of matrix $\mathbf{A}\mathbf{A}^t$.
(b) Use a computer to find the eigenvectors and eigenvalues of matrix $\mathbf{A}^t\mathbf{A}$.
(c) Use a computer to calculate SVD of $\mathbf{A}$.
(d) Compare the singular vectors and singular values in Step (c) with the eigenvalues and eigenvectors computed in Steps (a) and (b). Use text to discuss your comparison.
(e) Use a computer to verify that the column vectors of $\mathbf{U}$ and $\mathbf{V}$ in the SVD of Step (c) are orthonormal to each other.

5.12 Conduct an SVD analysis of the December standardized anomalies data of sea level pressure (SLP) at Darwin and Tahiti from 1961 to 2010. You can find the data from the Internet or from the website of this book.

(a) Write a computer code to organize the data into a 2×50 space-time data matrix.
(b) Make the SVD calculation for this space-time matrix.
(c) Plot the first singular vector in $\mathbf{V}$ against time from 1961 to 2010 as a time series curve, which is called the first principal component, denoted by PC1.
(d) Interpret the first singular vector in $\mathbf{U}$, which is called the first empirical orthogonal function (EOF1), as weights of Darwin and Tahiti stations.
(e) Check the historical El Niño events between 1961 and 2010 from the Internet, and interpret the extreme values of PC1.

5.13 Plot PC2 against time from the SVD analysis in the previous problem. Discuss the singular values λ_1 and λ_2. Interpret PC2 in reference to Trenberth (2020).

5.14 Conduct an SVD analysis similar to the previous two problems for the January standardized SLP anomalies data at Darwin and Tahiti from 1961 to 2010.

5.15 Conduct an SVD analysis similar to the previous problem for the monthly standardized SLP anomalies data at Darwin and Tahiti from 1961 to 2010. This problem includes anomalies for every month. The space-time data is a 2×600 matrix.

5.16 For the observed data of the monthly surface air temperature at five stations of your interest from January 1961 to December 2010, form a space-time data matrix, compute the SVD of this matrix, and interpret your results from the perspective of climate science. Plot PC1, PC2, and PC3 against time. You may find your data from the internet, e.g., the NOAA Climate Data Online website `www.ncdc.noaa.gov/cdo-web`

5.17 Do the same analysis as the previous problem, but for the monthly precipitation data at the same stations in the same time period.

5.18 For a Reanalysis dataset, conduct an SVD analysis similar to the previous problem for the monthly surface temperature data over 10 grid boxes of your choice in the time

period of 1961–2010. The space-time data is a 10×600 matrix. You can use your preferred Reanalysis dataset and download the data from the Internet, e.g., NCEP/NCAR Reanalysis, and ECMWF ERA.

5.19 Let $\mathbf{C} = \mathbf{A}\mathbf{A}^t$ and $\mathbf{A}$ is any real-valued rectangular matrix. Show that

(a) the eigenvalues of $\mathbf{C}$ are nonnegative;
(b) if $\mathbf{v} = \mathbf{A}^t\mathbf{u}$, then $\mathbf{v}$ is a eigenvector of $\mathbf{C}^t = \mathbf{A}^t\mathbf{A}$.

5.20 Given that

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{5.106}$$

write down the second-order polynomial corresponding to the following matrix expression:

$$P(x_1, x_2) = \mathbf{x}^t\mathbf{A}\mathbf{A}^t\mathbf{x}. \tag{5.107}$$

This is known as the quadratic form of the matrix $\mathbf{A}\mathbf{A}^t$.

5.21 If $\mathbf{x}$ is a unit vector, use calculus to find the maximum value of $P(x_1, x_2)$ in the previous problem. How is your solution related to the eigenvalue of $\mathbf{A}\mathbf{A}^t$?