Matrix Decompositions

Gaston Sanchez

November 7, 2016

References

Most of the content in this unit is a curated collection of notes and quotes from:

- **Mathematical Tools for Applied Multivariate Analysis** by Douglas Carroll, Paul Green, and Anil Chaturvedi.
- **Introduction to Multivariate Analysis for the Social Sciences** by John Van de Geer.
- **Understanding Complex Datasets: Data Mining with Matrix Decomposition** by David Skillicorn.

Matrix Decompositions

A matrix decomposition is a way of expressing a matrix $\mathbf{A}$ as the product of a set of new—typically two or three—matrices, usually simpler in some way, that shed light on the structures or relationships implicit in $\mathbf{A}$. By "simpler" we mean more compact matrices, with less dimensions, less number of rows, or less number of columns, perhaps triangular matrices or even matrices almost full of zeros, except in their diagonal. Decompositions are like factorizations of polynomials, they make it easier to study the properties of the objects (here: matrices). Also, given a decomposition, computation generally becomes easier.

There are many different types of matrix decompositions, and each of them reveal different kinds of underlying structure. We are going to focus our discussion on two of them: the Singular Value Decomposition (SVD), and the Eigen Value Decomposition (EVD). We will also briefly comment on the QR decomposition and the Cholesky decomposition. R provides several functions to decompose a matrix:

- `svd()`: singular value decomposition
- `eigen()`: eigenvalue decomposition (spectral decomposition)
- `qr()`: QR decomposition
- `chol()`: Cholesky decomposition

In this unit, we concentrate on the two types of matrices important in statistics: *general rectangular matrices* used to represent data tables, and *positive semi-definite matrices* used to represent covariance matrices, correlation matrices, and any matrix that results from a crossproduct.

More formally, a matrix decomposition can be described by an equation:

$$\mathbf{A} = \mathbf{CWF}$$

where the dimensions of the matrices are as follows:

- $\mathbf{A}$ is $n \times p$ (we assume for simplicity that $n > p$)

- **C** is $n \times k$ (usually $k < p$)
- **W** is $k \times k$ (usually diagonal)
- **F** is $k \times p$

The following figure illustrates a matrix decomposition:

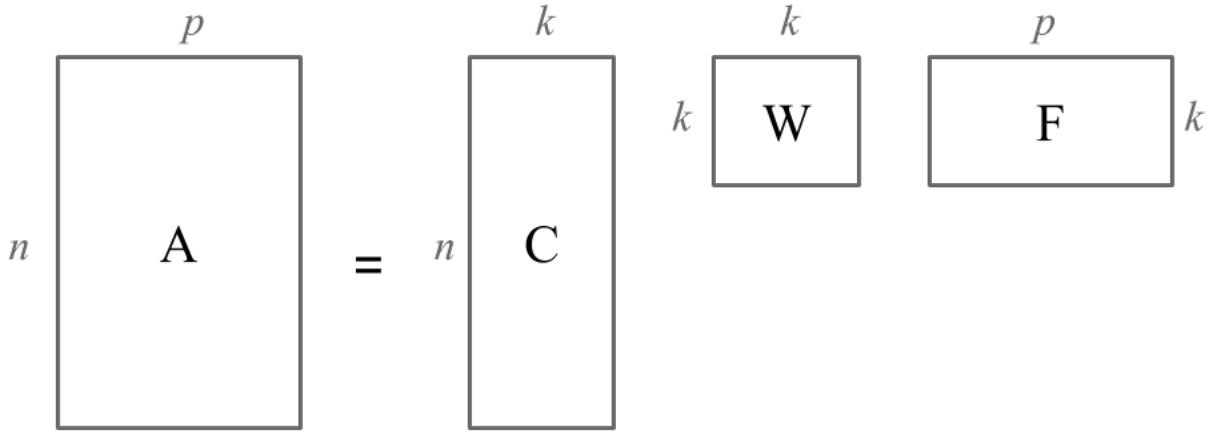

Figure 1: matrix decomposition

The matrix **C** has the same number of rows as **A**. Each row of **C** gives a different view of the object described by the corresponding row of **A**. In other words, the i-th row of **C** provides k pieces of information that together are a new view of the i-th object.

The matrix **F** has the same number of columns as **A**. Each column of **F** gives a different view of the variable described by the corresponding column of **A**, in terms of k pieces of information, rather than the n pieces of information in **A**.

The role of k is to force a representation for the data that is more compact than its original form. Choosing $k = p$ still gives a sensible decomposition, but it is usually the case that k is chosen to be smaller than p. We are implicitly assuming that a more compact representation will capture underlying regularities in the data that might be obscured by the form in which the data is found in **A**, usually because **A** expresses the data in a way that contains redundancies.

The matrix **W** has entries that reflect connections among the different latent or implicit regularities. For us, **W** will always be a diagonal matrix. Some decompositions do not create this middle matrix, but we can always imagine that it is there as the $k \times k$ identity matrix **I**.

Usually k is smaller, often mush smaller, than p, but a few matrix decompositions allow $k > p$. In this case, the underlying factors must somehow be of particular simple kind, so that the matrix decomposition is still forced to discover a compact representation.

Rank

There is one general concept which plays a fundamental role in many decompositions: the **rank**. If **X** is a $n \times p$ matrix, then **X** is of **full column rank** if $\mathbf{Xb} = \mathbf{0}$ only if $\mathbf{b} = \mathbf{0}$. We also say that the

columns of $\mathbf{X}$ are linearly independent. Full row rank is defined in a similar way.

If $\mathbf{X}$ can be written as the product $\mathbf{AB}$, with $\mathbf{A}$ an $n \times k$ matrix of full column rank and $\mathbf{B}$ an $k \times p$ matrix of full row rank, then $\mathbf{X}$ is said to have rank k. The decomposition $\mathbf{X} = \mathbf{AB}$ is a *full rank decomposition*.

The *rank* of a matrix is the minimum number of row or column vectors needed to generate the rows or columns of the matrix exactly through linear combinations. Geometrically, this algebraic concept is equivalent to the *dimensionality* of the matrix. In practice, however, no large matrix is of low rank, but we can approximate it "optimally" by a matrix of low rank and then view this approximate matrix in a low-dimensional space.

Suppose that $\mathbf{X}$ is an $n \times p$ matrix with rank r. Then the idea of matrix approximation is to find another $n \times p$ matrix $\hat{\mathbf{X}}$ of lower rank $k < r$ that resembles $\mathbf{X}$ "as closely as" possible. Closeness can be measured in any reasonable way, but least-squares approximation makes the solution of the problem particularly simple. Hence we want to find a matrix $\hat{\mathbf{X}}$ that minimizes the following objective function over all possible rank k matrices:

$$trace[(\mathbf{X} - \hat{\mathbf{X}})(\mathbf{X} - \hat{\mathbf{X}})^T]$$

Normalization

Because matrix decompositions are numerical computations, the magnitued of the values in different columns must be comparable, or else the larger magnitudes will have a greater influence on the result than the smaller ones.

One standard way to adjust attribute values is to subtract the mean from the entries in each column, which centers the value around zero; and then divide each entry in each column by the standard deviation of the column mean.

Correlation matrices

Given a data matrix $\mathbf{A}$ with mean-centered and standardized columns, we can form the matrices $\mathbf{A}^T\mathbf{A}$ and $\mathbf{AA}^T$. The matrix $\mathbf{AA}^T$ is the *correlation matrix* of the objects. The magnitude of the ij-th entry indicates the amount of correlation between the i-th and the j-th object. Similarly, the matrix $\mathbf{A}^T\mathbf{A}$ is the *correlation matrix* of the variables, and its entries indicate the amount of correlation between pairs of variables. Both of these matrices are symmetric.

The correlation matrices can also be decomposed, and the resulting matrices analyzed to give new insights into the structures present in the data. However, assuming that $n > p$, the matrix $\mathbf{AA}^T$ can be very large. Calculating a decomposition for such a matrix can often be difficult because of numerical instability.

The matrices used for data analysis are often very large, so it is useful to have some sense of the complexity of computing matrix decompositions. Because matrix decompositions are numerical algorithms, it is also important to be aware of how numerical magnitudes affect results; this can sometimes cause computational problems such as instability.

Interpreting Decompositions

The equation that describes a decomposition:

$$\mathbf{A} = \mathbf{CWF}$$

does not explain how to compute one, or how such decomposition can reveal the structures implicit in a data matrix. Nowadays, the computation of a matrix decomposition is straightforward; software to compute each one is readily available, and understanding how the algorithms work is not necessary to be able to interpret the results.

Seeing how a matrix decomposition reveals structure in a dataset is more complicated. Each decomposition reveals a different kind of implicit structure and, for each decomposition, there are various ways to interpret the results.

Singular Value Decomposition (SVD)

One of the most important decompositions in matrix algebra is known as the **Singular Value Decomposition**, commonly known as SVD. It can be applied to **any** type of matrix—rectangular or square, singular or nonsigular. The Singular Value Decomposition expresses any matrix, such as an $n \times p$ matrix $\mathbf{X}$ as the product of three other matrices:

$$\mathbf{X} = \mathbf{UDV}^{\mathsf{T}}$$

where $\mathbf{U}$ and $\mathbf{V}$ are each orthonormal sections and $\mathbf{D}$ is a diagonal matrix of ordered positive values. The rank of $\mathbf{X}$ is given by r, the number of such positive values (which are called singular values). Furthermore, $r(\mathbf{X}) \leq min(n, p)$.

We can think of the SVD structure as *the basic structure of a matrix*. What do we mean by "basic"? Well, this has to do with what each of the matrices $\mathbf{UDV}^{\mathsf{T}}$ represent.

The matrix $\mathbf{U}$ is the orthonormalized matrix which is the most basic component. It's like the skeleton of the matrix.

The matrix $\mathbf{D}$ is referred to as the *spectrum* and it is a scale component. Note that all the values in the diagonal of $\mathbf{D}$ are non-negative numbers. This matrix is also unique. It is a like a fingerprint of a matrix. It is assumed that the singular values are ordered from largest to smallest.

Finally, the matrix $\mathbf{V}$ is the orientation or correlational component.

$$\mathbf{X} = \begin{bmatrix} u_{11} & \cdots & u_{1p} \\ u_{21} & \cdots & u_{2p} \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{np} \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_p \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{p1} \\ \vdots & \ddots & \vdots \\ v_{1p} & \cdots & v_{pp} \end{bmatrix}$$

Alternatively,

$$\begin{bmatrix} \\ \mathbf{X} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{U} \\ \\ \end{bmatrix} \begin{bmatrix} \\ \mathbf{D} \\ \\ \end{bmatrix} \begin{bmatrix} \\ \mathbf{V}^\mathsf{T} \\ \\ \end{bmatrix}$$

The SVD says that we can express the basic structure of a matrix $\mathbf{X}$ with the product of an orthonormal matrix $\mathbf{U}$, a diagonal matrix $\mathbf{D}$, and an orthonormal matrix $\mathbf{V}$.

- $\mathbf{U}$ is a $n \times p$ column **orthonormal** matrix containing the left singular vectors
- $\mathbf{D}$ is a $p \times p$ **diagonal** matrix containing the singular values of $\mathbf{X}$
- $\mathbf{V}$ is a $p \times p$ column **orthonormal** matrix containing the right singular vectors

Relation of SVD and Cross-Product Matrices

The cross-product matrix of columns can be expressed as:

$$\mathbf{X}^\mathsf{T}\mathbf{X} = \mathbf{V}\mathbf{D}^\mathbf{2}\mathbf{V}^\mathsf{T}$$

The cross-product matrix of rows can be expressed as:

$$\mathbf{X}\mathbf{X}^\mathsf{T} = \mathbf{U}\mathbf{D}^\mathbf{2}\mathbf{U}^\mathsf{T}$$

One of the interest things about SVD is that $\mathbf{U}$ and $\mathbf{V}$ are matrices whose columns are eigenvectors of product moment matrices that are *derived* from $\mathbf{X}$. Specifically,

- $\mathbf{U}$ is the matrix of eigenvectors of (symmetric) $\mathbf{X}\mathbf{X}^\mathsf{T}$ of order $n \times n$
- $\mathbf{V}$ is the matrix of eigenvectors of (symmetric) $\mathbf{X}^\mathsf{T}\mathbf{X}$ of oreder $p \times p$

Of additional interest is the fact that $\mathbf{D}$ is a diagonal matrix whose main diagonal entries are the square roots of $\mathbf{U}^2$, the *common* matrix of eigenvalues of $\mathbf{X}\mathbf{X}^\mathsf{T}$ and $\mathbf{X}^\mathsf{T}\mathbf{X}$.

SVD Rank-Reduction Theorem

A very interesting and alternative way to represent the SVD is with the following formula:

$$\mathbf{X} = \sum_{k=1}^{p} \lambda_k \mathbf{u_k} \mathbf{v_k'}$$

This equation expresses the SVD as a sum of p rank 1 matrices. This result is formalized in what is known as the SVD theorem described by Carl Eckart and Gale Young in 1936, and it is often referred to as the Eckart-Young theorem. This theorem applies to practically any arbitrary rectangular matrix.

The SVD theorem of Eckart and Young is related to the important problem of approximating a matrix. The basic result says that if $\mathbf{X}$ is an $n \times p$ rectangular matrix, then the best r-dimensional approximation $\hat{\mathbf{X}}$ to $\mathbf{X}$ is obtained by minimizing:

$$min\|\mathbf{X} - \hat{\mathbf{X}}\|^2$$

This type of approximation is a least squares approximation and the solution is obtained by taking the first r elements of matrices $\mathbf{U}, \mathbf{D}, \mathbf{V}$ so that the $n \times r$ matrix $\hat{\mathbf{X}} = \mathbf{U_r D_r V_r^\top}$

If you think about it, the SVD is of great utility because it tells us that the best 1-rank approximation, in the least squares sense, of any matrix $\mathbf{X}$ is $\lambda_1 \mathbf{u_1 v_1^\top}$. This implies that, from a conceptual standpoint, we can approximate the $n \times p$ numbers in $\mathbf{X}$ with just $n + p + 1$ values: n numbers in $\mathbf{u_1}$, p numbers in $\mathbf{v_1}$, and one scalar λ_1.

The SVD theorem says that a rectangular matrix $\mathbf{X}$ can be broken down into the product of three matrices—an orthogonal matrix $\mathbf{U}$, a diagonal matrix $\mathbf{D}$, and the transpose of an orthogonal matrix $\mathbf{V}$.

In terms of the diagonal elements $\lambda_1, \lambda_2, \ldots, \lambda_r$ of $\mathbf{\Lambda}$, the columns $\mathbf{u_1}, \ldots, \mathbf{u_r}$ of $\mathbf{U}$, and the columns $\mathbf{v_1}, \ldots, \mathbf{v_r}$ of $\mathbf{V}$, the basic structure of $\mathbf{X}$ may be written as

$$\mathbf{X} = \lambda_1 \mathbf{u_1 v_1^\top} + \lambda_2 \mathbf{u_2 v_2^\top} + \cdots + \lambda_r \mathbf{u_r v_r^\top}$$

which shows that the matrix $\mathbf{X}$ of rank r is a linear combination of r matrices of rank 1.

QR Decomposition

Another type of decomposition that applies to any rectangular matrix $\mathbf{X}$, is the so-called *QR decomposition* that factorizes $\mathbf{X}$ as:

$$\mathbf{X} = \mathbf{QR}$$

with $\mathbf{Q}$ an $n \times p$ orthogonal matrix and $\mathbf{R}$ a $p \times p$ nonsingular, upper-triangular matrix.

The QR decomposition can be obtained via Gram-Schmidt orthogonalization. To create the $\mathbf{Q}$ matrix the columns of $\mathbf{X}$ can be orthonormalized using the procedure we now describe.

Let $\mathbf{x_1}, \ldots, \mathbf{x_p}$ be linearly independent, n-vector columns of $\mathbf{X}$. The Gram-Schmidt algorithm creates an orthonormal basis $\mathbf{q_1}, \ldots, \mathbf{q_p}$ for the linear manifold spanned by $\mathbf{x_1}, \ldots, \mathbf{x_p}$.

$q_1 = x_1 / \|x_1\|$

This algorithm can exhibit numerical instability. But, it is easily modified to resolve such problems. Since the vectors $\mathbf{q_1}, \ldots, \mathbf{q_p}$ returned from the Gram-Schmidt algorithm are an orthonormal basis

Algorithm 1 Gram-Schmidt algorithm

$q_1 = x_1/\|x_1\|$
for $i = 2, \ldots, p$ **do**
  $q_i = x_i - \sum_{j=1}^{i-1}(x_i^T q_j)q_j$
  $q_i := q_i/\|q_i\|$
end for

for the columns of $\mathbf{X}$ it must be that $\mathbf{X} = \mathbf{QR}$. The order in which the $\mathbf{q_i}$ have been constructed guarantees that $\mathbf{R}$ is upper-triangular.

```
# Gram Schmidt algorithm
# x: matrix to be decomposed
# epsilon: threshold tolerance
gram_schmidt <- function(x, epsilon = 1e-15) {
  m <- ncol(x)
  Q <- x
  k <- 0
  R <- matrix(0, m, m)
  for (i in 1:m) {
    if (i > 1) {
      for (j in 1:(i-1)) {
        a <- sum(Q[,i]*Q[,j])
        R[j,i] <- a
        Q[,i] <- Q[,i] - (a * Q[,j])
      }
    }
    a <- sqrt(sum(Q[,i]^2))
    if (a < epsilon) next()
    Q[,i] <- Q[,i] / a
    R[i,i] <- a
    k <- k + 1
  }
  return(list(Q = Q, R = R, rank = k))
}
```

It can happen in these computation that orthogonalization during QR computations makes a column equal to zero. In that case we know that the matrix is column-singular and we eliminate the column from the QR decomposition. Thus we find $\mathbf{X} = \mathbf{QR}$ with $\mathbf{Q}$ of dimension $n \times r$, and $\mathbf{R}$ non-singular and upper triangular of dimension $r \times p$. The number r is the rank of $\mathbf{X}$.

R provides the function `qr()`

```
set.seed(567)
X <- matrix(rnorm(15), 5, 3)
Y <- qr(X)
Y
```

```
## $qr
##            [,1]        [,2]        [,3]
```

```
## [1,] -1.7548695 -0.25674659  0.93731406
## [2,]  0.1932747  2.91816108  2.10887375
## [3,] -0.3651317  0.74157245  1.75386839
## [4,] -0.8206287  0.01078994 -0.08255592
## [5,]  0.1258341  0.65991624 -0.81897123
##
## $rank
## [1] 3
##
## $qraux
## [1] 1.374243 1.120268 1.567865
##
## $pivot
## [1] 1 2 3
##
## attr(,"class")
## [1] "qr"
```

```
qr.Q(Y)
```

```
##            [,1]        [,2]       [,3]
## [1,] -0.3742434 -0.17334138 -0.2405130
## [2,] -0.1932747 -0.14464704 -0.1540561
## [3,]  0.3651317 -0.69551624 -0.5835491
## [4,]  0.8206287  0.09272077  0.2250201
## [5,] -0.1258341 -0.67578844  0.7261244
```

```
qr.R(Y)
```

```
##          [,1]       [,2]      [,3]
## [1,] -1.75487 -0.2567466 0.9373141
## [2,]  0.00000  2.9181611 2.1088738
## [3,]  0.00000  0.0000000 1.7538684
```

Using the QR decomposition the least-squares normal equations $\mathbf{X}^T\mathbf{X}\mathbf{b} = \mathbf{X}^T\mathbf{y}$ become:

$$\mathbf{R}^T\mathbf{R}\mathbf{b} = \mathbf{R}^T\mathbf{Q}^T\mathbf{y}$$

Eigenvalues and Eigenvectors

Eigenvalues play a fundamental role in statistics. This is particularly true in the case of multivariate analysis where eigenvalues of sample covariance matrices and related quantities provide the foundation for many tools that are used for statistical inference.

In this section we will address the computation of eigenvalues corresponding to a real, positive-semidefinite (i.e. symmetric), $p \times p$ matrix $\mathbf{A}$.

Vectors, which under a given transformation map into themselves of multiples of themselves, are called invariant vectors under that transformation. It follows that such vectors satisfy the relation:

$$\mathbf{Ax} = \lambda\mathbf{x}$$

where λ is a scalar.

The matrix equation

$$\mathbf{Ax} = \lambda\mathbf{x}$$

can be arranged as follows:

$$\mathbf{Ax} - \lambda\mathbf{x} = \mathbf{0}$$

or equivalently

$$\mathbf{Ax} - \lambda\mathbf{Ix} = \mathbf{0}$$

Next, we can factor out $\mathbf{x}$ to get

$$(\mathbf{Ax} - \lambda\mathbf{I})\,\mathbf{x} = \mathbf{0}$$

Recap

Obtaining the eigenstructure of a (square) matrix involves solving the characteristic quation

$$|\mathbf{A} - \lambda_i\mathbf{I}| = 0$$

If $\mathbf{A}$ is of order $n \times n$, then we can obtain n roots of the equation; these roots are called the eigenvalues.

Power Method

Among all the set of methods which can be used to find eigenvalues and eigenvectors, one of the basic procedures following a successive approximation approach is precisely the **Power Method**.

In its simplest form, the Power Method (PM) allows us to find *the largest* eigenvector and its corresponding eigenvalue. To be more precise, the PM allows us to find an approximation for the first eigenvalue of a symmetric matrix $\mathbf{S}$. The basic idea of the power method is to choose an arbitrary vector $\mathbf{w_0}$ to which we will apply the symmetric matrix $\mathbf{S}$ repeatedly to form the following sequence:

$$\begin{aligned}
\mathbf{w_1} &= \mathbf{Sw_0} \\
\mathbf{w_2} &= \mathbf{Sw_1} = \mathbf{S^2w_0} \\
\mathbf{w_3} &= \mathbf{Sw_2} = \mathbf{S^3w_0} \\
&\vdots \\
\mathbf{w_k} &= \mathbf{Sw_{k-1}} = \mathbf{S^kw_0}
\end{aligned}$$

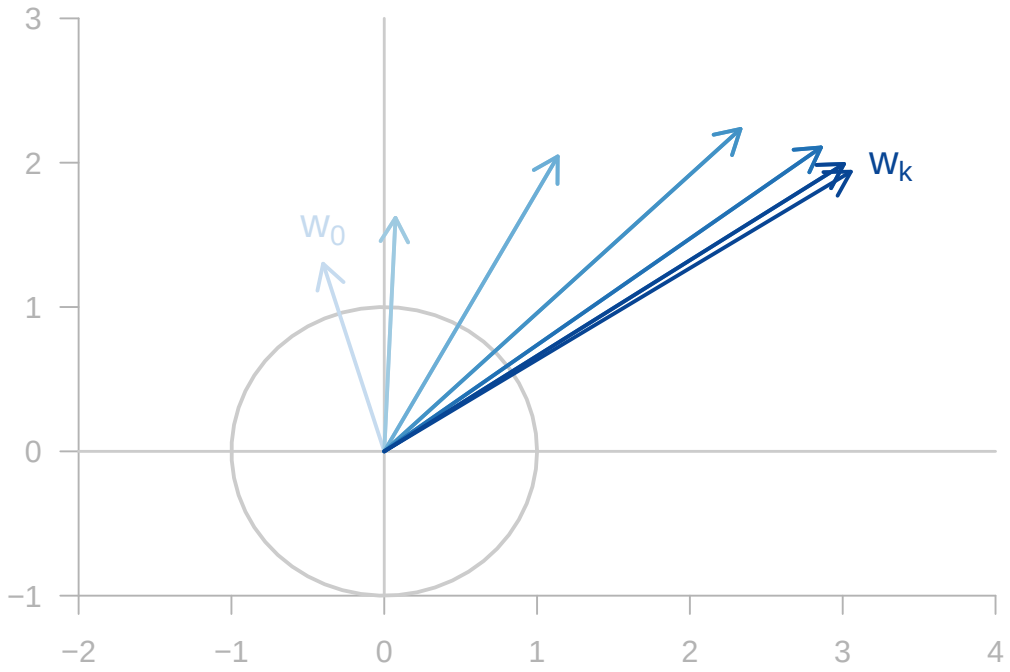

Figure 2: Illustration of the sequence of vectors in the Power Method

As you can see, the PM reduces to simply calculate the powers of $\mathbf{S}$ multiplied to the initial vector $\mathbf{w_0}$. Hence the name of *power method*.

In practice, we must rescale the obtained vector $\mathbf{w_k}$ at each step in order to avoid an eventual overflow or underflow. Also, the rescaling will allows us to judge whether the sequence is converging. Assuming a reasonable scaling strategy, the sequence of iterates will usually converge to the dominant eigenvector of $\mathbf{S}$. In other words, after some iterations, the vector $\mathbf{w_{k-1}}$ and $\mathbf{w_k}$ will be very similar, if not identical. The obtained vector is the dominant eigenvector. To get the corresponding eigenvalue we calculate the so-called *Rayleigh quotient* given by:

$$\lambda = \frac{\mathbf{w_k'} \mathbf{S'} \mathbf{w_k}}{\|\mathbf{w_k}\|^2}$$

Power Method Algorithm

There are some conditions for the power method to be succesfully used. One of them is that the matrix must have a *dominant* eigenvalue. The starting vector $\mathbf{w_0}$ must be nonzero. Very important, we need to scale each of the vectors $\mathbf{w_k}$, otherwise the algorithm will "explode".

The Power Method is of a striking simplicity. The only thing we need, computationally speaking, is the operation of matrix multiplication. Let's consider a more detailed version of the PM algorithm walking through it step by step:

1. Start with an arbitraty initial vector $\mathbf{w}$
2. obtain product $\tilde{\mathbf{w}} = \mathbf{S}\mathbf{w}$

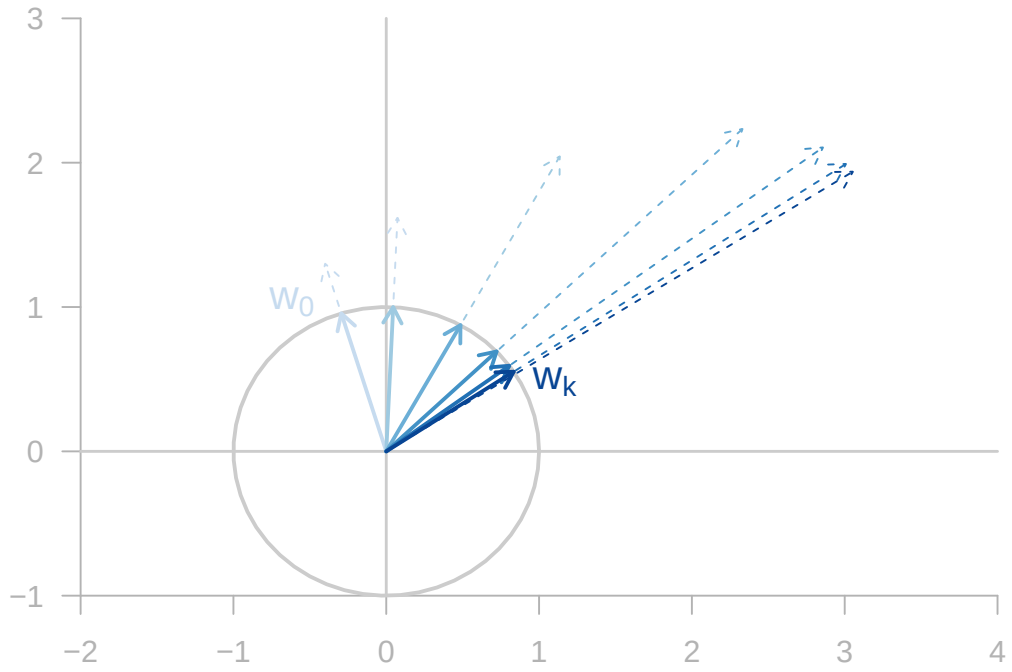

Figure 3: Sequence of vectors before and after scaling to unit norm

3. normalize $\tilde{\mathbf{w}}$

$$\mathbf{w} = \frac{\tilde{\mathbf{w}}}{\|\tilde{\mathbf{w}}\|}$$

4. compare $\mathbf{w}$ with its previous version
5. repeat steps 2 till 4 until convergence

Convergence of the Power Method

To see why and how the power method converges to the dominant eigenvalue, we need an important assumption. Let's say the matrix $\mathbf{S}$ has p eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$, and that they are ordered in decreasing way $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_p|$. Note that the first eigenvalue is strictly greater than the second one. This is a very important assumption. In the same way, we'll assume that the matrix $\mathbf{S}$ has p linearly independent vectors $\mathbf{u_1}, \ldots, \mathbf{u_p}$ ordered in such a way that $\mathbf{u_j}$ corresponds to λ_j.

The initial vector $\mathbf{w_0}$ may be expressed as a linear combination of $\mathbf{u_1}, \ldots, \mathbf{u_p}$

$$\mathbf{w_0} = a_1\mathbf{u_1} + \cdots + a_p\mathbf{u_p}$$

At every step of the iterative process the vector $\mathbf{w_m}$ is given by:

$$\mathbf{S}^m = a_1\lambda_1^m\mathbf{u_1} + \cdots + a_p\lambda_p^m\mathbf{u_p}$$

Since λ_1 is the dominant eigenvalue, the component in the direction of $\mathbf{u_1}$ becomes relatively greater than the other components as m increases. If we knew λ_1 in advance, we could rescale at each step by dividing by it to get:

$$\left(\frac{1}{\lambda_1^m}\right)\mathbf{S}^m = a_1\mathbf{u_1} + \cdots + a_p\left(\frac{\lambda_p^m}{\lambda_1^m}\right)\mathbf{u_p}$$

which converges to the eigenvector $a_1\mathbf{u_1}$, provided that a_1 is nonzero. Of course, in real life this scaling strategy is not possible—we don't know λ_1. Consequenlty, the eigenvector is determined only up to a constant multiple, which is not a concern since the really important thing is the *direction* not the length of the vector.

The speed of the convergence depends on how bigger λ_1 is respect with to λ_2, and on the choice of the initial vector $\mathbf{w_0}$. If λ_1 is not much larger than λ_2, then the convergence will be slow.

One of the advantages of the power method is that it is a sequential method; this means that we can obtain $\mathbf{w_1}, \mathbf{w_2}$, and so on, so that if we only need the first k vectors, we can stop the procedure at the desired stage. Once we've obtained the first eigenvector $\mathbf{w_1}$, we can compute the second vector by reducing the matrix $\mathbf{S}$ by the amount explained by the first principal component. This operation of reduction is called **deflation** and the residual matrix is obtained as:

$$\mathbf{E} = \mathbf{S} - \mathbf{z_1'z_1}$$

In order to calculate the second eigenvalue and its corresponding eigenvector, we operate on $\mathbf{E}$ in the same way as the operations on $\mathbf{S}$ to obtain $\mathbf{w_1}$. The quantity $\mathbf{z_2'z_2}$ will be the amount of variation explained by the second PC.