Chapter 7

Matrix Decomposition

This chapter is dedicated to explaining the concept of **matrix decomposition or factorization**, its definition, and the process. You will learn how you can decompose a matrix to its constituent elements.

7.1. Introduction

What is the ***benefit of decomposing a matrix***? What does that mean? When we *decompose anything*, we *break it into its constituent elements*. Assume we are going to disintegrate a **tool** (a car or a watch!). Such action helps us to understand the core particles and their tasks. Furthermore, it helps to have a better understanding of how that specific tool works and its characteristics! *Assume that the tool is a matrix which we would like to decompose*. There are different approaches to decompose a matrix. However, perhaps the most commonly used ones are matrix eigendecomposition and Singular Value Decomposition (SVD).

7.2. Matrix Eigendecomposition

In this section, I am going to show you the definition of eigendecomposition and the subsequent concepts necessary to understand it.

7.2.1 Eigenvector and Eigenvalue

Before we move on, we should know the definition of eigenvector and eigenvalue. The definition of eigenvector and eigenvalue are somehow connected.

Definition

Assuming we have the square matrix of $\mathbf{A} \in \mathbb{R}^{N \times N}$. The nonzero vector $\mathbf{v} \in \mathbb{R}^{N \times 1}$ is an eigenvector and scalar λ is its associated eigenvalue if we have:

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}$$

From the above definition, it is clear than if $\mathbf{v}$ is an eigenvector, any vector $\alpha\mathbf{v}, \alpha \in \mathbb{R}$ is also an eigenvector with the same eigenvalue λ. Therefore, if we have one eigenvector, then we have infinite ones!

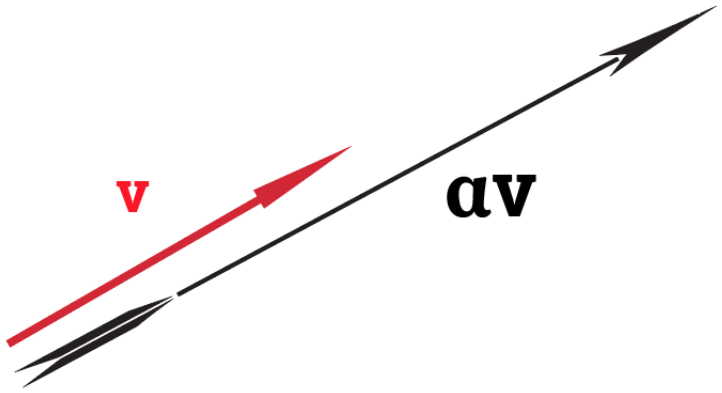

Fig. 7.1. Since $\mathbf{v}$ is an eigenvector, any vector $\alpha\mathbf{v}$ is also an eigenvector

Due to that, it is customary to only work with eigenvectors that have **unit norm**. It is simple to construct an eigenvector with the unit norm. Assume $\mathbf{v}$ is our eigenvector. Then, the following vector is also an eigenvector with the unit norm:

$$\mathbf{w} = \alpha\mathbf{v} = \frac{1}{||\mathbf{v}||}\mathbf{v}$$

where $||\mathbf{v}||$ is the norm of vector $\mathbf{v}$. We usually consider the euclidean norm.

7.2.2 The Process

Here, the process of how we decompose a matrix to its constituent elements is explained. It is called the **eigendecomposition of a matrix**.

Matrix Eigendecomposition

Assume we have the square matrix of $\mathbf{A} \in \mathbb{R}^{N \times N}$ which has N linear independent eigenvectors $\mathbf{v}^i, i \in 1, \ldots, N$. Then, we can factorize matrix $\mathbf{A}$ as below:

$$\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1}$$

where $\mathbf{V} \in \mathbb{R}^{N \times N}$ is the square matrix whose j^{th} column is the eigenvector $\mathbf{v}^j$ of $\mathbf{A}$, and Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, $\Lambda_{jj} = \lambda_j$.

Above, we basically concatenate eigenvectors $\mathbf{v}^i$ to form the $\mathbf{V}$ matrix as below:

$$\begin{bmatrix} | & | & \dots & | \\ | & | & \dots & | \\ \mathbf{v}^1 & \mathbf{v}^2 & \dots & \mathbf{v}^N \\ | & | & \dots & | \\ | & | & \dots & | \end{bmatrix}$$

7.2.3 Discussion on Matrix Eigendecomposition

Note that *we can only factorize **diagonalizable matrices** as above*. But the question is what is a diagonalizable matrix?

Diagnolizable Matrix

Assume we have the square matrix of $\mathbf{A} \in \mathbb{R}^{N \times N}$. It is called diagonalizable or nondefective if there exists an invertible matrix $\mathbf{H}$ such that $\mathbf{HAH}^{-1}$ is a diagonal matrix.

let's have a more precise definition of a matrix being singular or non-singular.

Singular Matrix

Assume we have the square matrix of $\mathbf{A} \in \mathbb{R}^{N \times N}$. It is called singular if and only if any of the eigenvalues (λ) are zero.

Under some circumstances, we can calculate the **matrix inverse** using the decomposition.

Matrix Inverse

Assume we have the square matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, it can be eigendecomposed and it is nonsingular. Therefore, we can calculate its inverse as below:

$$\mathbf{A}^{-1} = \mathbf{V}\blacksquare^{-1}\mathbf{V}^{-1}$$

Since Λ is diagonal, it inverse Λ^{-1} is also diagnoal and we can calculate it as:

$$\Lambda_{jj}^{-1} = \frac{1}{\lambda_j}$$

7.2.4 One Special Matrix Type and its Decomposition

We are interested to investigate *a special kind of matrix*: **Real symmetric matrix**. A real symmetric matrix is basically a symmetric matrix in which all elements belong to the space of real numbers $\mathbb{R}$.

For the **real symmetric matrix** of $\mathbf{A} \in \mathbb{R}^{N \times N}$, the eigenvalues are real numbers and we can choose eigenvectors in a way that they are orthogonal to each other. Then, we can factorize matrix $\mathbf{A}$ as below:

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$$

where $\mathbf{Q}$ is an orthogonal matrix whose columns are the eigenvectors of $\mathbf{A}$, and Λ is a diagonal matrix whose diagonal elements are the corresponding eigenvalues, $\Lambda_{jj} = \lambda_j$.

7.2.5 Useful Properties

So far, the concepts and how we can decompose a matrix were explained. Let's see how we can leverage it. Assuming $\mathbf{A} \in \mathbb{R}^{N \times N}$:

- The determinant of the matrix $\mathbf{A}$ equals the product of its eigenvalues.
- The trace of the matrix $\mathbf{A}$ equals the summation of its eigenvalues.
- If the eigenvalues of $\mathbf{A}$ are λ_i, and $\mathbf{A}$ is non-singular, then the eigenvalues of $\mathbf{A}^{-1}$ are simply $\frac{1}{\lambda_i}$.
- The eigenvectors of $\mathbf{A}^{-1}$ are the same as the eigenvectors of $\mathbf{A}$.

7.2.6 Using Python

Now, let's do some practical work. I want to use Python and NumPy to compute eigenvalues and eigenvectors.

```
1  # Import NumPy library
2  import numpy as np
3
4  # Define random 4x4 matrix using np.array
5  # Ref: https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/
       numpy.random.randint.html
6  N=4
7  A = np.random.randint(10, size=(N, N))
8  print('A:\n',A)
9
10 # Eigendecomposition
11 eigenvalues,eigenvectors= np.linalg.eig(A)
12
13 # Show values
14 print('Eigenvalues:', eigenvalues)
15 print('Eigenvectors:', eigenvectors)
16
17 # Create the diagonal matrix of \Lambda
18 Lambda = np.diag(eigenvalues)
19
20 # Create V, the matrix of eigenvectors
21 V = eigenvectors
22
23 # Check and Confirm the decomposition
24 A_ = np.matmul(np.matmul(V,Lambda),np.linalg.inv(V))
25 print('Computing A with decomposed elements:\n', A_)
```

Run the above code to see the results. Pretty simple. Right? In the above code, **line 24** aims to confirm if by using the decomposed elements $\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1}$ we can reconstruct $\mathbf{A}$. What is your observation?

7.3. Singular Value Decomposition (SVD)

We previously discussed matrix eigendecomposition as an approach to decompose a matrix using its eigenvalues and eigendecomposition. **There was one issue though:** For matrix eigendecomposition, *the matrix **MUST** be square*. The Singular Value Decomposition (SVD) does NOT have this limitation, and it makes it even more useful and powerful compared to eigendecomposition.

7.3.1 Preliminary Definitions

Let's have a preliminary definition before we move on.

Conjugate Transpose of a Matrix

The conjugate transpose or Hermitian transpose of $\mathbf{M} \in \mathbb{C}^{m \times n}$ is obtained taking the transpose and then taking the complex conjugate of each entry of matrix $\mathbf{M}$. The resulting matrix is denoted by $\mathbf{M}^*$ or $\mathbf{M}^H$.
The set $\mathbb{C}$ is the field of the complex numbers.

Now, let's define a special kind of matrices.

Unitary Matrix

A complex square matrix $\mathbf{M} \in \mathbb{C}^{n \times n}$ is unitary if its conjugate transpose $\mathbf{M}^*$ is also its inverse. It can be shown as below:

$$\mathbf{M}\mathbf{M}^* = \mathbf{M}^*\mathbf{M} = \mathbf{I}$$

where $\mathbb{C}$ is the field of the complex numbers.

NOTE: In the above definition, if $\mathbf{M}$ belongs to the field of real numbers $\mathbb{R}$, the matrix $\mathbf{M}$ will be orthogonal and $\mathbf{M}^* = \mathbf{M}^\mathbf{T}$.

7.3.2 Matrix decomposition with SVD

Now, let's define the main concept, Singular Value Decomposition (SVD).

Singular Value Decomposition

Assume we have the matrix of $\mathbf{A} \in \mathbb{C}^{M \times N}$. Then, we can factorize matrix $\mathbf{A}$ as below:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^*$$

where $\mathbf{U}$ is an $M \times M$ and $\mathbf{V}$ is an $N \times N$ matrix and both are unitary. The matrix Σ is a diagonal $M \times N$ matrix with non-negative real numbers on the diagonal.

NOTE: The matrix Σ is a diagonal matrix of size $M \times N$. So it does not have to be square!

The **diagonal elements** of Σ are known as the **singular values** of the $\mathbf{A}$. The **columns** of $\mathbf{U}$ are known as the **left-singular** vectors and are the eigenvectors of $\mathbf{A}\mathbf{A}^*$. The **columns** of $\mathbf{V}$ are known as the **right-singular** vectors and are the eigenvectors of $\mathbf{A}^*\mathbf{A}$.

The diagonal matrix Σ is **uniquely determined** by $\mathbf{A}$. The nonzero singular values of $\mathbf{A}$ are the square roots of the eigenvalues $\mathbf{A}^*\mathbf{A}$. That's why they are non-negative!

7.3.3 A Practical Implementation

Let's compute the singular value decomposition of a matrix using Python and NumPy.

```
1  # Import Numpy library
2  import numpy as np
3
4  # Define random 3x4 matrix using np.array
5  # Ref: https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/
       numpy.random.randint.html
6  M=3
7  N=4
8  A = np.random.randint(10, size=(M, N))
9  print('A:\n',A)
10
11 # Singular Value Decomposition
12 # With full_matrices=True, U and Vt matrices will be squared.
13 U, S, Vt = np.linalg.svd(A, full_matrices=True)
14
15 # Show the shape of outputs
16 print('The shape of U:', U.shape)
17 print('The shape of S:', S.shape)
18 print('The shape of V^{T}:', Vt.shape)
19
20 # Create the diagonal matrix of \Lambda
21 Sigma = np.diag(S)
22
23 # Matrix A is fat! Sigma is not diagonal!
24 if Sigma.shape[1] != Vt.shape[0]:
25     zero_columns_count = Vt.shape[0] - Sigma.shape[1]
26     additive = np.zeros((Sigma.shape[0],zero_columns_count), dtype=np
       .float32)
27     Sigma = np.concatenate((Sigma,additive), axis=1)
28
29 # Matrix A is fat! Sigma is not diagonal!
30 if Sigma.shape[0] != U.shape[1]:
31     zero_rows_count = U.shape[1] - Sigma.shape[0]
32     additive = np.zeros((zero_rows_count,Sigma.shape[1]), dtype=np.
       float32)
33     Sigma = np.concatenate((Sigma,additive), axis=0)
34
35 # Check and Confirm the decomposition
36 A_ = np.matmul(np.matmul(U,Sigma),Vt)
37 print('Computing A with decomposed elements:\n', A_)
```

In the above code, the matrices' shapes is just get printed. Try to change the code above to see the outputs. Starting from **line 20**, the code reconstruct matrix **A** using the decomposed vector by SVD to confirm the computation!

7.3.4 Applications of SVD

Moore–Penrose Pseudoinverse

Before, digging into how SVD can help us with pseudoinverse calculations, let's talked about what is a Pseudoinverse?

Moore–Penrose Pseudoinverse

Assuming we have the matrix of $\mathbf{A} \in \mathbb{C}^{M \times N}$. Then, we have a pseudoinverse of $\mathbf{A}$, defined as $\mathbf{A}^+ \in \mathbb{C}^{M \times N}$, that satisfies all of the following criteria:

- $\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$
- $\mathbf{A}^+\mathbf{A}\mathbf{A}^+ = \mathbf{A}^+$
- $(\mathbf{A}\mathbf{A}^+)^* = \mathbf{A}\mathbf{A}^+$
- $(\mathbf{A}^+\mathbf{A})^* = \mathbf{A}^+\mathbf{A}$

How we can *calculate the **Pseudoinverse** of a matrix using SVD*? If we have the SVD of a matrix $\mathbf{A}$ as $\mathbf{U}\Sigma\mathbf{V}^*$, then $\mathbf{A}^+$ can be calculated as below:

$$\mathbf{A}^+ = \mathbf{V}\Sigma^+\mathbf{U}^*$$

where Σ^+ is the pseudoinverse of Σ and we create it by replacing each non-zero diagonal entry σ in Σ by $\frac{1}{\sigma}$ and transposing the resulting matrix.

Matrix Rank Determination

Previously, the linear dependence and the matrix ranks were discussed. Now, as we have SVD, a more general approach compared to eigendecomposition, we can conclude the following directions.

Matrix Rank: Assume we have the SVD of a matrix $\mathbf{A}$ as $\mathbf{U}\Sigma\mathbf{V}^*$. The rank of $\mathbf{A}$ equals the number of non-zero singular values which is the number of non-zero diagonal elements in Σ.

7.4. Conclusion

In this chapter, the matrix decomposition (factorization) concept and two of its most used techniques were introduced. You learn what is matrix decomposition and how to do it. You also practiced to implement it in Python and NumPy. If you would like to know more about matrix decomposition, you can refer to [11, 12].