16.2 What is the Singular-Value Decomposition

The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler. For the case of simplicity we will focus on the SVD for real-valued matrices and ignore the case for complex numbers.

$$A = U \cdot \Sigma \cdot V^T \tag{16.1}$$

Where A is the real $n \times m$ matrix that we wish to decompose, U is an $m \times m$ matrix, Σ represented by the uppercase Greek letter sigma) is an $m \times n$ diagonal matrix, and V^T is the V transpose of an $n \times n$ matrix where T is a superscript.

> The Singular Value Decomposition is a highlight of linear algebra.
>
> — Page 371, *Introduction to Linear Algebra*, 2016.

The diagonal values in the Σ matrix are known as the singular values of the original matrix A. The columns of the U matrix are called the left-singular vectors of A, and the columns of V are called the right-singular vectors of A. The SVD is calculated via iterative numerical methods. We will not go into the details of these methods. Every rectangular matrix has a singular value decomposition, although the resulting matrices may contain complex numbers and the limitations of floating point arithmetic may cause some matrices to fail to decompose neatly.

> The singular value decomposition (SVD) provides another way to factorize a matrix, into singular vectors and singular values. The SVD allows us to discover some of the same kind of information as the eigendecomposition. However, the SVD is more generally applicable.
>
> — Pages 44-45, *Deep Learning*, 2016.

The SVD is used widely both in the calculation of other matrix operations, such as matrix inverse, but also as a data reduction method in machine learning. SVD can also be used in least squares linear regression, image compression, and denoising data.

> The singular value decomposition (SVD) has numerous applications in statistics, machine learning, and computer science. Applying the SVD to a matrix is like looking inside it with X-ray vision...
>
> — Page 297, *No Bullshit Guide To Linear Algebra*, 2017.

16.3 Calculate Singular-Value Decomposition

The SVD can be calculated by calling the `svd()` function. The function takes a matrix and returns the U, Σ and V^T elements. The Σ diagonal matrix is returned as a vector of singular values. The V matrix is returned in a transposed form, e.g. V^T. The example below defines a 3×2 matrix and calculates the singular-value decomposition.

```
# singular-value decomposition
from numpy import array
from scipy.linalg import svd
# define a matrix
A = array([
  [1, 2],
  [3, 4],
  [5, 6]])
print(A)
# factorize
U, s, V = svd(A)
print(U)
print(s)
print(V)
```

Listing 16.1: Example of calculating a singular-value decomposition.

Running the example first prints the defined 3×2 matrix, then the 3×3 U matrix, 2 element Σ vector, and 2×2 V^T matrix elements calculated from the decomposition.

```
[[1 2]
 [3 4]
 [5 6]]

[[-0.2298477  0.88346102 0.40824829]
 [-0.52474482 0.24078249 -0.81649658]
 [-0.81964194 -0.40189603 0.40824829]]

[ 9.52551809 0.51430058]

[[-0.61962948 -0.78489445]
 [-0.78489445 0.61962948]]
```

Listing 16.2: Sample output from calculating a singular-value decomposition.

16.4 Reconstruct Matrix

The original matrix can be reconstructed from the U, Σ, and V^T elements. The U, s, and V elements returned from the `svd()` cannot be multiplied directly. The s vector must be converted into a diagonal matrix using the `diag()` function. By default, this function will create a square matrix that is $m \times m$, relative to our original matrix. This causes a problem as the size of the matrices do not fit the rules of matrix multiplication, where the number of columns in a matrix must match the number of rows in the subsequent matrix. After creating the square Σ diagonal matrix, the sizes of the matrices are relative to the original $n \times m$ matrix that we are decomposing, as follows:

$$U(m \times m) \cdot \Sigma(m \times m) \cdot V^T(n \times n) \tag{16.2}$$

Where, in fact, we require:

$$U(m \times m) \cdot \Sigma(m \times n) \cdot V^T(n \times n) \tag{16.3}$$

We can achieve this by creating a new Σ matrix of all zero values that is $m \times n$ (e.g. more rows) and populate the first $n \times n$ part of the matrix with the square diagonal matrix calculated via `diag()`.

```
# reconstruct rectangular matrix from svd
from numpy import array
from numpy import diag
from numpy import zeros
from scipy.linalg import svd
# define matrix
A = array([
  [1, 2],
  [3, 4],
  [5, 6]])
print(A)
# factorize
U, s, V = svd(A)
# create m x n Sigma matrix
Sigma = zeros((A.shape[0], A.shape[1]))
# populate Sigma with n x n diagonal matrix
Sigma[:A.shape[1], :A.shape[1]] = diag(s)
# reconstruct matrix
B = U.dot(Sigma.dot(V))
print(B)
```

Listing 16.3: Example of reconstructing a rectangular matrix from a SVD.

Running the example first prints the original matrix, then the matrix reconstructed from the SVD elements.

```
[[1 2]
 [3 4]
 [5 6]]

[[ 1.  2.]
 [ 3.  4.]
 [ 5.  6.]]
```

Listing 16.4: Sample output from reconstructing a rectangular matrix from a SVD.

The above complication with the Σ diagonal only exists with the case where m and n are not equal. The diagonal matrix can be used directly when reconstructing a square matrix, as follows.

```
# reconstruct square matrix from svd
from numpy import array
from numpy import diag
from scipy.linalg import svd
# define matrix
A = array([
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]])
print(A)
# factorize
U, s, V = svd(A)
# create n x n Sigma matrix
```

```
Sigma = diag(s)
# reconstruct matrix
B = U.dot(Sigma.dot(V))
print(B)
```

Listing 16.5: Example of reconstructing a square matrix from a SVD.

Running the example prints the original 3×3 matrix and the version reconstructed directly from the SVD elements.

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

[[ 1.  2.  3.]
 [ 4.  5.  6.]
 [ 7.  8.  9.]]
```

Listing 16.6: Sample output from reconstructing a square matrix from a SVD.

16.5 Pseudoinverse

The pseudoinverse is the generalization of the matrix inverse for square matrices to rectangular matrices where the number of rows and columns are not equal. It is also called the Moore-Penrose Inverse after two independent discoverers of the method or the Generalized Inverse.

> Matrix inversion is not defined for matrices that are not square. [...] When A has more columns than rows, then solving a linear equation using the pseudoinverse provides one of the many possible solutions.
>
> — Page 46, *Deep Learning*, 2016.

The pseudoinverse is denoted as A^+, where A is the matrix that is being inverted and $+$ is a superscript. The pseudoinverse is calculated using the singular value decomposition of A:

$$A^+ = V \cdot D^+ \cdot U^T \tag{16.4}$$

Or, without the dot notation:

$$A^+ = V \cdot D^+ \cdot U^T \tag{16.5}$$

Where A^+ is the pseudoinverse, D^+ is the pseudoinverse of the diagonal matrix Σ and V^T is the transpose of V^T. We can get U and V from the SVD operation.

$$A = U \cdot \Sigma \cdot V^T \tag{16.6}$$

The D^+ can be calculated by creating a diagonal matrix from Σ, calculating the reciprocal of each non-zero element in Σ, and taking the transpose if the original matrix was rectangular.

$$\Sigma = \begin{pmatrix} s_{1,1} & 0 & 0 \\ 0 & s_{2,2} & 0 \\ 0 & 0 & s_{3,3} \end{pmatrix} \tag{16.7}$$