1 Singular Value Decomposition (SVD)

The singular value decomposition (SVD) is among the most important matrix factorizations of the computational era, providing a foundation for nearly all of the data methods in this book. The SVD provides a numerically stable matrix decomposition that can be used for a variety of purposes and is guaranteed to exist. We will use the SVD to obtain optimal low-rank approximations to matrices and to perform pseudo-inverses of non-square matrices to find a solution to the system of equations $\mathbf{Ax} = \mathbf{b}$. The SVD will also be used as the underlying algorithm of principal component analysis (PCA), where high-dimensional data is decomposed into its most statistically descriptive factors. SVD/PCA has been applied to a wide variety of problems in science and engineering.

In a sense, the SVD generalizes the concept of the fast Fourier transform (FFT), which will be the subject of the next chapter. Many engineering texts begin with the FFT, as it is the basis of many classical analytical and numerical results. However, the FFT works in idealized settings, and the SVD is a more generic data-driven technique. Because this book is focused on data, we begin with the SVD, which may be thought of as providing a basis that is *tailored* to the specific data, as opposed to the FFT, which provides a *generic* basis.

In many domains, complex systems will generate data that is naturally arranged in large matrices, or more generally in arrays. For example, a time series of data from an experiment or a simulation may be arranged in a matrix, with each column containing all of the measurements at a given time. If the data at each instant in time is multi-dimensional, as in a high-resolution simulation of the weather in three spatial dimensions, it is possible to reshape or *flatten* this data into a high-dimensional column vector, forming the columns of a large matrix. Similarly, the pixel values in a grayscale image may be stored in a matrix, or these images may be reshaped into large column vectors in a matrix to represent the frames of a movie. Remarkably, the data generated by these systems is typically low-rank, meaning that there are a few dominant patterns that explain the high-dimensional data. The SVD is a numerically robust and efficient method of extracting these patterns from data.

1.1 Overview

Here we introduce the singular value decomposition (SVD) and develop an intuition for how to apply the SVD by demonstrating its use on a number of motivating examples. The SVD will provide a foundation for many other techniques developed in this book, including classification methods in Chapter 5, the dynamic mode decomposition (DMD) in Chapter 7, and the proper orthogonal decomposition (POD) in Chapter 12. Detailed mathematical properties are discussed in the following sections.

High dimensionality is a common challenge in processing data from complex systems. These systems may involve large measured data sets including audio, image, or video data. The data may also be generated from a physical system, such as neural recordings from a brain, or fluid velocity measurements from a simulation or experiment. In many naturally occurring systems, it is observed that data exhibit dominant patterns, which may be characterized by a low-dimensional attractor or manifold [333, 334].

As an example, consider images, which typically contain a large number of measurements (pixels), and are therefore elements of a high-dimensional vector space. However, most images are highly compressible, meaning that the relevant information may be represented in a much lower-dimensional subspace. The compressibility of images will be discussed in depth throughout this book. Complex fluid systems, such as the Earth's atmosphere or the turbulent wake behind a vehicle, also provide compelling examples of the low-dimensional structure underlying a high-dimensional state space. Although high-fidelity fluid simulations typically require at least millions or billions of degrees of freedom, there are often dominant coherent structures in the flow, such as periodic vortex shedding behind vehicles or hurricanes in the weather.

The SVD provides a systematic way to determine a low-dimensional approximation to high-dimensional data in terms of dominant patterns. This technique is *data-driven* in that patterns are discovered purely from data, without the addition of expert knowledge or intuition. The SVD is numerically stable and provides a hierarchical representation of the data in terms of a new coordinate system defined by dominant correlations within the data. Moreover, the SVD is guaranteed to exist for any matrix, unlike the eigendecomposition.

The SVD has many powerful applications beyond dimensionality reduction of high-dimensional data. It is used to compute the pseudo-inverse of non-square matrices, providing solutions to under-determined or over-determined matrix equations, $\mathbf{A}\mathbf{x} = \mathbf{b}$. We will also use the SVD to de-noise data sets. The SVD is likewise important to characterize the input and output geometry of a linear map between vector spaces. These applications will all be explored in this chapter, providing an intuition for matrices and high-dimensional data.

Definition of the SVD

Generally, we are interested in analyzing a large data set $\mathbf{X} \in \mathbb{C}^{n\times m}$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}. \tag{1.1}$$

The columns $\mathbf{x}_k \in \mathbb{C}^n$ may be measurements from simulations or experiments. For example, columns may represent images that have been reshaped into column vectors with as many elements as pixels in the image. The column vectors may also represent the state of a physical system that is evolving in time, such as the fluid velocity at a set of discrete points, a set of neural measurements, or the state of a weather simulation with one square kilometer resolution.

The index k is a label indicating the kth distinct set of measurements. For many of the examples in this book, $\mathbf{X}$ will consist of a *time series* of data, and $\mathbf{x}_k = \mathbf{x}(k\Delta t)$. Often the *state dimension* n is very large, on the order of millions or billions of degrees of freedom.

The columns are often called *snapshots*, and m is the number of snapshots in $\mathbf{X}$. For many systems $n \gg m$, resulting in a *tall-skinny* matrix, as opposed to a *short-fat* matrix when $n \ll m$.

The SVD is a unique matrix decomposition that exists for every complex-valued matrix $\mathbf{X} \in \mathbb{C}^{n \times m}$:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{1.2}$$

where $\mathbf{U} \in \mathbb{C}^{n \times n}$ and $\mathbf{V} \in \mathbb{C}^{m \times m}$ are *unitary* matrices[1] with orthonormal columns, and $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ is a matrix with real, non-negative entries on the diagonal and zeros off the diagonal. Here * denotes the complex conjugate transpose.[2] As we will discover throughout this chapter, the condition that $\mathbf{U}$ and $\mathbf{V}$ are unitary is used extensively.

When $n \geq m$, the matrix $\mathbf{\Sigma}$ has at most m non-zero elements on the diagonal, and may be written as

$$\mathbf{\Sigma} = \begin{bmatrix} \hat{\mathbf{\Sigma}} \\ \mathbf{0} \end{bmatrix}.$$

Therefore, it is possible to *exactly* represent $\mathbf{X}$ using the *economy* SVD:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = \begin{bmatrix} \hat{\mathbf{U}} & \hat{\mathbf{U}}^\perp \end{bmatrix} \begin{bmatrix} \hat{\mathbf{\Sigma}} \\ \mathbf{0} \end{bmatrix} \mathbf{V}^* = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\mathbf{V}^*. \tag{1.3}$$

The full SVD and economy SVD are shown in Fig. 1.1. The columns of $\hat{\mathbf{U}}^\perp$ span a vector space that is complementary and orthogonal to that spanned by $\hat{\mathbf{U}}$. The columns of $\mathbf{U}$ are called *left singular vectors* of $\mathbf{X}$ and the columns of $\mathbf{V}$ are *right singular vectors*. The diagonal elements of $\hat{\mathbf{\Sigma}} \in \mathbb{C}^{m \times m}$ are called *singular values* and they are ordered from largest to smallest. The rank of $\mathbf{X}$ is equal to the number of non-zero singular values. We will show in Section 1.2 that the SVD can also be used to obtain an optimal rank-r approximation of $\mathbf{X}$ for $r < m$.

Computing the SVD

The SVD is a cornerstone of computational science and engineering, and the numerical implementation of the SVD is both important and mathematically enlightening. That said, most standard numerical implementations are mature and a simple interface exists in many modern computer languages, allowing us to abstract away the details underlying the SVD computation. For most purposes, we simply use the SVD as a part of a larger effort, and we take for granted the existence of efficient and stable numerical algorithms. Numerically, the SVD may be computed by first reducing the matrix $\mathbf{X}$ to a bidiagonal matrix and then using an iterative algorithm to compute the SVD of the bidiagonal matrix. For matrices with high aspect ratio (i.e., $n \gg m$), then the first step may be achieved by first computing a QR factorization to reduce $\mathbf{X}$ to an upper triangular matrix, followed by Householder reflections to reduce this upper triangular matrix into a bidiagonal form. The second step may be performed using a modified QR algorithm developed by Golub & Kahan [284]. Details of the QR factorization are beyond the scope of this book, although it is a straightforward greedy method that is closely related to Gram–Schmidt orthogonalization.

[1] A square matrix $\mathbf{U}$ is unitary if $\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}$.
[2] For real-valued matrices, this is the same as the regular transpose $\mathbf{X}^* = \mathbf{X}^T$.

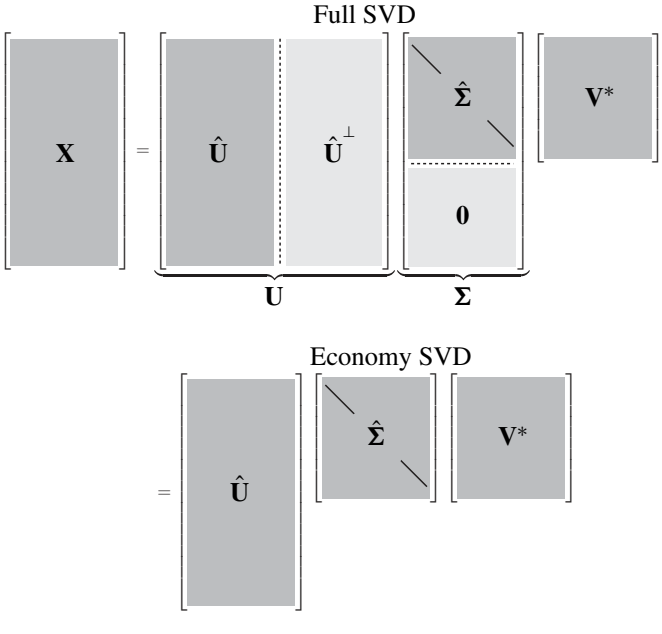

Figure 1.1 Schematic of matrices in the full and economy SVD.

There are numerous variations, alternatives, and important results on the computation of the SVD; for a more thorough discussion, see [143, 284, 285, 317, 388, 709]. Randomized numerical algorithms are increasingly used to compute the SVD of very large matrices as discussed in Section 1.8.

MATLAB

In MATLAB, computing the SVD is straightforward:

```
>> X = randn(5,3);    % Create a 5x3 random data matrix
>> [U,S,V] = svd(X); % Singular value decomposition
```

For non-square matrices $\mathbf{X}$, the economy SVD is more efficient:

```
>> [Uhat,Shat,V] = svd(X,'econ');  % Economy sized SVD
```

Python[3]

```
>>> import numpy as np
>>> X = np.random.rand(5, 3)  # Create random data matrix
>>> U, S, VT = np.linalg.svd(X,full_matrices=True) #Full SVD
>>> Uhat, Shat, VThat = np.linalg.svd(X,full_matrices=False)
      # Economy SVD
```

R

```
> X <- replicate(3, rnorm(5))
> s <- svd(X)
> U <- s$u
> S <- diag(s$d)
> V <- s$v
```

[3] Note that Python outputs the transpose of $\mathbf{V}$

Mathematica

```
In:= X=RandomReal[{0,1},{5,3}]
In:= {U,S,V} = SingularValueDecomposition[X]
```

Other Languages

The SVD is also available in other languages, such as Fortran and C++. In fact, most SVD implementations are based on the LAPACK (Linear Algebra Package) [19] in Fortran. The SVD routine is designated **DGESVD** in LAPACK, and this is wrapped in the C++ libraries **Armadillo** and **Eigen**.

Historical Perspective

The SVD has a long and rich history, ranging from early work developing the theoretical foundations to modern work on computational stability and efficiency. There is an excellent historical review by Stewart [674], which provides context and many important details. The review focuses on the early theoretical work of Beltrami and Jordan (1873), Sylvester (1889), Schmidt (1907), and Weyl (1912). It also discusses more recent work, including the seminal computational work of Golub and collaborators [284, 285]. In addition, there are many excellent chapters on the SVD in modern texts [24, 419, 709].

Uses in This Book and Assumptions of the Reader

The SVD is the basis for many related techniques in dimensionality reduction. These methods include principal component analysis (PCA) in statistics [338, 339, 550], the Karhunen–Loève transform (KLT) [373, 452], empirical orthogonal functions (EOFs) in climate [458], the proper orthogonal decomposition (POD) in fluid dynamics [334], and canonical correlation analysis (CCA) [175]. Although developed independently in a range of diverse fields, many of these methods only differ in how the data is collected and pre-processed. There is an excellent discussion about the relationship between the SVD, the KLT, and PCA by Gerbrands [274].

The SVD is also widely used in system identification and control theory to obtain reduced-order models that are balanced in the sense that states are hierarchically ordered in terms of their ability to be observed by measurements and controlled by actuation [508].

For this chapter, we assume that the reader is familiar with linear algebra, with some experience in computation and numerics. For review, there are a number of excellent books on numerical linear algebra, with discussions on the SVD [24, 419, 709].

1.2 Matrix Approximation

Perhaps the most useful and defining property of the SVD is that it provides an *optimal* low-rank approximation to a matrix $\mathbf{X}$. In fact, the SVD provides a *hierarchy* of low-rank approximations, since a rank-r approximation is obtained by keeping the leading r singular values and vectors, and discarding the rest.

Because $\mathbf{\Sigma}$ is diagonal, it is possible to express the matrix $\mathbf{X} = \mathbf{U\Sigma V}^*$ as a sum of rank-one matrices:

$$\mathbf{X} = \sum_{k=1}^{m} \sigma_k \mathbf{u}_k \mathbf{v}_k^* = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \cdots + \sigma_m \mathbf{u}_m \mathbf{v}_m^*, \tag{1.4}$$

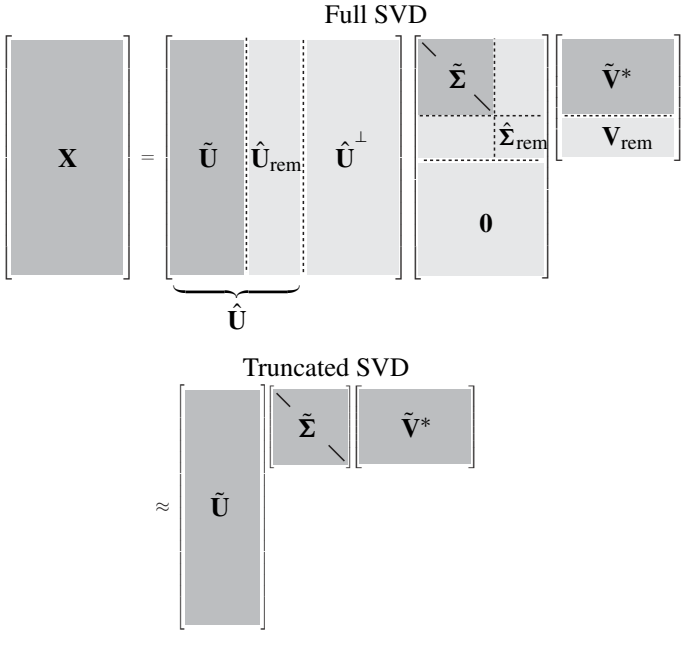

Figure 1.2 Schematic of truncated SVD. The subscript '$_{\text{rem}}$' denotes the remainder of $\hat{\mathbf{U}}$, $\hat{\mathbf{\Sigma}}$, or $\mathbf{V}$ after truncation.

where σ_k is the kth diagonal entry of $\mathbf{\Sigma}$, and $\mathbf{u}_k$ and $\mathbf{v}_k$ are the kth columns of $\mathbf{U}$ and $\mathbf{V}$, respectively. This is known as the *dyadic* summation. The singular values σ_k are arranged in decreasing order, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$, so each subsequent rank-one matrix $\sigma_k \mathbf{u}_k \mathbf{v}_k^*$ is less important than the previous matrix in capturing the information in $\mathbf{X}$. For many systems, the singular values σ_k decrease rapidly, and it is possible to obtain a good approximation of $\mathbf{X}$ by truncating at some rank r:

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^* = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^*. \tag{1.5}$$

Here, we establish the notation that a truncated SVD basis (and the resulting approximated matrix $\tilde{\mathbf{X}}$) will be denoted by $\tilde{\mathbf{X}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$, where $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ contain the first r columns of $\mathbf{U}$ and $\mathbf{V}$, and $\tilde{\mathbf{\Sigma}}$ contains the first $r \times r$ sub-block of $\mathbf{\Sigma}$. The truncated SVD is illustrated in Fig. 1.2, with $\tilde{\mathbf{U}}$, $\tilde{\mathbf{\Sigma}}$, and $\tilde{\mathbf{V}}$ denoting the truncated matrices. If $\mathbf{X}$ does not have full rank, then some of the singular values in $\hat{\mathbf{\Sigma}}$ may be zero, and the truncated SVD may still be exact. However, for truncation values r that are smaller than the number of non-zero singular values (i.e., the rank of $\mathbf{X}$), the truncated SVD only approximates $\mathbf{X}$.

For a given rank r, there is no better approximation for $\mathbf{X}$, in the ℓ_2 sense, than the truncated SVD approximation $\tilde{\mathbf{X}}$. The Eckart–Young theorem below will state this precisely and provide expressions for the error of the truncated approximation. There are numerous choices for the truncation rank r, and they are discussed in Section 1.7. Thus, high-dimensional data may be well described by a few dominant patterns given by the columns of $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$.

This is an important property of the SVD, and we will return to it many times. There are numerous examples of data sets that contain high-dimensional measurements, resulting

in a large data matrix $\mathbf{X}$. However, there are often dominant low-dimensional patterns in the data, and the truncated SVD basis $\tilde{\mathbf{U}}$ provides a coordinate transformation from the high-dimensional measurement space into a low-dimensional pattern space. This has the benefit of *reducing* the size and dimension of large data sets, yielding a tractable basis for visualization and analysis. Finally, many systems considered in this text are *dynamic* (see Chapter 7), and the SVD basis provides a hierarchy of modes that characterize the observed attractor, on which we may project a low-dimensional dynamical system to obtain reduced-order models (see Chapter 13).

Optimal Approximation and Error Bounds

Schmidt (of Gram–Schmidt) generalized the SVD to function spaces and developed an approximation theorem, establishing the truncated SVD $\tilde{\mathbf{X}}$ as the optimal low-rank approximation of the underlying matrix $\mathbf{X}$ [637]. Schmidt's approximation theorem was rediscovered by Eckart and Young [227], and is sometimes referred to as the Eckart–Young theorem.

Theorem 1.1 (Eckart–Young [227]) *The optimal rank-r approximation to* $\mathbf{X}$*, in a least-squares sense, is given by the rank-r SVD truncation* $\tilde{\mathbf{X}}$*:*

$$\underset{\tilde{\mathbf{X}},\ \text{s.t. rank}(\tilde{\mathbf{X}})=r}{\operatorname{argmin}} \quad \|\mathbf{X}-\tilde{\mathbf{X}}\|_F = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*. \tag{1.6}$$

Again, $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ denote the first r leading columns of $\mathbf{U}$ and $\mathbf{V}$, and $\tilde{\mathbf{\Sigma}}$ contains the leading $r \times r$ sub-block of $\mathbf{\Sigma}$. The Frobenius norm above is defined as $\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|X_{ij}|^2}$, which is equivalent to the 2-norm of the vectorized matrix $\mathbf{X}(:)$.

Thus, the Eckart–Young theorem guarantees that the truncated SVD provides the best matrix approximation of a given rank in the Frobenius norm. It is also possible to exactly quantify the error of the rank-r SVD approximation:

$$\|\mathbf{X}-\tilde{\mathbf{X}}\|_F^2 = \sum_{k=r+1}^{m}\sigma_k^2. \tag{1.7}$$

Thus, all other rank-r matrices $\tilde{\mathbf{X}}$ will have at least this much error. Because the error scales with the size and magnitude of $\mathbf{X}$, it is often more useful to consider the relative error

$$\frac{\|\mathbf{X}-\tilde{\mathbf{X}}\|_F^2}{\|\mathbf{X}\|_F^2}. \tag{1.8}$$

This expression for the relative error in the Frobenius norm has two intuitive interpretations. If the columns of $\mathbf{X}$ are velocity fields, for example from a discretized fluid flow simulation, then this error is related to the fraction of the *kinetic energy* that is missing in the approximation $\tilde{\mathbf{X}}$. More generally, the squared Frobenius norm error of mean-subtracted data has the interpretation of the amount of missing variance in the approximation $\tilde{\mathbf{X}}$. This statistical interpretation will be explored more in Section 1.5.

Remarkably, the SVD also provides an optimal rank-r approximation in the matrix 2-norm, also known as the spectral norm:

$$\underset{\tilde{\mathbf{X}},\ \text{s.t. rank}(\tilde{\mathbf{X}})=r}{\operatorname{argmin}} \quad \|\mathbf{X}-\tilde{\mathbf{X}}\|_2 = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*. \tag{1.9}$$

The 2-norm of a matrix $\mathbf{X}$ is induced by the vector 2-norm and is given by

$$\|\mathbf{X}\|_2 = \max_{\mathbf{v}\neq\mathbf{0}} \frac{\|\mathbf{X}\mathbf{v}\|_2}{\|\mathbf{v}\|_2}.$$

The error expression for the rank-r SVD approximation is even simpler in the 2-norm:

$$\|\mathbf{X} - \tilde{\mathbf{X}}\|_2 = \sigma_{r+1}. \tag{1.10}$$

This error expression is rather simple to derive by expanding

$$\mathbf{X} - \tilde{\mathbf{X}} = \sum_{k=r+1}^{m} \sigma_k \mathbf{u}_k \mathbf{v}_k^*. \tag{1.11}$$

Since each of the $\mathbf{v}_k$ vectors is orthonormal, the maximum $\|(\mathbf{X}-\tilde{\mathbf{X}})\mathbf{v}\|_2 = \sigma_{r+1}$ is achieved for $\mathbf{v} = \mathbf{v}_{r+1}$.

Example: Image Compression

We demonstrate the idea of matrix approximation with a simple example: image compression. A recurring theme throughout this book is that large data sets often contain underlying patterns that facilitate low-rank representations. Natural images present a simple and intuitive example of this inherent *compressibility*. A grayscale image may be thought of as a real-valued matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, where n and m are the number of pixels in the vertical and horizontal directions, respectively.[3] Depending on the basis of representation (pixel space, Fourier frequency domain, SVD transform coordinates), images may have very compact approximations.

Consider the image of Mordecai the snow dog in Fig. 1.3. This image has 2000×1500 pixels. It is possible to take the SVD of this image and plot the diagonal singular values, as in Fig. 1.4. Figure 1.3 shows the approximate matrix $\tilde{\mathbf{X}}$ for various truncation values r. By $r = 100$, the reconstructed image is quite accurate, and the singular values account for almost 80% of the total cumulative sum of the singular values. The squared error is less than 4% in the Frobenius norm. The SVD truncation results in a compression of the original image, since only the first 100 columns of $\mathbf{U}$ and $\mathbf{V}$, along with the first 100 diagonal elements of $\mathbf{\Sigma}$, must be stored in $\tilde{\mathbf{U}}$, $\tilde{\mathbf{\Sigma}}$, and $\tilde{\mathbf{V}}$.

Code 1.1 [MATLAB] Use SVD to compress image.

```
% First, we load the image
A=imread('../DATA/dog.jpg');
X=double(rgb2gray(A)); % Convert RBG->gray, 256 bit->double.
nx = size(X,1); ny = size(X,2);
imagesc(X), axis off, colormap gray

% Take the SVD
[U,S,V] = svd(X);

% Approximate matrix with truncated SVD for various ranks r
for r=[5 20 100]   % Truncation value
    Xapprox = U(:,1:r)*S(1:r,1:r)*V(:,1:r)'; % Approx. image
    figure, imagesc(Xapprox), axis off
    title(['r=',num2str(r,'%d')]);
end
```

[3] It is not uncommon for image size to be specified as horizontal by vertical, i.e., $\mathbf{X}^T \in \mathbb{R}^{m\times n}$, although we stick with vertical by horizontal to be consistent with generic matrix notation.

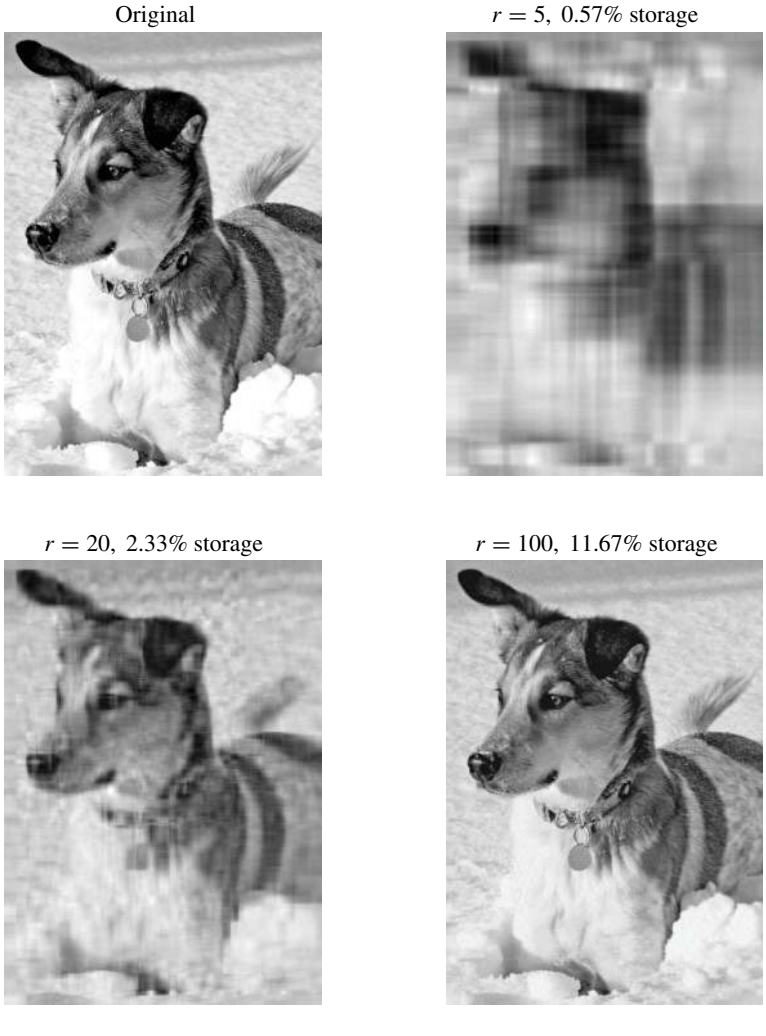

Figure 1.3 Image compression of Mordecai the snow dog, truncating the SVD at various ranks r. Original image resolution is 2000×1500.

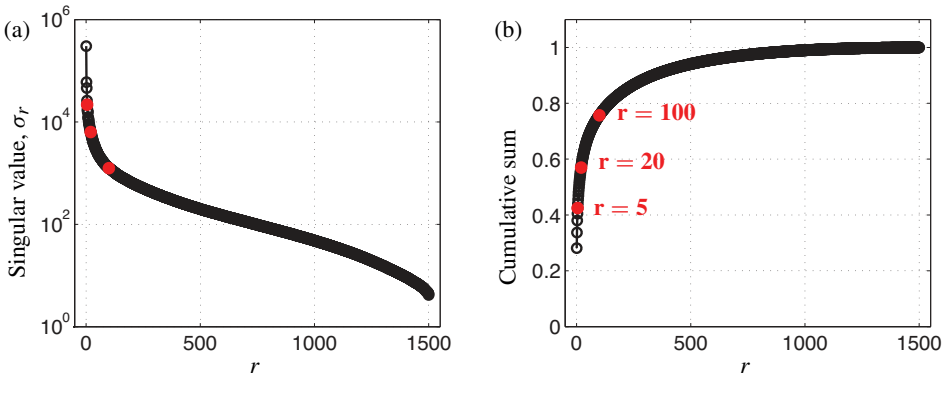

Figure 1.4 (a) Singular values σ_r and (b) cumulative sum $\sum_{k=1}^{r} \sigma_k$ of the first r singular values.

```
% Plot singular values and cumulative sum
subplot(1,2,1), semilogy(diag(S),'k')
subplot(1,2,2), plot(cumsum(diag(S))/sum(diag(S)),'k')
```

Code 1.1 [Python] Use SVD to compress image.

```
# First, we load the image
from matplotlib.image import imread
A = imread(os.path.join('..','DATA','dog.jpg'))
X = np.mean(A, -1); # Convert RGB to grayscale
img = plt.imshow(X)
# Take the SVD
U, S, VT = np.linalg.svd(X,full_matrices=False)
S = np.diag(S)
# Approximate matrix with truncated SVD for various ranks r
for r in (5, 20, 100):    # Construct approximate image
    Xapprox = U[:,:r] @ S[0:r,:r] @ VT[:r,:]
    img = plt.imshow(Xapprox)
    plt.show()
# Plot singular values and cumulative sum
plt.semilogy(np.diag(S))
plt.plot(np.cumsum(np.diag(S))/np.sum(np.diag(S)))
```

1.3 Mathematical Properties and Manipulations

Here we describe important mathematical properties of the SVD, including geometric interpretations of the unitary matrices $\mathbf{U}$ and $\mathbf{V}$, as well as a discussion of the SVD in terms of dominant correlations in the data $\mathbf{X}$. The relationship between the SVD and correlations in the data will be explored more in Section 1.5 on principal component analysis.

Interpretation as Dominant Correlations

The SVD is closely related to an eigenvalue problem involving the correlation matrices $\mathbf{XX}^*$ and $\mathbf{X}^*\mathbf{X}$, shown in Fig. 1.5 for a specific image, and in Figs. 1.6 and 1.7 for generic matrices. If we plug (1.3) into the row-wise correlation matrix $\mathbf{XX}^*$ and the column-wise correlation matrix $\mathbf{X}^*\mathbf{X}$, we find

$$\mathbf{XX}^* = \mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}\\ \mathbf{0}\end{bmatrix}\mathbf{V}^*\mathbf{V}\begin{bmatrix}\hat{\mathbf{\Sigma}} & \mathbf{0}\end{bmatrix}\mathbf{U}^* = \mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{U}^*, \tag{1.12a}$$

$$\mathbf{X}^*\mathbf{X} = \mathbf{V}\begin{bmatrix}\hat{\mathbf{\Sigma}} & \mathbf{0}\end{bmatrix}\mathbf{U}^*\mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}\\ \mathbf{0}\end{bmatrix}\mathbf{V}^* = \mathbf{V}\hat{\mathbf{\Sigma}}^2\mathbf{V}^*. \tag{1.12b}$$

Recalling that $\mathbf{U}$ and $\mathbf{V}$ are unitary, $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ are solutions to the following eigenvalue problems:

$$\mathbf{XX}^*\mathbf{U} = \mathbf{U}\begin{bmatrix}\hat{\mathbf{\Sigma}}^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \tag{1.13a}$$

$$\mathbf{X}^*\mathbf{XV} = \mathbf{V}\hat{\mathbf{\Sigma}}^2. \tag{1.13b}$$