2 Dimensionality Reduction

High-dimensional data refers to datasets with a large number of variables, features, or attributes describing each observation. In many cases, the number of dimensions (features) can be in the hundreds, thousands, or even millions, as seen in areas such as genomics, image processing, and text analysis. 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. While high-dimensional data can capture rich and detailed information, it also introduces unique challenges—often called the "curse of dimensionality"—where the sheer number of features makes patterns harder to detect, increases computational demands, and can lead to overfitting in machine learning models. In many naturally occurring systems, it is observed that data exhibit dominant patterns, which may be characterized by a low-dimensional attractor or manifold [4]. Analyzing high-dimensional data effectively requires techniques such as dimensionality reduction, feature selection, or regularization to identify the most relevant information while preserving the underlying structure of the data.

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. 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 [4].

To simplify the dimensionality reduction concept, Fig(17) shows an artwork by Tim Noble and sue Webster. This artwork can be linked to the concept of dimensionality reduction in data analysis. The pile of trash in the foreground represents the raw, high-dimensional data—messy, complex, and seemingly without clear meaning when viewed directly. However, when illuminated from the right perspective, its shadow on the wall reveals a clear, simplified image of two people, analogous to how dimensionality reduction techniques project complex, high-dimensional datasets into a lower-dimensional space that preserves the essential structure or meaning. In the same way the shadow filters out irrelevant details and organizes the chaos into a recognizable form, dimensionality reduction extracts the most important features from data, making patterns and relationships easier to interpret while discarding noise.

Interestingly, several online retailers use dimensionality reduction methods in

Figure 17: "Dirty White Trash (With Gulls)" artwork by Tim Noble and Sue Webster to simplify the concept of dimensionality reduction.(a) is a pile of trash in the foreground that represents the raw, high-dimensional data—messy, complex, and seemingly without clear meaning when viewed directly. (b) It is a shadow that filters out irrelevant details and organizes the chaos into a recognizable form.

their recommendation lists. For example, Netflix and Amazon use dimensionality reduction to deal with the huge and complex data linking millions of users to thousands of movies or products. Instead of working with a massive table full of mostly empty spaces, they use methods like Singular Value Decomposition (SVD) to shrink the data into two smaller tables: one for users and one for items. In this smaller space, each movie or product is described by a few hidden features—like a taste for "sci-fi thrillers" or "romantic comedies"—and each user is described by how strongly they match those features. The system then recommends items whose features are most similar to what a user likes. This makes the process faster, handles missing data well, and powers things like Netflix's "Because you watched..." suggestions and Amazon's "Customers who bought this also bought..." lists.

In summary, the dimensionality reduction methods provide 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. In this section, two of the most common methods used for dimensionality reduction applications are covered.

2.1 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a fundamental matrix factorization technique with numerous applications across various fields such as mathematics, statistics, engineering, signal processing, machine learning, and more. 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. It allows us to extract and untangle correlated information from high-dimensional datasets. Moreover, the SVD is guaranteed to exist for any matrix, unlike the eigendecomposition. In the following, the graphical and mathematical representations of the SVD method will be explored to provide a deeper understanding. The graphical view offers an intuitive way to see SVD as a sequence of rotations and scalings that transform one space into another. This perspective helps build a clear mental model of how SVD acts on data, making it easier to understand applications such as dimensionality reduction, noise filtering, and latent feature extraction. The mathematical view then complements this intuition by presenting the formal decomposition of a matrix $\mathbf{A}$ into $\mathbf{U\Sigma V}^*$, revealing the precise structure of the method, clarifying the role of each component, and enabling detailed calculations, property analysis, and connections to related concepts like eigenvalues, orthogonality, and rank. By combining both perspectives, you gain not only the computational ability to apply SVD but also the conceptual insight to interpret its results and adapt it effectively to a variety of real-world problems in data analysis and machine learning.

2.1.1 Graphical Representation

A graphical explanation of Singular Value Decomposition (SVD) shows how it transforms data step by step, making the abstract algebra easier to visualize. In even the earliest experience of linear algebra, the concept of a matrix transforming a vector via multiplication was defined. For instance, the vector $\mathbf{v}$ when multiplied by a matrix $\mathbf{A}$ produces a new vector ω that is now aligned, generically, in a new direction with a new length. To be more specific, and as shown in Fig(18,a,b), the following example illustrates a particular transformation [5]:

$$\mathbf{v} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} \qquad \Longrightarrow \qquad \omega = \mathbf{Av} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

Thus generically, matrix multiplication will rotate and stretch (compress) a given vector as prescribed by the matrix $\mathbf{A}$.
The rotation and stretching of a transformation can be precisely controlled by the proper construction of the matrix $\mathbf{A}$. In particular, it is well known that in a two-dimensional space, the rotation matrix:

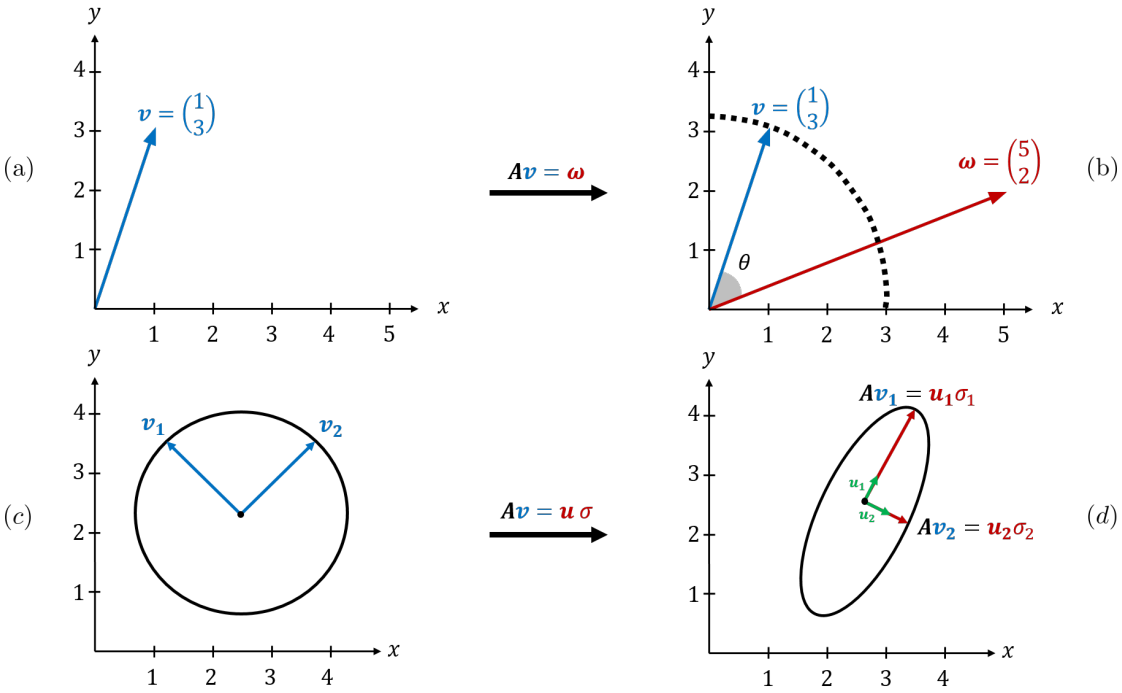

Figure 18: Graphical Representation of the SVD.

$$\mathbf{A}_{\text{Rotation}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

takes a given vector $\mathbf{v}$ and rotates it by an angle θ to produce the vector ω. Thus, rotation can be directly specified without the vector being scaled. To scale the vector in length, the matrix:

$$\mathbf{A}_{\text{Scaling}} = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}$$

can be applied to the vector $\mathbf{v}$. This multiplies the length of the vector $\mathbf{v}$ by α. If $\alpha = 2(0.5)$, then the vector is twice (half) its original length. The combination of the above two matrices gives arbitrary control of rotation and scaling in a two-dimensional vector space. The same applies to shapes such as a circle, shown in Fig(18,c), defined by the basis vectors $\mathbf{v_1}, \mathbf{v_2}$, then any linear transformation by a matrix $\mathbf{A}$ would change it to an ellipse, shown in Fig(18,d) as a result of rotation and stretching which is defined by the transformed new basis vectors $\mathbf{Av_1} = \mathbf{u_1}\sigma_1$ and $\mathbf{Av_2} = \mathbf{u_2}\sigma_2$ where σ_1, σ_2 represent the amount of scaling applied on the original vectors. This results in a system of linear equations:

$$\mathbf{Av_1} = \mathbf{u_1}\sigma_1 \qquad \text{and} \qquad \mathbf{Av_2} = \mathbf{u_2}\sigma_2$$

Formulating these two equations in a matrix format:

$$\mathbf{A}\begin{bmatrix} | & | \\ \mathbf{v_1} & \mathbf{v_2} \\ | & | \end{bmatrix} = \begin{bmatrix} | & | \\ \mathbf{u_1} & \mathbf{u_2} \\ | & | \end{bmatrix}\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

Solving for matrix $\mathbf{A}$, gives:

$$\mathbf{A} = \begin{bmatrix} | & | \\ \mathbf{u_1} & \mathbf{u_2} \\ | & | \end{bmatrix}\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}\begin{bmatrix} - & \mathbf{v_1^*} & - \\ - & \mathbf{v_2^*} & - \end{bmatrix}$$

The vectors $\mathbf{u_i}$ are called the left singular vectors, while the vectors $\mathbf{v_i}$ are called the right singular vectors, and σ_i are the singular values (which in this case represent the length of the major and minor axes of the ellipse). Here $*$ denotes the complex conjugate of the vectors $\mathbf{v_i}$. As shown in Fig(19), you can think of SVD as breaking a transformation into three parts: first, a rotation (matrix $\mathbf{V}^*$) that aligns the data with new coordinate axes; second, a scaling (matrix Σ) that stretches or compresses the data along these axes according to the singular values; and finally, another rotation (matrix $\mathbf{U}$) that reorients the scaled data into its final position.

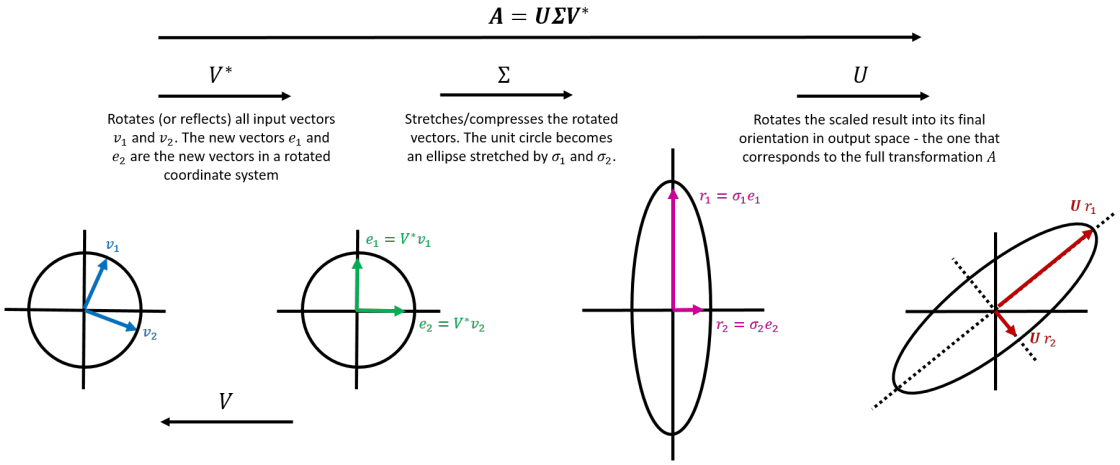

Figure 19: Graphical Representation of the SVD methods and the transformation effect of each matrix on a circle.

Generalizing the SVD expression for a linear transformation $\mathbf{A}$ that maps vectors from an n-dimensional space to an n-dimensional space gives:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^* \tag{38}$$

This visual representation helps you see that SVD isn't just numbers—it's a geometric process that changes orientation and scale, which is why it's so useful in applications such as dimensionality reduction, noise removal, and uncovering latent structures in datasets.

2.1.2 Mathematical Representation

Generally, we are interested in analyzing a large dataset $\mathbf{A}(m \times n)$:

$$\mathbf{A} = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix} \tag{39}$$

The columns of $\mathbf{A}$ 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 a resolution of one square kilometer [4]. Often, the state dimension is very large, on the order of millions or billions of degrees of freedom. The columns are called snapshots, and n is the number of snapshots in $\mathbf{A}$.
For many systems, $m \gg n$. In this case, the singular value decomposition (SVD) exists for any complex-valued matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ as a unique factorization:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{40}$$

where $\mathbf{U} \in \mathbb{C}^{m\times m}$ and $\mathbf{V} \in \mathbb{C}^{n\times n}$ are unitary matrices [1] with orthonormal columns[2] and they are called the left and right singular vectors of $\mathbf{A}$, respectively. The columns of $\mathbf{U}$ are hierarchically arranged, e.g., $\mathbf{U}_1$ is more important than $\mathbf{U}_2$, in terms of their ability to describe the variance in matrix $\mathbf{A}$. Here $*$ denotes the complex conjugate transpose [3]. The matrix $\mathbf{\Sigma} \in \mathbb{C}^{m\times n}$ is diagonal with positive real entries and they are called singular values and they are ordered from largest to smallest such that $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_{\min(m,n)} \geq 0$. The full SVD of the matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ is shown in Fig(20,a).

When $m \gg n$, the system is overdetermined—that is, it has many more equations (rows) than unknowns (columns). Because $\mathbf{A}$ has only n columns, it can have at most n linearly independent columns. Consequently,only the first n columns of

[1] A square matrix $\mathbf{U}$ is unitary if $\mathbf{U}\mathbf{U}^* = \mathbf{U}^*\mathbf{U} = \mathbf{I}$

[2] Orthonormal columns (or equivalently rows) are vectors that they all have norm 1 and the inner product of any distinct pair of them is 0.

[3] For real-value matrices, this is the same as the regular transpose $\mathbf{V}^* = \mathbf{V}^T$

U are required to to represent and describe **A**'s action. The remaining columns of **U** still exist to make **U** a square orthogonal matrix, but they don't contribute to **A**. They correspond to directions that **A** never reaches. Such directions are often referred to as "silent" components or the "null-space" directions of **A**. A similar reasoning applies to the $\mathbf{\Sigma}$ which contains at most n nonzero diagonal elements. The diagonals beyond the nth row, correspond to directions in **U** that do not map from any input — there's no data there — so those entries must be zero. Because of this redundancy, it is possible to exactly represent **A** in a reduced version using the *economy* (or reduced) SVD, as shown in Fig(20,b) and can be written as follows:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = \begin{bmatrix}\hat{\mathbf{U}} & \hat{\mathbf{U}}^{\perp}\end{bmatrix}\begin{bmatrix}\hat{\mathbf{\Sigma}} \\ \mathbf{0}\end{bmatrix}\mathbf{V}^* = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\mathbf{V}^* \tag{41}$$

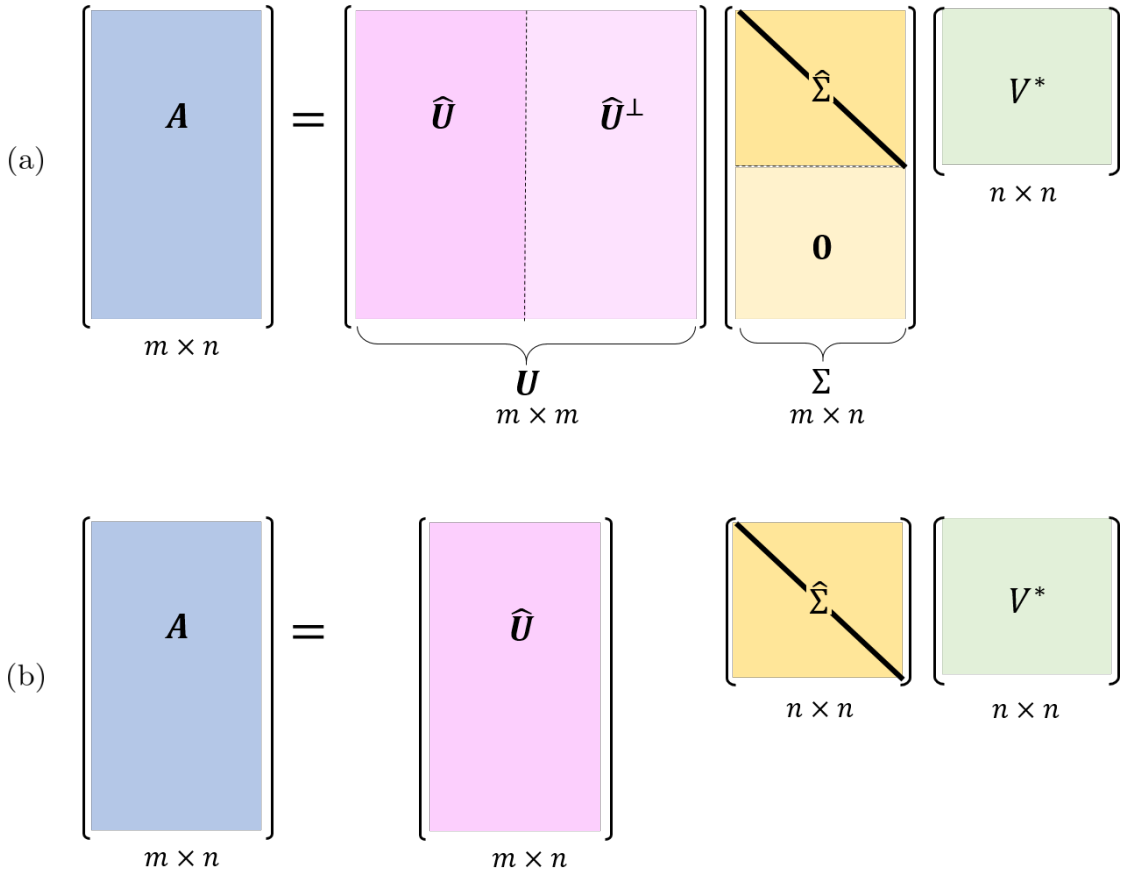

Figure 20: Schematic of matrices in the full, in (a), and economy, in (b), SVD.

The above theorem guarantees the existence of the SVD, but in practice, it remains

to be computed. This is a fairly straightforward process if one considers the following matrix products:

$$\begin{aligned}\mathbf{A}^*\mathbf{A} &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*)^* (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*) \\ &= \mathbf{V}\mathbf{\Sigma}\mathbf{U}^*\mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \\ &= \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^* \end{aligned} \tag{42}$$

and

$$\begin{aligned}\mathbf{A}\mathbf{A}^* &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*) (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*)^* \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*\mathbf{V}\mathbf{\Sigma}\mathbf{U}^* \\ &= \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^* \end{aligned} \tag{43}$$

Multiplying Eq.(42) and Eq.(43) on the right by $\mathbf{V}$ and $\mathbf{U}$, respectively, gives the following two self-consistent eigenvalue problems:

$$\mathbf{A}^*\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{\Sigma}^2 \tag{44a}$$

$$\mathbf{A}\mathbf{A}^*\mathbf{U} = \mathbf{U}\mathbf{\Sigma}^2 \tag{44b}$$

Thus, if the normalized eigenvectors are found for these two equations, then the orthonormal basis vectors are produced for $\mathbf{V}$ and $\mathbf{U}$. Likewise, the square root of the eigenvalues of these equations produces the singular values σ_i.
Let's have a simple 2×2 example to clarify this. Consider the following matrix:

$$\mathbf{A} = \begin{bmatrix} 3 & 0 \\ 4 & 5 \end{bmatrix}$$

We begin with calculating $\mathbf{A}^*\mathbf{A}$:

$$\mathbf{A}^*\mathbf{A} = \begin{bmatrix} 3 & 4 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix}$$

The eigenvalues σ^2, which will be denoted as $\sigma^2 = \lambda$ for simplicity, can be calculated as:

$$
\begin{aligned}
|\mathbf{A}^*\mathbf{A} - \lambda\mathbb{I}| &= \begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \\
&= \begin{bmatrix} 25-\lambda & 20 \\ 20 & 25-\lambda \end{bmatrix} \\
&= 0
\end{aligned}
$$

Computing this gives two eigenvalues $\lambda_1 = \sigma_1^2 = 45$ and $\lambda_2 = \sigma_2^2 = 5$. Therefore, we can say that $\sigma_1 = \sqrt{45}$ and $\sigma_2 = \sqrt{5}$. Let's move on to calculate the eigenvectors $\mathbf{V}$ of $\mathbf{A}^*\mathbf{A}$ (or the right singular vectors of $\mathbf{A}$):

$$
\text{For} \quad \sigma_1 = \sqrt{45}: \qquad \mathbf{A}^*\mathbf{A}\mathbf{V_1} = \sigma^2\mathbf{V_1}
$$

$$
\begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 45 \begin{bmatrix} 1 \\ 1 \end{bmatrix}
$$

$$
\text{For} \quad \sigma_2 = \sqrt{5}: \qquad \mathbf{A}^*\mathbf{A}\mathbf{V_2} = \sigma^2\mathbf{V_2}
$$

$$
\begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = 5 \begin{bmatrix} -1 \\ 1 \end{bmatrix}
$$

Remember that both $\mathbf{V_1}$ and $\mathbf{V_2}$ should be **orthonormal** and therefore, they need to be normalized and rescaled to length 1. Thus, we rewrite them as:

$$
\mathbf{V_1} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \mathbf{V_2} = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}
$$

Now, let's calculate $\mathbf{A}\mathbf{A}^*$:

$$
\mathbf{A}\mathbf{A}^* = \begin{bmatrix} 3 & 0 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ 0 & 5 \end{bmatrix} = \begin{bmatrix} 9 & 12 \\ 12 & 41 \end{bmatrix}
$$

Calculating the eigenvalues of $\mathbf{A}\mathbf{A}^*$ results in similar eigenvalues as $\mathbf{A}^*\mathbf{A}$, i.e., $\sigma_1 = \sqrt{45}$ and $\sigma_2 = \sqrt{5}$. Let's move on then to calculate the eigenvectors $\mathbf{U}$ of $\mathbf{A}\mathbf{A}^*$ (or the left singular vectors of $\mathbf{A}$):

$$\text{For} \quad \sigma_1 = \sqrt{45}: \qquad \mathbf{A}\mathbf{A}^*\mathbf{U_1} = \sigma^2\mathbf{U_1}$$
$$\begin{bmatrix} 9 & 12 \\ 12 & 41 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = 45 \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

$$\text{For} \quad \sigma_2 = \sqrt{5}: \qquad \mathbf{A}\mathbf{A}^*\mathbf{U_2} = \sigma^2\mathbf{U_2}$$
$$\begin{bmatrix} 9 & 12 \\ 12 & 41 \end{bmatrix} \begin{bmatrix} -3 \\ 1 \end{bmatrix} = 5 \begin{bmatrix} -3 \\ 1 \end{bmatrix}$$

Normalising $\mathbf{U_1}$ and $\mathbf{U_2}$ gives:

$$\mathbf{U_1} = \frac{1}{\sqrt{10}} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \qquad \mathbf{U_2} = \frac{1}{\sqrt{10}} \begin{bmatrix} -3 \\ 1 \end{bmatrix}$$

As a result, the Singular Value Decomposition of the matrix $\mathbf{A}$ can be written as:

$$\mathbf{U} = \frac{1}{\sqrt{10}} \begin{bmatrix} 1 & -3 \\ 3 & 1 \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} \sqrt{45} & 0 \\ 0 & \sqrt{5} \end{bmatrix} \qquad \mathbf{V} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

By computing $\sigma_1\mathbf{U_1}\mathbf{V_1}^* + \sigma_2\mathbf{U_2}\mathbf{V_2}^*$, one can verify that this indeed reconstructs the original matrix $\mathbf{A}$.

For large datasets, the SVD can be easily computed in MATLAB or Python via a single line of code with one command!

2.2 Principal Component Analysis (PCA)

Principal component analysis (PCA), also called proper mode decomposition (POD), Hotelling Transform, empirical component analysis (EOF), reduced order modeling (ROM), and karhunen-loéve decomposition, is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components [9, 4, 5]. PCA is one of the central uses of SVD, providing a data-driven, hierarchical coordinate system to represent high-dimensional correlated data. The main question that PCA always tries to answer is: can low dimensional reductions of the dynamics and behavior be produced from seemingly complex, noisy, correlated,

redundant, perhaps random data when the governing equations are not known? PCA can quantify low dimensional dynamics arising in such areas as turbulent fluid flows [10], structural vibration [11, 12], damage detection [13], and neural decision-making strategies [14], to name just a few application areas. As we move forward on this topic, it will also become obvious that the breadth of applications is staggering and includes image processing and single analysis.

To clarify the idea behind the PCA algorithm and its key associated concepts with the SVD, a simple model example of a spring-mass system will be examined, as shown in Fig(21). This is a fairly easy problem that can be solved using the second Newton's law of motion $\mathbf{F} = m\mathbf{a}$.

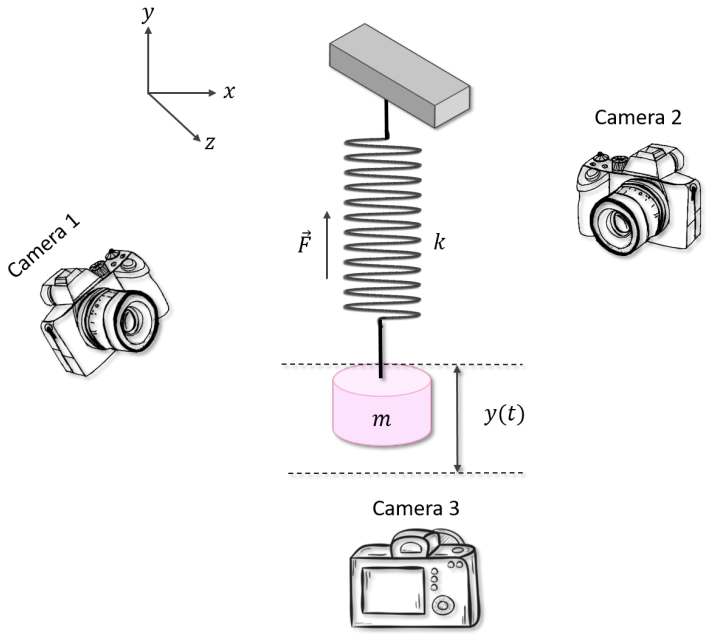

Figure 21: Spring-mass system. The mass m is suspended with a spring with Hooke's constant k and oscillates with a displacement $y(t)$. The motion data of this system is captured by three cameras from different angles.

The spring-mass system is a simple experiment: a mass m is suspended by a spring with Hooke's constant k. If the mass is perturbed or taken from equilibrium in the y-direction only with a displacement of $y(t)$, we know that the governing equation simply starts from Newton's second law as follows:

$$\mathbf{F} = m\mathbf{a}$$

It's known that the force $\mathbf{F}$ is the Hooke's force pointing upwards, opposite to the spring's motion, and the acceleration $\mathbf{a}$ is the second derivative of the displacement

$y(t)$. Therefore, one can write:

$$-ky(t) = m\frac{d^2y(t)}{dt^2} \Longrightarrow \frac{d^2y(t)}{dt^2} = -\frac{k}{m}y(t) = -\omega^2 y(t) \Longrightarrow \frac{d^2y(t)}{dt^2} + \omega^2 y(t) = 0$$

This is a homogeneous second-order ordinary differential equation. To solve it, we get the auxiliary equation which is $\lambda^2 + \omega^2 = 0$. The solutions of this equation are complex values, i.e., $\lambda = \alpha \pm i\beta = 0 \pm i\omega$. Therefore, the general solution would take the form of:

$$y(t) = e^{\alpha t}\left[A\cos(\omega t) + B\sin(\omega t)\right]$$

As a result, one can write the general solution of this system as:

$$y(t) = A\cos(\omega t) + B\sin(\omega t)$$

This solution essentially states that the state of the system can be described as a one-degree-of-freedom system. The question here now is: suppose that we don't know the governing equation of this system, can we know that this system has only one degree of freedom just by collecting measurements of its motion? To answer this question, we need a dataset for a start! Let's prove this using the PCA algorithm following the next steps.

1. **Data Collection**. Assume that the mass m oscillates in the y-direction only after applying a small perturbation on its motion. Thus the mass starts to oscillate. Three cameras are then used, as shown in Fig(21), to record the motion from different angles. Each camera produces a three-dimensional representation of the data. If we denote the data from the three cameras with the subscripts a, b, and c, then the data collected are represented by the following:

   $$\text{Camera 1} : (\mathbf{x}_a, \mathbf{y}_a, \mathbf{z}_a) \qquad \text{Camera 2} : (\mathbf{x}_b, \mathbf{y}_b, , \mathbf{z}_b) \qquad \text{Camera 3} : (\mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$$

   where each set $(\mathbf{x}_j, \mathbf{y}_j, \mathbf{z}_j)$ consists of three vectors of data collected over time with each having a length n. Indeed, one can imagine that most of this data will be quite noisy, perhaps redundant, and certainly not produced from an optimal viewpoint. But through the process of PCA, these can be circumvented to extract the ideal or simplified behavior.

   All the data collected can then be gathered into a single matrix:

$$\mathbf{X} = \begin{bmatrix} — & \mathbf{x}_a & — \\ — & \mathbf{y}_a & — \\ — & \mathbf{z}_a & — \\ — & \mathbf{x}_b & — \\ — & \mathbf{y}_b & — \\ — & \mathbf{z}_b & — \\ — & \mathbf{x}_c & — \\ — & \mathbf{y}_c & — \\ — & \mathbf{z}_c & — \end{bmatrix} \tag{45}$$

Thus the matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ where m represents the number of measurements and n is the number of data points taken from the camera over time. Fig(22) shows the acceleration data obtained from a simple harmonic oscillator in three dimensions from three different orientations. It is clear from the obtained data that the dynamics have one degree of freedom along one axis. Let's see if PCA can prove this.

2. **Compute the mean**. The second step is to calculate the mean $\overline{\mathbf{x}}_j$, i.e., the mean across the axis that contains each observation.

$$\overline{\mathbf{x}}_i = \frac{1}{n} \sum_{j=1}^{n} \mathbf{X}_{ij} \qquad \Longrightarrow \qquad \overline{\mathbf{X}} = \begin{bmatrix} \overline{\mathbf{x}}_1 \\ \overline{\mathbf{x}}_2 \\ \vdots \\ \overline{\mathbf{x}}_m \end{bmatrix} \tag{46}$$

3. **Subtract $\overline{\mathbf{X}}$ from $\mathbf{X}$**. This results a mean-subtracted dataset $\mathbf{B}$ that is centered around the origin:

$$\mathbf{B} = \mathbf{X} - \overline{\mathbf{X}} \tag{47}$$

4. **Construct the covariance matrix $\mathbf{C}$**. After centering the dataset, the covariance matrix is computed as follows:

$$\mathbf{C} = \frac{1}{n-1} \mathbf{B}\mathbf{B}^* \tag{48}$$

where $\dfrac{1}{n-1}$ is the normalization constant. The covariance matrix is a symmetric square $m \times m$ matrix whose diagonal elements represent the variance of the collected measurements. Large variances correspond to the *dynamics of interest*, whereas low variances are assumed to correspond to *uninteresting*

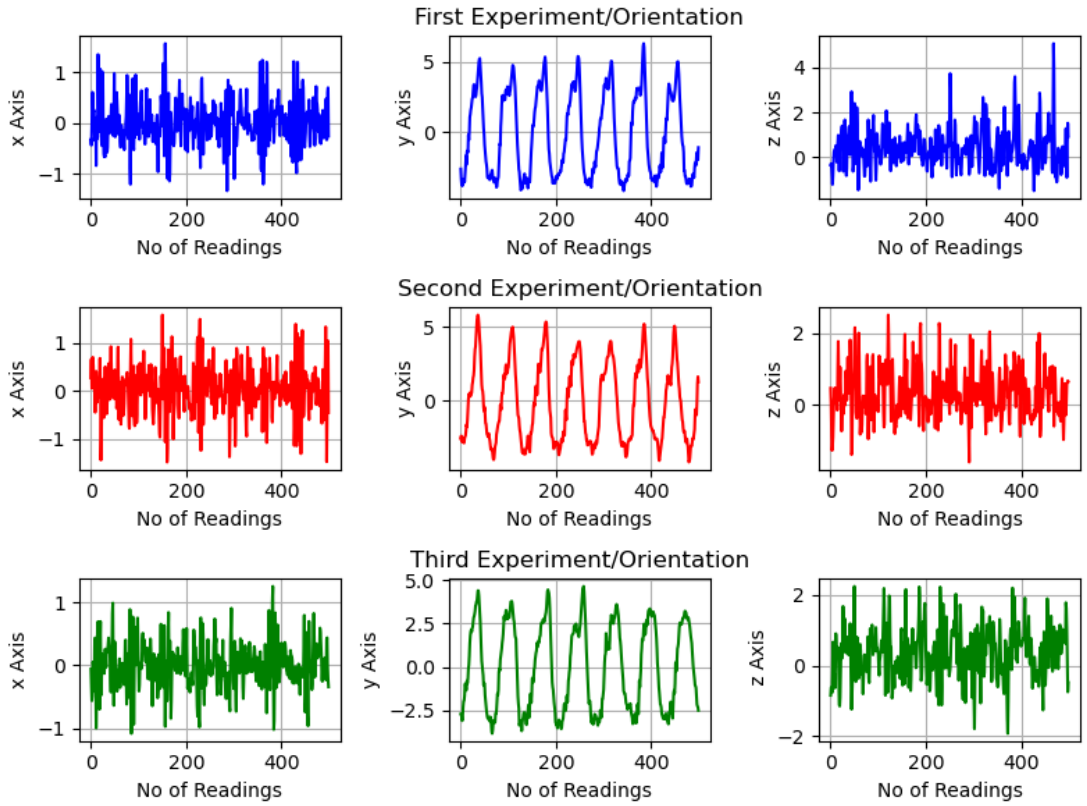

Figure 22: The 3D experimental acceleration data collected from a simple harmonic oscillator from three different orientations. The software used to collect the data is PhyPhox.

dynamics. The off-diagonal elements are the covariances between each pair of measurements. These elements tell us about the correlations between the variables. A large off-diagonal term represents two events that have a high degree of correlation, whereas a small off-diagonal coefficient means that there is little correlation in the data. This means that the off-diagonal terms represent the redundancy (or the dependency) in the measurements. The insight given by the covariance matrix leads to our ultimate goals:

- removing redundancy, and hence reducing the dimensionality of the data by keeping only the important dimensions.
- identifying those signals with maximal variance that hold the most important information about the studied system.

In a mathematical sense, we are simply asking to represent matrix $\mathbf{C}$ so that the

diagonals are ordered from largest to smallest and the off-diagonals are zero, i.e., our task is to diagonalize the covariance matrix.

5. **Compute the the eigenvalues and eigenvectors of the covariance matrix C**. The concept of diagonalization is critical for understanding the underpinnings of many physical and engineering systems. In this process of diagonalization, the correct coordinates, or basis functions, are revealed to reduce the given system to its low-dimensional essence. The eigendecomposition of the covariance matrix $\mathbf{C}$ can be written as:

$$\mathbf{CV} = \mathbf{V\Lambda} \qquad \Longrightarrow \qquad \mathbf{C} = \mathbf{V\Lambda V}^{-1} \tag{49}$$

where $\mathbf{V}$ is a $m \times m$ matrix of the eigenvectors of the covariance matrix arranged in columns that represent the directions of the axes where there is the most variance (most information). Whereas, $\mathbf{\Lambda}$ is a $m \times m$ diagonal matrix whose entries correspond to the eigenvalues that are simply the coefficients attached to eigenvectors, which give the amount of variance carried in each eigenvector.

6. **Sort and select**. The main goal of using PCA is the creation of fewer dimensions that hold the most important information about the system, i.e., the highest variance. Therefore, it is essential to first rank the eigenvectors in order of their eigenvalues, highest to lowest variance, so that they are in order of significance, as shown in Eq.(50).

$$\mathbf{V} = \begin{bmatrix} | & | & \cdots & | & \cdots & | \\ v_1 & v_2 & \cdots & v_k & \cdots & v_m \\ | & | & \cdots & | & \cdots & | \end{bmatrix} \quad \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \lambda_k & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_m \end{bmatrix} \tag{50}$$

Here, the eigenvalues are sorted in a way that $\lambda_1 > \lambda_2 > \cdots > \lambda_k > \cdots \lambda_m$. To implement the concept of reducing the dimensionality of the system, the top k eigenvectors, $\mathbf{V}_k$, and their corresponding eigenvalues, λ_k are selected, where k is the number of dimensions we want to keep, as in Eq.(51). The truncated slice of eigenvectors represents the basis of the new orthogonal and independent subspace, i.e., the subspace of principal components that hold most of the system's information.

$$\mathbf{V}_k = \begin{bmatrix} | & | & \cdots & | \\ v_1 & v_2 & \cdots & v_k \\ | & | & \cdots & | \end{bmatrix} \quad \mathbf{\Lambda}_k = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_k \end{bmatrix} \tag{51}$$

7. **Project the data along the principal components**. The end goal here is to transform the original, noisy, redundant, and high-dimensional dataset $\mathbf{X}$ onto the new orthogonal and independent subspace, i.e., the subspace of principal components or PCA scores (or modes). This suggests that instead of working directly with the matrix $\mathbf{X}$, we consider working with its transformed low-dimensional version, or the principal component basis. To compute the principal components of the dataset,

$$\mathbf{Y} = \mathbf{V}_k^T \, \mathbf{B} \tag{52}$$

The matrix $\mathbf{Y}$ is a $m \times k$ matrix that contains k principal components (or columns) that hold rich information about the behavior and the dynamics of the given physical system.

Fig(23, left) shows the distribution of the eigenvalues of this system. It is clear that only one eigenvalue holds up to more than 70% of the information of the dynamics. This eigenvalue corresponds to the single degree of freedom the system exhibits. Additionally, if we only look at the first three PCA scores of the covariance matrix, as shown in Fig(23, right), it can be shown that the first score reflects the oscillatory behavior of the harmonic system. However, the other two scores barely have noticeable variations, around 0. The first PCA score corresponds to the one degree of freedom of the harmonic system, whereas the other two scores correspond to the redundancy (or noise) associated with the collected experimental data. More on how to code this can be found here.

2.3 PCA vs SVD

So far, we have seen that principal component analysis (PCA) and singular value decomposition (SVD) are two fundamental techniques in linear algebra and data analysis. They play a crucial role in reducing the dimensionality of data and extracting essential information. Understanding their differences and the role of SVD in PCA empowers data scientists to choose the right technique for their specific tasks. These differences can be briefly listed as follows:

- **Mathematical Relationship:** PCA can be seen as a specific application of SVD. PCA primarily deals with the covariance structure of the data. It's a

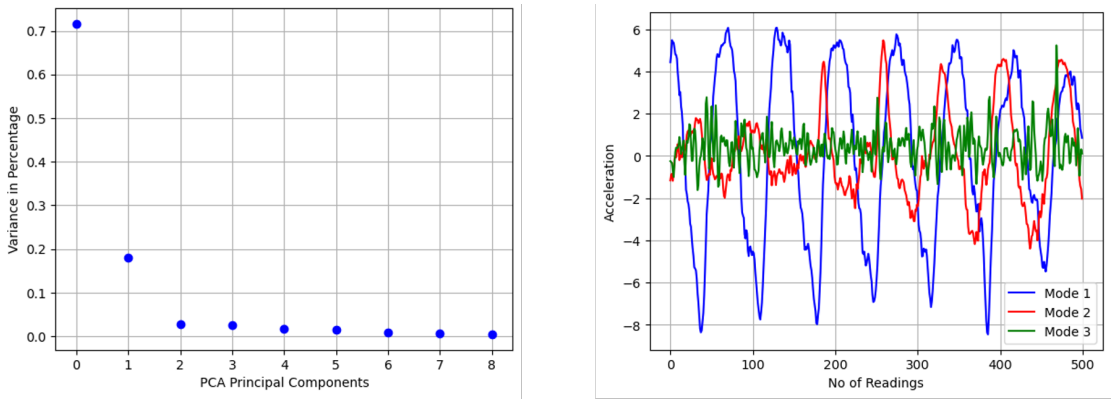

Figure 23: (Left): The eigenvalues of the covariance matrix of the simple harmonic oscillator. (Right): The first three PCA scores of the system.

statistical procedure that transforms the coordinates of a dataset into a new coordinate system. SVD, on the other hand, does not rely on a covariance matrix. It is a factorization of the original data matrix, and it decomposes the original data into three matrices. The principal components obtained through PCA are essentially the left singular vectors of the data matrix.

- **Purpose:** PCA is primarily used for dimensionality reduction and visualization, while SVD has broader applications, including matrix approximation and recommendation systems.
- **Orthogonality:** Both PCA and SVD produce orthogonal vectors (components) that capture uncorrelated directions of maximum variance.
- **Eigenvalues vs. Singular Values:** PCA involves computing eigenvectors and eigenvalues of the covariance matrix, while SVD directly deals with singular values. Note that eigenvalues are only defined for square matrices, whereas singular values are defined for all matrices, and that is why SVD is seen as a more general matrix decomposition technique with broader applications in linear algebra and data analysis.
- **Convergence:** Taking the SVD of the mean-subtracted matrix $\mathbf{X}$ results in the PCA.

2.4 Choosing the Optimal Truncation Order

When applying dimensionality reduction techniques such as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD), the choice of truncation order—that is, the number of components or singular values to retain—is a critical

step. A common approach is to examine the explained variance ratio (in PCA) or the singular value spectrum (in SVD) and select the smallest number of components that capture a sufficient proportion of the total variance or energy of the data, often between 90–99% depending on the application. This ensures that most of the essential information is preserved while discarding noise and redundancy. For example: Suppose a PCA is applied to a dataset and the explained variance ratios of the first few principal components are: PC1 = 55%, PC2 = 25%, PC3 = 12%, PC4 = 5%, PC5 = 2%, with the remaining components each contributing $< 1\%$. By summing these values, we see that the first three components already capture 92% of the total variance $(55 + 25 + 12)$, which may be sufficient for most applications. Therefore, a reasonable truncation order would be 3, as adding more components contributes only marginal improvements while increasing computational cost.

Alternatively, the "elbow method" can be used, where the truncation order is selected at the point of diminishing returns in the variance explained or energy captured, as shown in Fig.(24). In practice, this corresponds to identifying where adding additional components results in only marginal improvements in performance or reconstruction accuracy. The choice of truncation can also be influenced by computational constraints or the requirements of downstream tasks (for example, maintaining classification accuracy, reducing reconstruction error, or enabling real-time computation).

A higher truncation rank generally produces a model that more closely approximates the original matrix A and thus achieves greater accuracy. However, this comes at the expense of increased model complexity, as retaining a larger number of singular values requires more memory and computational resources. On the other hand, choosing a lower truncation rank reduces complexity and computational demands, but the resulting model may lose critical information, leading to reduced accuracy. Striking the right balance is therefore essential: in some applications, such as data compression or noise filtering, a lower truncation rank may be desirable, whereas in tasks that require high precision, such as medical image analysis or safety-critical engineering systems, a higher truncation rank may be justified.

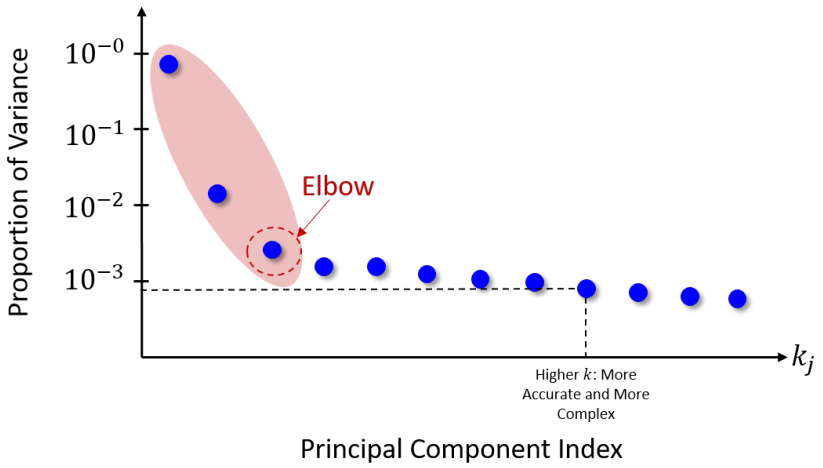

Figure 24: Elbow Method for choosing the optimal number of singular values.