Singular value decomposition

Non-Square Matrix

Let's first understand what a non-square matrix is, with an example, and how it relates to linear transformations.
A non-square matrix is a matrix where the number of rows and columns are not equal. These matrices are important in transforming vectors from one dimensional space to another.

Example 1: Transforming 2D Vectors to 3D Space

Consider the matrix:

$$A = \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 3 & 1 \end{bmatrix}$$

This is a 3×2 matrix. That means it takes 2D vectors as input and transforms them into 3D vectors.
Now, consider unit vectors in 2D space:

- i = (1,0)
- j = (0,1)

Each column of the matrix tells us where these basis vectors go after the transformation:

- The first column $[1,2,3]^T$ is the transformed vector of i.
- The second column $[0,0,1]^T$ is the transformed vector of j.

So, this matrix takes the 2D input space and maps it to 3D space. The original 2D coordinate system is thus embedded into a 3D coordinate system.

- Input Space: Dimension where the original vectors lie (here, 2D).
- Output Space: Dimension where the transformed vectors lie (here, 3D).

Example 2: Transforming 3D Vectors to 2D Space

Now consider another matrix:

$$B = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

This is a 2×3 matrix. It takes 3D vectors as input and transforms them into 2D vectors.
Let's multiply it with a 3D vector:

$$\begin{bmatrix}1 & 3 & 5\\ 2 & 4 & 6\end{bmatrix}.\begin{bmatrix}1\\ 2\\ 3\end{bmatrix} = \begin{bmatrix}1(1) & 3(2) & 5(3)\\ 2(1) & 4(2) & 6(3)\end{bmatrix} = \begin{bmatrix}22\\ 28\end{bmatrix}$$

The result is a 2×1 vector, showing that the 3D input has been projected into a 2D space.

General Concept

If we have a matrix of size m × n:

- n is the dimension of the input vector (input space).
- m is the dimension of the output vector (output space).

So, any non-square matrix A ∈ $R^{m\times n}$ performs a linear transformation from an n-dimensional vector space to an m-dimensional vector space.

Rectangular Diagonal Matrix

Let's understand what a rectangular diagonal matrix is, along with an example.
A rectangular diagonal matrix is a non-square matrix where the non-zero elements lie along a diagonal, and it typically combines two operations during a linear transformation:

1. Dimension reduction or projection, and
2. Scaling of the components.

Example 1: Composition of Two Transformations

Consider the matrix:

$$\begin{bmatrix}a & 0 & 0\\ 0 & b & 0\end{bmatrix}$$

This is a 2×3 matrix. We can break this transformation into two steps:

$$\begin{bmatrix}a & 0\\ 0 & b\end{bmatrix}.\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$$

- The first matrix $\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$ is a projection from 3D space to 2D space.
- The second matrix $\begin{bmatrix}a & 0\\ 0 & b\end{bmatrix}$ performs scaling along the new 2D axes.

So, the full transformation first reduces the dimension from 3D to 2D, then scales the vector components by a and b respectively.

Intuition

The matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ is a 2×3 matrix. It selects the first two coordinates of a 3D vector, effectively projecting it onto the x–y plane (discarding the z-coordinate). Then, the diagonal matrix $\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$ scales the x and y components individually.

Example 2: Extended Case

Now consider a slightly different matrix:

$$\begin{bmatrix} a & 0 \\ 0 & b \\ 0 & 0 \end{bmatrix}$$

This is a 3×2 matrix.

We can decompose it as:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

- The first matrix is 3×2 and projects the scaled vector back into 3D space.
- The second matrix is 2×2 and performs the scaling.
  In this case:
- The scaling happens first,
- Then, the embedding or projection into 3D happens by appending a zero in the third dimension.

When dealing with a rectangular diagonal matrix:

- It represents a composition of two transformations.
- You can often interpret the transformation as:
  Scaling → Projection or Embedding (depending on matrix shape).

Whenever you see a rectangular diagonal matrix of size m × n, remember:

- If m < n , it's a projection (dimension reduction).
- If m > n , it's an embedding (adding extra dimensions with zero padding).
- The diagonal values scale the relevant components.

What is SVD?

Singular Value Decomposition (SVD) is a matrix factorization technique that decomposes any real or complex matrix into three other matrices:

$$A = U\Sigma V^{T}$$

Where:

- A is the original m × n matrix.
- U is an m × m orthogonal matrix whose columns are the left singular vectors.
- Σ is an m × n diagonal matrix with singular values (non-negative real numbers) on the diagonal.

- $V^\top$ is the transpose of an n × n orthogonal matrix V, whose columns are the right singular vectors.

Applications of SVD

1. Machine Learning and Data Science
    - SVD is used in Principal Component Analysis (PCA) for dimensionality reduction, especially when dealing with high-dimensional data.
    - It's widely applied in recommendation systems, like the Netflix recommendation algorithm.
2. Natural Language Processing (NLP)
    - In Latent Semantic Analysis (LSA), SVD helps extract semantic meaning from large text corpora by reducing the dimensions of term-document matrices.
3. Computer Vision
    - SVD is used for image compression. By keeping only the top singular values, an image can be reconstructed with less storage but high visual fidelity.
4. Signal Processing
    - Helps in noise reduction and signal separation, which is useful in communication systems and audio signal processing.
5. Numerical Linear Algebra
    - Used for matrix inversion, especially for matrices that are ill-conditioned or not invertible, making it a numerically stable method.
6. Psychometrics
    - In psychology and education, SVD is used to extract latent traits from psychological or educational test data.
7. Bioinformatics
    - SVD helps analyze gene expression data, identifying underlying patterns of gene activity.
8. Quantum Computing
    - Applied in quantum state tomography to analyze and reconstruct quantum states.

SVD – The Equation and Its Relationship with Eigen Decomposition

Now the question is: what are U, Σ, and V in SVD?
The answer lies in eigen decomposition.

Eigen Decomposition:

If we have a square matrix, we can break it down as:

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

Where:

- A is an n × n matrix
- V is the matrix of eigenvectors
- Λ is the diagonal matrix of eigenvalues

Special Case: Symmetric Matrix

If A is not only square but also symmetric, then:

- V becomes orthogonal
- $V^{-1} = V^T$
- Λ remains a diagonal matrix

So the decomposition becomes:

$$A = V\Lambda V^T$$

Connecting Eigen Decomposition with SVD

SVD Equation:

$$A = U\Sigma V^T$$

Here, A can be non-square. That's the power of SVD — it works for any real m × n matrix.

Now we explore the relationship between this SVD equation and eigen decomposition.

To do this, we convert A into a square symmetric matrix using the trick of multiplying it with its transpose:

Step 1: Compute AA^T

$$AA^T = (U\Sigma V^T)(U\Sigma V^T)^T$$

$$AA^T = U\Sigma V^T V\Sigma^T U^T$$

Since V is orthogonal, V^TV = I, so:

$$AA^T = U\Sigma\Sigma^T U^T$$

Let:

X = $\Sigma\Sigma^T$

Then:

$$AA^T = UXU^T$$

This is the eigen decomposition of AA^T, where:

- U contains the eigenvectors of AA^T
- X = $\Sigma\Sigma^T$ contains the eigenvalues

Step 2: Compute A^TA

$$A^TA = (U\Sigma V^T)^T(U\Sigma V^T)$$

$$A^TA = V\Sigma^TU^TU\Sigma V^T$$

$$A^TA = V\Sigma^T\Sigma V^T$$

Let:

$$Y = \Sigma^T\Sigma$$

Then:

$$A^TA = VYV^T$$

This is the eigen decomposition of A^TA, where:

- V contains the eigenvectors of A^TA
- Y = $\Sigma^T\Sigma$ contains the eigenvalues

Summary of Results:

$$AA^T = UXU^T \quad \text{where } X = \Sigma\Sigma^T$$

$$A^TA = VYV^T \quad \text{where } Y = \Sigma^T\Sigma$$

Both AA^T and A^TA Aare symmetric matrices and their decompositions resemble eigen decomposition.

Interpretation of U and V

Now, going back to our original SVD equation:

$$A = U\Sigma V^T$$

We ask: What are U and V?

- U : Columns are eigenvectors of AA^T
- V : Columns are eigenvectors of A^TA

So we now understand:

- U is derived from AA^T
- V is derived from A^TA

These matrices are not directly related to the original A , but rather indirectly through the symmetric matrices AA^T and A^TA.

Singular Vectors and Values

With respect to A :

- u and v are called singular vectors

- Both U and V are orthogonal matrices
- Σ contains the singular values (square roots of eigenvalues of AA^T or A^TA)
- U : Left singular vectors
- V : Right singular vectors

Now the question arises: What is Σ in SVD?

We know from the SVD (Singular Value Decomposition) that:

$$X = \Sigma\Sigma^T \quad \text{or} \quad X = \Sigma^T\Sigma$$

Now, let's recall eigen decomposition:

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

Here, Λ is a diagonal matrix whose entries are the eigenvalues of matrix A.
In the context of SVD, the matrices X and Y refer to:

- $X = AA^T$
- $Y = A^TA$

These matrices share non-zero eigenvalues.

Let's take an example with a 2 × 3 matrix A.
So, the SVD of A is:

A = $U\Sigma V^T$

Where:

- U is a 2 × 2 matrix (left singular vectors),
- Σ is a 2 × 3 rectangular diagonal matrix (singular values),
- V^T is a 3 × 3 matrix (right singular vectors).

Let's define Σ :

$$\Sigma = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \end{bmatrix}$$

Now we compute:
X = $\Sigma\Sigma^T$

$$X = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & b \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} a^2 & 0 \\ 0 & b^2 \end{bmatrix}$$

Y = $\Sigma^T\Sigma$

$$Y = \begin{bmatrix} a & 0 \\ 0 & b \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \end{bmatrix} = \begin{bmatrix} a^2 & 0 & 0 \\ 0 & b^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

What do X and Y represent?

- X = $\Sigma\Sigma^T$ is a diagonal matrix whose entries are the eigenvalues of AA^T.
- Y = $\Sigma^T\Sigma$ is a diagonal matrix whose entries are the eigenvalues of A^TA.

So:

- X and Y have eigenvalues a^2, b^2 (and 0 in the case of Y, which we can ignore for simplicity).
- Taking square roots of these eigenvalues gives us a and b, which are the singular values.

Thus, in SVD:

- The diagonal entries of Σ , i.e., a and b, are the square roots of the eigenvalues of either AA^T or A^TA.
- These are called the singular values.

Final SVD Summary:

$$A = U\Sigma V^T$$

- U: Columns are the left singular vectors, which are the eigenvectors of AA^T
- V: Columns of V (before transposing) are the right singular vectors, which are eigenvectors of A^TA
- Σ: Contains the singular values a , b , which are the square roots of eigenvalues of AA^T or A^TA

Geometric Intuition of SVD: Understanding A = $\boldsymbol{U\Sigma V^T}$

Let's build geometric intuition behind the decomposition:

$$A = U\Sigma V^T$$

Where A is a matrix that applies a linear transformation to vector space. This transformation can be broken into three parts:

1. V^T – Rotation (or reflection) of the coordinate system
2. Σ – Stretching or scaling along the new axes
3. U – Another rotation (or reflection) applied to the stretched result

Together, these create the full transformation represented by matrix A .

Case: Symmetric Matrix

If A is symmetric, we can write it as:

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

- Λ is a diagonal matrix of eigenvalues
- V is an orthogonal matrix (its columns are the eigenvectors)

For symmetric matrices, SVD and eigen decomposition align, and you can clearly visualize the transformation.

Transformation Steps (Visualized)

Assume the symmetric matrix:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

And a standard 2D Cartesian plane.

Step 1: Apply V^{-1}

This rotates the vector space counterclockwise—since V is orthogonal, the rotation maintains the 90° angle between basis vectors.

Step 2: Apply Λ

Now, the vectors are stretched along the new axes (eigenvectors) by the corresponding eigenvalues. Each axis is scaled independently—this is the heart of PCA and spectral analysis.

Step 3: Apply V

This rotates the stretched space clockwise, restoring alignment with the original basis but with stretched axes because applying V after Λ undoes the initial rotation (V^{-1}).

Interpretation

- The original x-axis is rotated to a new position (the red line in your visual).
- The y-axis similarly rotates to its new orientation.
- After scaling, and final rotation, we get the transformed version of space that matrix A represents.

This layered understanding of matrix transformation is super helpful for applications like PCA, image compression, and more.

Understanding SVD Geometrically: Case of a 2 × 3 Matrix

Let's consider a matrix A ∈ $R^{2\times3}$, and understand how the transformation works step-by-step when we apply:

$$A = U\Sigma V^T$$

We're operating in two spaces here:

- The input space (3D, because A has 3 columns).
- The output space (2D, because A has 2 rows).

Step 1: Apply V^T — Rotate in Input Space (3D)

- V is the matrix of eigenvectors of A^TA.
- Since $A^TA \in R^{3\times3}$, it lives in 3D space.
- Hence, V ∈ $R^{3\times3}$, and it's an orthogonal matrix (its vectors are at 90° angles to each other).
- When we apply V^T, we rotate the basis vectors in the 3D input space. This

rotation aligns our vectors along the red lines shown in the diagram.

Step 2: Apply Σ — Dimension Reduction + Stretching

- The singular value matrix Σ for a 2 × 3 matrix looks like this:

$$\Sigma = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \end{bmatrix}$$

Breaking Σ into Σ_2 and Σ_1 helps separate two effects: projection from 3D to 2D, and independent stretching along new 2D axes.
We break this down into two conceptual matrices:

- $\Sigma_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$: This step **removes the z-component**, reducing vectors from 3D to 2D (projecting onto the XY-plane).
- $\Sigma_1 = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$: Applies **scaling/stretching** in the 2D plane along x and y directions.

Step 3: Apply U— Final Rotation in Output Space (2D)

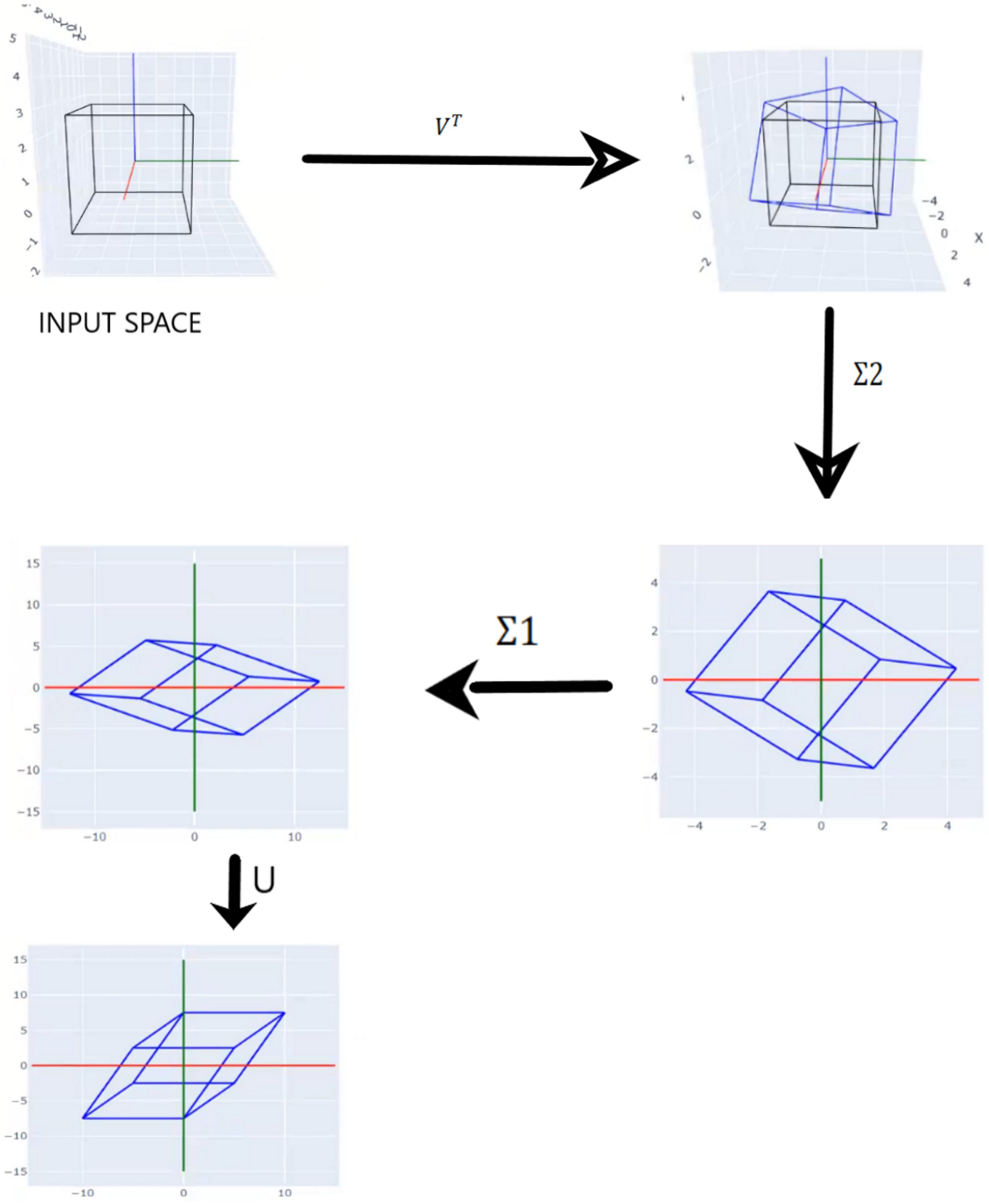
- U is the matrix of eigenvectors of AA^T, which is 2×2, so its vectors live in 2D.
- U ∈ $R^{2\times2}$, and it's also orthogonal.
- It rotates the already reduced and stretched vectors clockwise, bringing them back onto the original black axes.

Summary: Transformation Flow

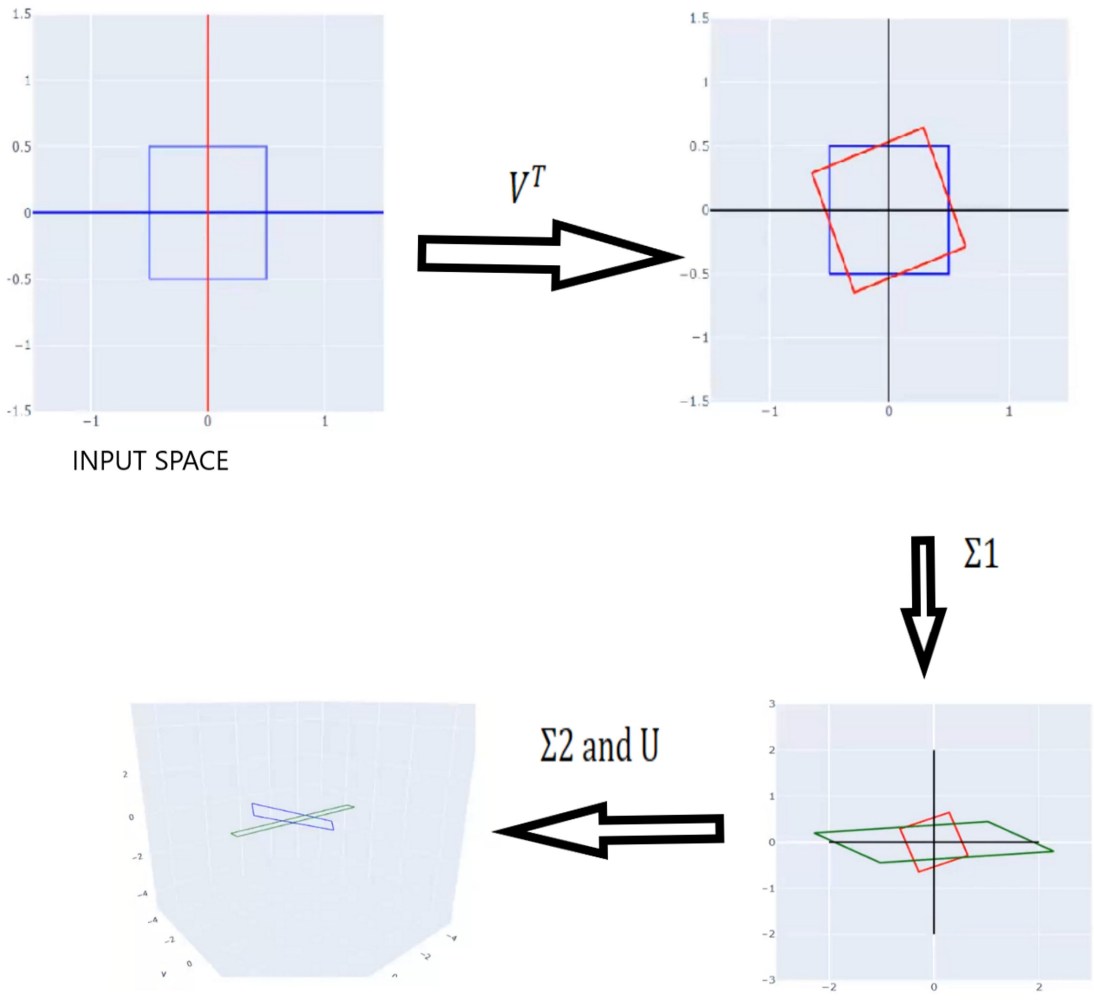
For any matrix A ∈ $R^{2\times3}$, the full SVD transformation works like this:

1. "V^T – Rotate the input (3D) space to align with principal directions."
2. "Σ_2 – Project the input from 3D down to a 2D plane."
3. "Σ_1 – Stretch the 2D plane along new axes (singular value scaling)."
4. "U – Rotate the stretched 2D plane to align with the output basis."

DEMO 1 [2 * 3]



DEMO 2 [3*2]



How to Calculate SVD

The Singular Value Decomposition (SVD) of a matrix A is given by:

$$A = U\Sigma V^T$$

Where:

- U and V are orthogonal matrices,
- Σ is a diagonal matrix with non-negative real numbers (the singular values of A).

Using Eigen Decomposition to Derive SVD

If we are given a matrix A , and need to compute U , Σ , and V^T, we can use the following method based on eigen decomposition:

1. Compute AA^T:
    - This matrix is symmetric and positive semi-definite.
    - Perform eigen decomposition on AA^T :

$$A A^T = U \Lambda U^T$$

- The columns of U are the eigenvectors of $A A^T$, and Λ contains the eigenvalues.

2. Compute $A^T A$:
   - Similarly, perform eigen decomposition :
   $$A^T A = V \Lambda V^T$$
   - The columns of V are the eigenvectors of $A^T A$.
3. Relating Eigenvalues to Singular Values:
   - The singular values σ_i are the square roots of the non-zero eigenvalues λ_i :
   $$\Sigma = \sqrt{\Lambda}$$

Numerical Stability Note

- This approach using eigen decomposition works conceptually, but it's numerically unstable and not used in practice.

SVD in PCA

In Principal Component Analysis (PCA), we aim to find the principal components of the data. There are two main approaches to achieve this:

1. Using Eigen Decomposition of the Covariance Matrix
2. Using Singular Value Decomposition (SVD) Directly

1. Eigen Decomposition Approach

- First, we center the data (subtract the mean from each feature).
- Then, we compute the covariance matrix of the data.
  - For a dataset with shape n × d , the covariance matrix has shape d × d.
- We perform eigen decomposition on the covariance matrix:
$$\mathrm{Cov}(X) = Q \Lambda Q^T$$
  - The eigenvectors represent the directions of principal components.
  - The eigenvalues represent the variance captured along each principal component.

2. SVD Approach (More Efficient)

- Instead of computing the covariance matrix, we can directly apply SVD to the centered data matrix X .
- Perform SVD:
$$X = U \Sigma V^T$$
  - The columns of V (or rows of V^T) give the principal directions.

- The singular values in Σ relate to the variance captured along each component.

Advantages of SVD in PCA:

- No need to compute the covariance matrix.
- More numerically stable and efficient, especially for high-dimensional data.
- Commonly used in practical implementations like scikit-learn's PCA.

Understanding the Relationship Between SVD and Covariance in PCA

Let's consider the Iris dataset. It contains 150 rows (samples) and 4 columns (features):

- Sepal length
- Sepal width
- Petal length
- Petal width

Hence, the data matrix X has dimensions 150 × 4.
If we calculate the covariance matrix of this data, it will be of shape 4 × 4 , since covariance is calculated between features.

Step-by-step: How Covariance is Computed

To compute the covariance matrix, we follow two key steps:

1. Mean Center the Data:
   Subtract the mean of each column (feature) from that column.
   Let the mean-centered matrix be X_c.
2. Apply the Covariance Formula:

$$\mathrm{Cov}(X) = \frac{X_c^T X_c}{n-1}$$

Where:

- n is the number of samples (150 in this case)
- X_c is the centered data matrix

How SVD Connects to Covariance

Suppose we directly apply SVD to the mean-centered data matrix X_c:

$$X_c = U\Sigma V^T$$

Now let's compute:

$$X_c^T X_c = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^T U^T U\Sigma V^T = V\Sigma^2 V^T$$

This is the eigen decomposition of $X_c^T X_c$, meaning:

- V contains the eigenvectors of $X_c^T X_c$
- Σ^2 contains the eigenvalues

So, $X_c^T X_c$ (which is proportional to the covariance matrix) has the same eigenvectors as those obtained from SVD.

- When we perform SVD on X_c, we automatically get the principal components in V without explicitly computing the covariance matrix.
- That's why SVD is often preferred: it gives us the same result as eigen decomposition of the covariance matrix, but is computationally more stable and efficient.
- This principle is used in libraries like scikit-learn for PCA under the hood.

Why SVD is Powerful in PCA

One of the major benefits of using SVD in PCA is that it allows us to obtain the eigenvalues and eigenvectors without explicitly computing the covariance matrix. If we directly take our mean-centered data matrix X_c and apply SVD:

$$X_c = U\Sigma V^T$$

- The columns of V are the principal directions (eigenvectors of the covariance matrix).
- The values in Σ (the singular values) are the square roots of the eigenvalues of $X_c^T X_c$.
    - That is, Σ^2 gives us the eigenvalues.

Key Insight:

We are getting the same eigenvectors and eigenvalues as we would from eigen decomposition of the covariance matrix, but without ever computing the covariance matrix directly.

This makes SVD:

- Faster
- More numerically stable
- Widely used in real-world PCA implementations, like in scikit-learn.