Singular Value Decomposition (SVD)

Singular Value Decomposition is a factorization (breaking down) method in linear algebra, from it's name it decomposes a given matrix into **three other matrices** which gives a way to represent data in terms of **singular values**, but first let's explain some linear algebra concepts :

As established **matrix multiplication** is combination of transformation and a **matrix** as a transformation, for **SVD** let's detail the matrices we need:

Orthogonal & Diagonal Matrices

Orthogonal Matrix $\rightarrow$ square matrix $+$ every column vector is a **unit vector** $+$ all columns vector are perpendicular

Orthogonal Matrix is a rotation transformation

- The **transpose** of the orthogonal matrix is it's **inverse** which is the rotation in the inverse direction
- orthogonal matrices column vectors are **unitary** which means all the vector columns are unit vectors *length of* 1 so the **dot product** between distinct columns is 0

  Diagonal Matrix $\rightarrow$ square matrix $+$ zero on the elements outside the diagonal
- **Diagonal Matrix** scales each axis

Symmetric Matrix

Symmetric matrices it's **guaranteed** that they have full set of eigen vectors

- The **eigen vectors** of a symmetric matrix are **perpendicular**
- Taking the **eigen vectors** of the symmetric matrix and **normalize** them results in a **orthogonal matrix**

Having matrix X :

- Multiplying $XX^\intercal$ or $X^\intercal X$ results in a **symmetric matrix**
- Which have the same properties as the normal symmetric matrix

Let $XX^\intercal$ be called S_L and $X^\intercal X$ be called S_R :

- The S_L has orthonormal eigenvectors $u_1 \ldots, u_m \rightarrow$ forms the columns of vector U
- The S_R has orthonormal eigenvectors $v_1, \ldots, v_n \rightarrow$ forms the columns of vector V
  Both of these S_L and S_R shared **eigenvalues** are the values for the eigenvalues for matrix X and the $\sqrt{\lambda_i} = \sigma_i$ which are the diagonal elements in the Σ matrix

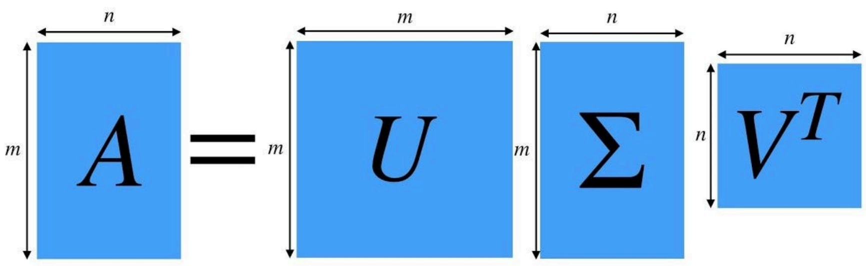
SVD

Given a **data matrix** A :

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

- U and $V^\intercal$ are both unitary matrices (**Orthogonal**), which means $U^\intercal U = UU^\intercal = \mathbb{I}$
- Σ is a **Diagonal matrix** and hierarchically ordered $\sigma_1 > \sigma_2 \cdots > \sigma_m \geq_0$ which is by importance

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



$$U = \begin{bmatrix} u_1 & u_2 & \dots & u_n \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

- The columns of U have the same **shape** as the columns in our **data matrix** A, also called the left singular matrix

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m \end{bmatrix}$$

- Σ called matrix of singular values

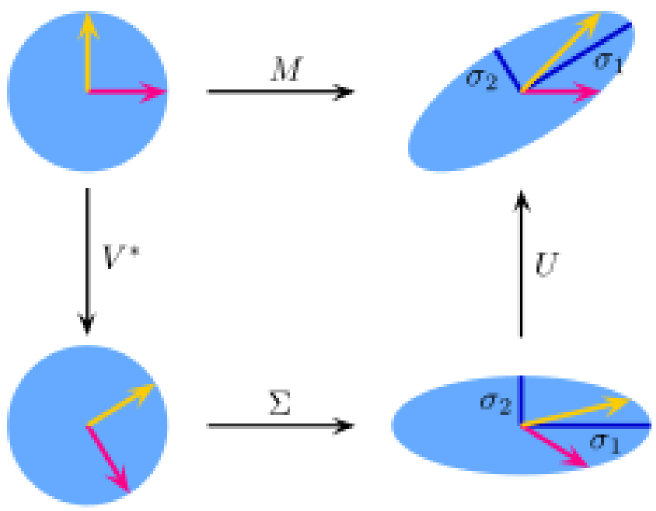
$$V^\intercal = \begin{bmatrix} \vdots & \vdots & & \vdots \\ v_1 & v_2 & \dots & v_m \\ \vdots & \vdots & & \vdots \end{bmatrix}^\intercal$$

- $V^\intercal$ also known as the right singular matrix

Interpretation of SVD

geometric viewpoint

If we tackled the **SVD** based on the intuition that the matrices are just **transformations** then the **SVD** simply decompose the matrix A into 3 simple **transformations**:



$$M = U \cdot \Sigma \cdot V^*$$

- $V^\intercal$ an orthogonal matrix which **rotate** the **input basis** into a new coordinate system
- Σ a diagonal matrix which Stretch each axis by σ
- U an orthogonal matrix which **rotate** the standard basis to the **output basis**

Conceptual viewpoint

The key to understand the **SVD** is viewing the columns of U, Σ and V as representing **concepts** that are hidden in the original matrix A, and in the process of decomposition there will be columns in U and V that correspond to the smallest singular values and in order to get the best approximation of the original matrix, we use the **Reduced SVD** which :

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

- Most of those columns in these decomposed matrices are useless since the one that effect the most comes first (columns are ordered in terms of importance)
- And they will carry zero values on them

Economy/Reduced SVD eliminate them by taking the first r columns which results in

$$A = U_r\Sigma_r V_r^\intercal$$

- $r = rank(A)$
- The U_r, Σ_r only took the first m relevant columns getting rid of the **zeros**

Truncated rank SVD this is dimensionality reduction, where it drops the **smallest singular values**, with $k < r$

$$A \approx U_k\Sigma_k V_k^\intercal$$

- This saves memory and speeds up the computations for the **SVD**
- It's the backbone of dimensionality reduction methods since it get rid of least relevant columns and only keeps the important columns which reduce the dimensions of the approximated full matrix A

SVD Eigenvectors & Eigenvalues interpretation

The SVD has a nice interpretation with the Eigenvectors & Eigenvalues and the correlation matrix $A^\intercal A$ and $AA^\intercal$:

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

Gives us the **correlation matrix** :

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

With $U_r^T U_r = \mathbb{I}$ the identity matrix

$$A^\intercal A = V\Sigma^2 V^\intercal$$

Multiplying both sides with V :

$$A^\intercal AV = V\Sigma^2$$

This reads as follows :

- The transformation matrix $A^\intercal A$ got V as an **Eigenvectors** matrix
- With Σ_r^2 being the **Eigenvalues** that scales the V_r columns on their span

and calculating $AA^\intercal$ results in :

$$AA^\intercal U = U\Sigma^2$$

- Both of U and V are **Eigenvector** for the correlation matrix with the same **Eigenvalues** Σ^2
- The Σ matrix is the **Square root** of the **Eigenvalues** of both the left and right matrix U, V

How to Calculate SVD ?

For intuition and simple examples the **SVD** can be calculated by solving for the **eigenvectors** of $A^\intercal A$ or $AA^\intercal$, but for large matrices and **numerical stability** we use these three :

1. **Golub-Reinsch Algorithm** which is the standard **SVD** solving algorithm used in library **NumPY** and **MATLAB**, for reference check : https://people.inf.ethz.ch/gander/talks/Vortrag2022.pdf
2. **Jacobi SVD Algorithm** it's slower but extremely stable and accurate, used in high-precision scientific research and computation not for production pipelines
3. **Randomized SVD** it's a modern fast approximation algorithm used for **large scale** and **Deep Learning** scale, instead of compute the **SVD** it samples a column space (think of it as a cross validation test error estimate), used in **Scikit-learn** `randomized_svd()` and Principal Component Analysis (PCA) implementation