University: Isfahan University - Faculty of Computer Engineering

Course: Linear Algebra

Movie Recommender System using SVD

Student Name: **Sheida Abedpour**
Student ID: 4003623025

2023 Jan

SVD:

SVD stands for Singular Value Decomposition. It is a mathematical technique used in linear algebra and numerical analysis. SVD decomposes a matrix into three other matrices, which can be useful in various applications, including data analysis, signal processing, and machine learning.

Given a matrix A of size m x n, the SVD of A is represented as:

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

where:

- U is an m x m orthogonal matrix, representing left singular vectors.
- Σ is an m x n diagonal matrix with non-negative real numbers on the diagonal, representing singular values.
- V^T is an n x n orthogonal matrix, representing right singular vectors.

The singular values on the diagonal of Σ are arranged in descending order.

SVD has applications in various fields, including dimensionality reduction, noise reduction, and in the context of collaborative filtering for recommendation systems.

In the context of collaborative filtering for recommendation systems, the user-item interaction matrix can be decomposed using SVD, and the resulting matrices U, Σ, and V^T can be used to make recommendations for missing values in the original matrix.

Power iteration:

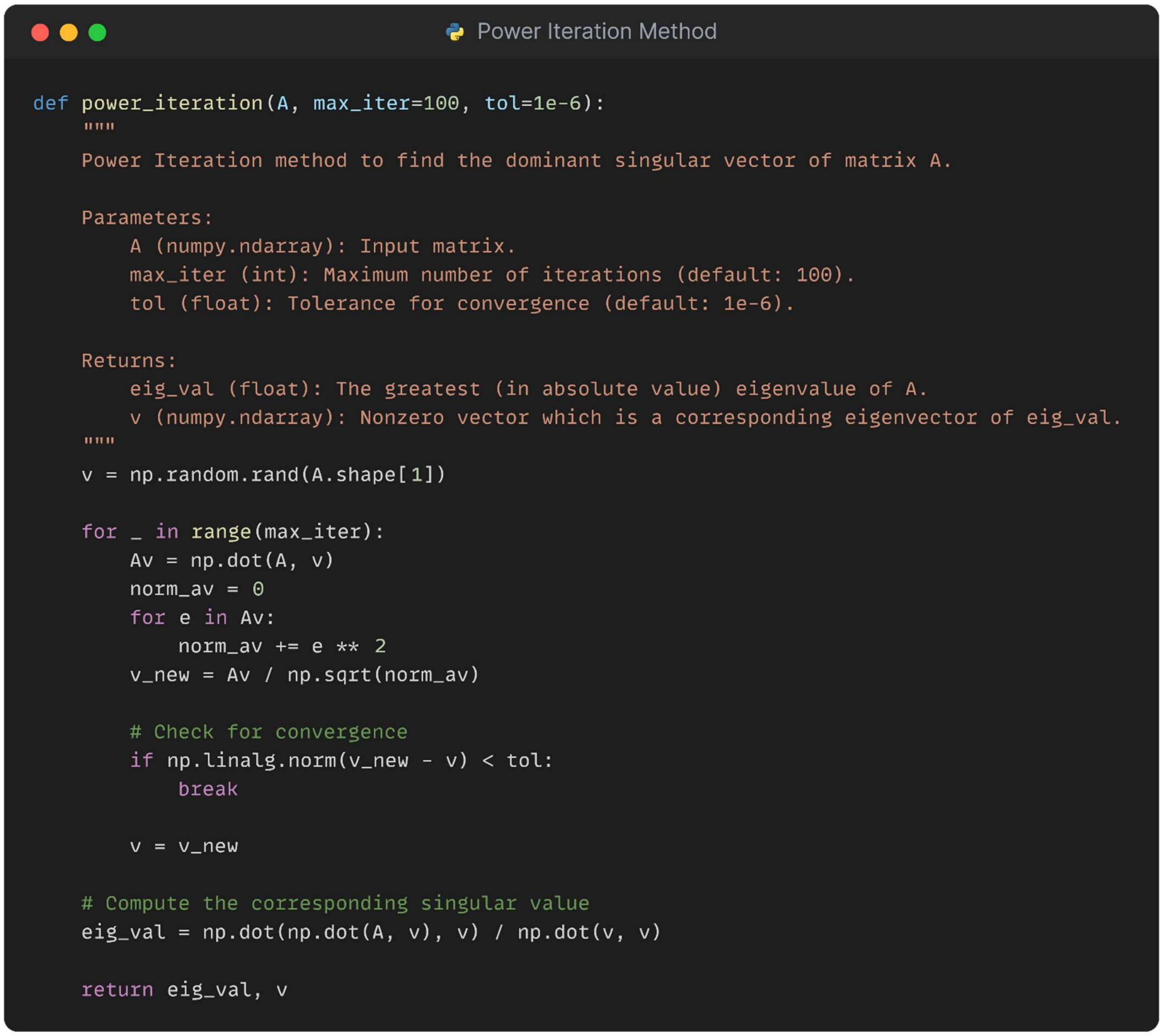
Power iteration is an iterative method used to find the dominant eigenvalue (or singular value) and its corresponding eigenvector (or singular vector) of a square (or rectangular) matrix. It is a simple and effective algorithm that converges to the dominant eigenvalue and eigenvector.

Power iteration starts with $\mathbf{b_0}$ which might be a random vector. At every iteration this vector is updated using following rule:

$$b_{k+1} = \frac{Ab_k}{\|Ab_k\|}$$

First we multiply $\mathbf{b_0}$ with original matrix A ($\mathbf{Ab_k}$) and divide result with the norm ($||\mathbf{Ab_k}||$). We'll continue until result has converged (updates are less than threshold).

Here's a implementation of the power iteration method:



```python
def power_iteration(A, max_iter=100, tol=1e-6):
    """
    Power Iteration method to find the dominant singular vector of matrix A.

    Parameters:
        A (numpy.ndarray): Input matrix.
        max_iter (int): Maximum number of iterations (default: 100).
        tol (float): Tolerance for convergence (default: 1e-6).

    Returns:
        eig_val (float): The greatest (in absolute value) eigenvalue of A.
        v (numpy.ndarray): Nonzero vector which is a corresponding eigenvector of eig_val.
    """
    v = np.random.rand(A.shape[1])

    for _ in range(max_iter):
        Av = np.dot(A, v)
        norm_av = 0
        for e in Av:
            norm_av += e ** 2
        v_new = Av / np.sqrt(norm_av)

        # Check for convergence
        if np.linalg.norm(v_new - v) < tol:
            break

        v = v_new

    # Compute the corresponding singular value
    eig_val = np.dot(np.dot(A, v), v) / np.dot(v, v)

    return eig_val, v
```

In the context of Singular Value Decomposition (SVD), deflation is a technique used to iteratively find subsequent singular vectors and values by subtracting their contributions from the original matrix. The term "deflation" is used because the process involves removing the effect of previously computed singular vectors from the matrix.

Here's a explanation of deflation in the context of SVD:

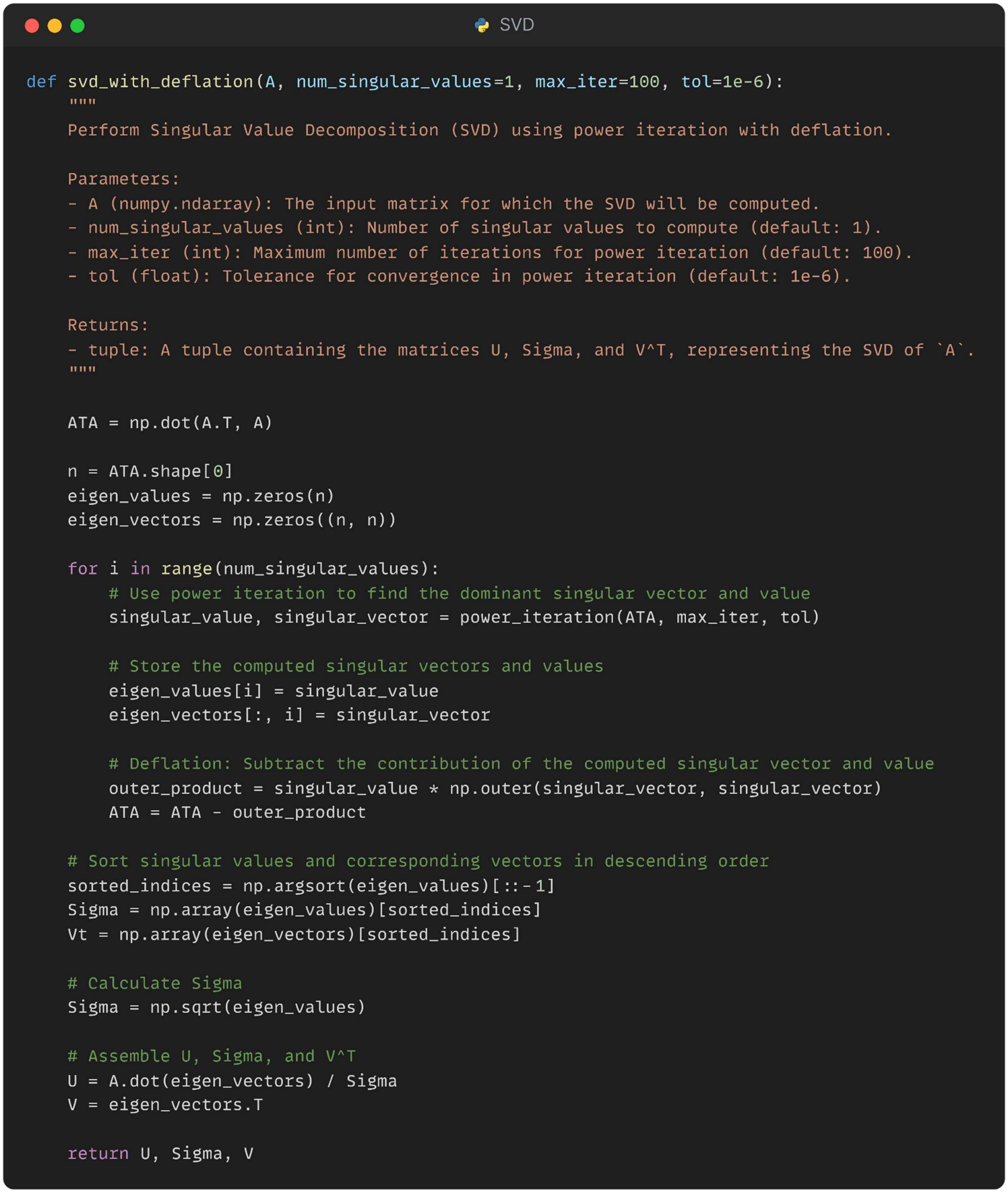
1. **Initial Step:**
    - Compute the matrix $A^T A$ to find the right singular vector using power iteration.
    - Obtain the dominant singular vector *v* and the corresponding singular value *σ* using power iteration.
2. **Deflation:**
    - Subtract the contribution of the computed singular vector and value from the original matrix *A*.
    - Update the matrix *A* to remove the effect of the already computed singular vector and value.
    - Repeat the power iteration on the updated matrix to find the next singular vector and value.
3. **Repeat:**
    - Repeat the process until the desired number of singular vectors and values are obtained.

In mathematical terms, if *u* is the left singular vector, *v* is the right singular vector, and *σ* is the singular value obtained in one iteration, deflation involves subtracting the outer product of *u* and *v* scaled by *σ* from the original matrix *A*:

$$A \leftarrow A - \sigma . uv^T$$

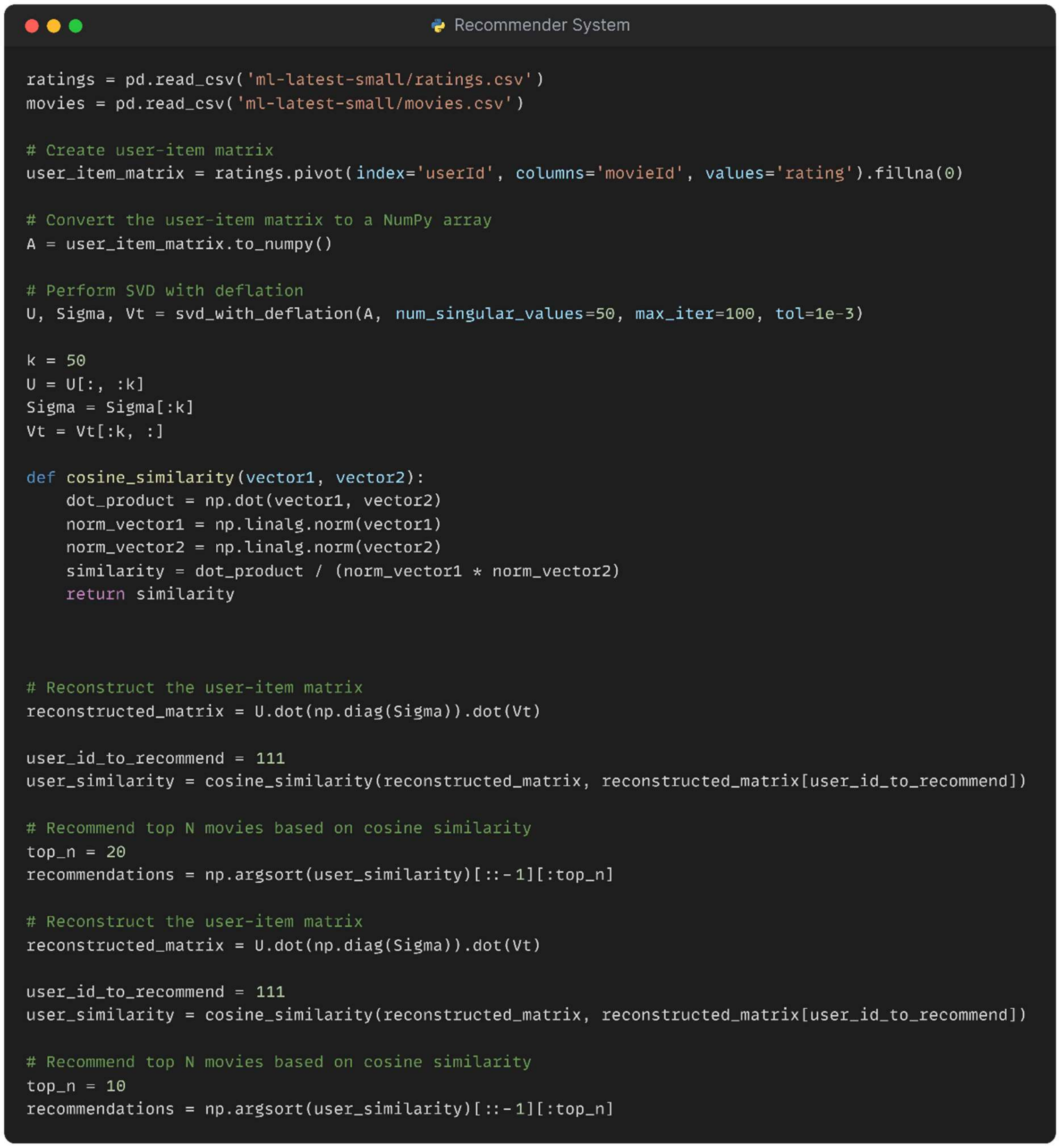
This process is necessary because the singular vectors and values are not unique, and each iteration helps in finding a new set of singular vectors and values orthogonal to the ones already obtained.

The goal of deflation is to iteratively find a series of singular vectors and values that approximate the original matrix *A*. The resulting matrices *U*, Σ, and V^T in the SVD are constructed based on these singular vectors and values.


SVD

```python
def svd_with_deflation(A, num_singular_values=1, max_iter=100, tol=1e-6):
    """
    Perform Singular Value Decomposition (SVD) using power iteration with deflation.

    Parameters:
    - A (numpy.ndarray): The input matrix for which the SVD will be computed.
    - num_singular_values (int): Number of singular values to compute (default: 1).
    - max_iter (int): Maximum number of iterations for power iteration (default: 100).
    - tol (float): Tolerance for convergence in power iteration (default: 1e-6).

    Returns:
    - tuple: A tuple containing the matrices U, Sigma, and V^T, representing the SVD of `A`.
    """

    ATA = np.dot(A.T, A)

    n = ATA.shape[0]
    eigen_values = np.zeros(n)
    eigen_vectors = np.zeros((n, n))

    for i in range(num_singular_values):
        # Use power iteration to find the dominant singular vector and value
        singular_value, singular_vector = power_iteration(ATA, max_iter, tol)

        # Store the computed singular vectors and values
        eigen_values[i] = singular_value
        eigen_vectors[:, i] = singular_vector

        # Deflation: Subtract the contribution of the computed singular vector and value
        outer_product = singular_value * np.outer(singular_vector, singular_vector)
        ATA = ATA - outer_product

    # Sort singular values and corresponding vectors in descending order
    sorted_indices = np.argsort(eigen_values)[::-1]
    Sigma = np.array(eigen_values)[sorted_indices]
    Vt = np.array(eigen_vectors)[sorted_indices]

    # Calculate Sigma
    Sigma = np.sqrt(eigen_values)

    # Assemble U, Sigma, and V^T
    U = A.dot(eigen_vectors) / Sigma
    V = eigen_vectors.T

    return U, Sigma, V
```

codesnap.dev



```python
ratings = pd.read_csv('ml-latest-small/ratings.csv')
movies = pd.read_csv('ml-latest-small/movies.csv')

# Create user-item matrix
user_item_matrix = ratings.pivot(index='userId', columns='movieId', values='rating').fillna(0)

# Convert the user-item matrix to a NumPy array
A = user_item_matrix.to_numpy()

# Perform SVD with deflation
U, Sigma, Vt = svd_with_deflation(A, num_singular_values=50, max_iter=100, tol=1e-3)

k = 50
U = U[:, :k]
Sigma = Sigma[:k]
Vt = Vt[:k, :]

def cosine_similarity(vector1, vector2):
    dot_product = np.dot(vector1, vector2)
    norm_vector1 = np.linalg.norm(vector1)
    norm_vector2 = np.linalg.norm(vector2)
    similarity = dot_product / (norm_vector1 * norm_vector2)
    return similarity



# Reconstruct the user-item matrix
reconstructed_matrix = U.dot(np.diag(Sigma)).dot(Vt)

user_id_to_recommend = 111
user_similarity = cosine_similarity(reconstructed_matrix, reconstructed_matrix[user_id_to_recommend])

# Recommend top N movies based on cosine similarity
top_n = 20
recommendations = np.argsort(user_similarity)[::-1][:top_n]

# Reconstruct the user-item matrix
reconstructed_matrix = U.dot(np.diag(Sigma)).dot(Vt)

user_id_to_recommend = 111
user_similarity = cosine_similarity(reconstructed_matrix, reconstructed_matrix[user_id_to_recommend])

# Recommend top N movies based on cosine similarity
top_n = 10
recommendations = np.argsort(user_similarity)[::-1][:top_n]
```

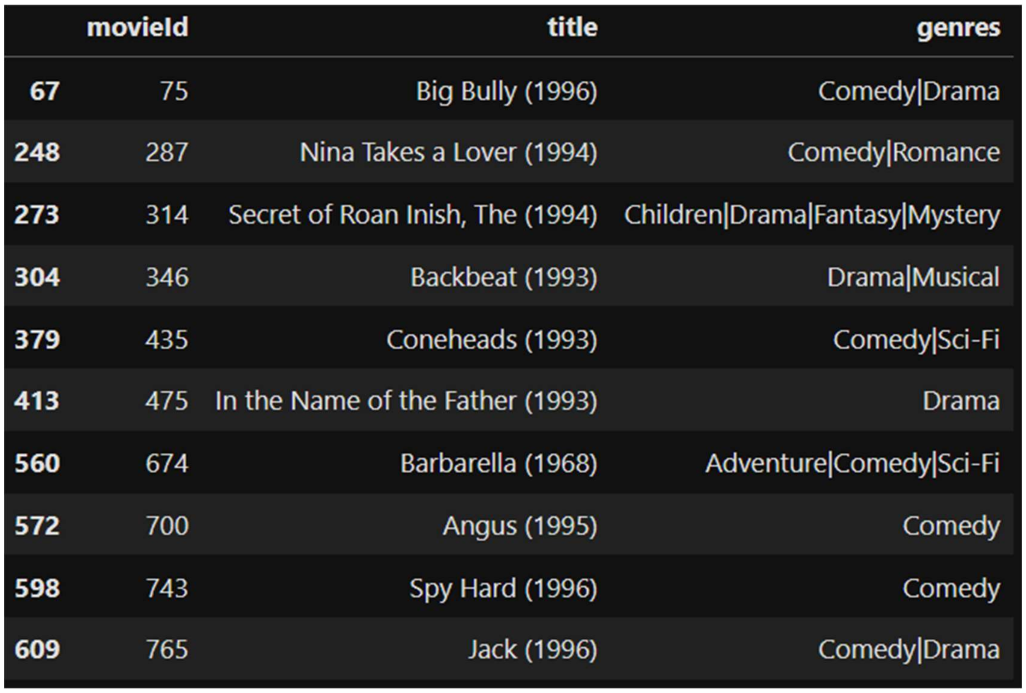
codesnap.dev

In the context of Singular Value Decomposition (SVD), the parameter *k* represents the number of singular values (and their corresponding singular vectors) to retain during the truncation step. It determines the dimensionality of the approximation to the original matrix.

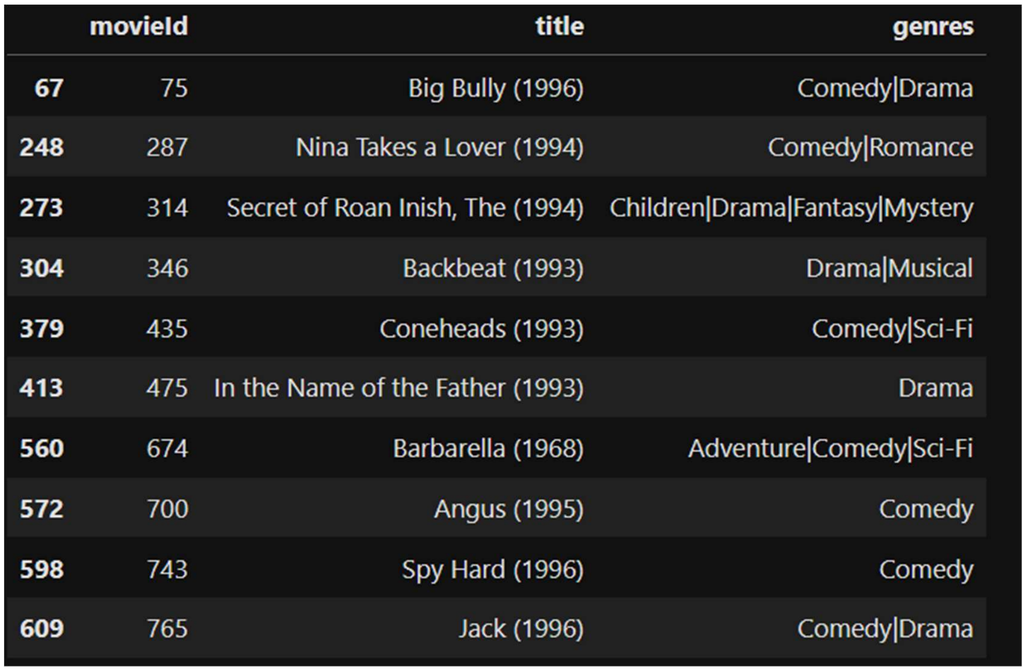
When you perform SVD, the singular values are sorted in descending order, and the corresponding singular vectors are arranged accordingly. Truncating U, Σ, and V^T to include only the first *k* components means that you are retaining the information captured by the top *k* singular values and their associated vectors.

The choice of *k* is a trade-off between dimensionality reduction and preserving information. Larger values of *k* result in a closer approximation to the original matrix but may retain more dimensions. Smaller values of *k* reduce the dimensionality more aggressively but may lose some information.

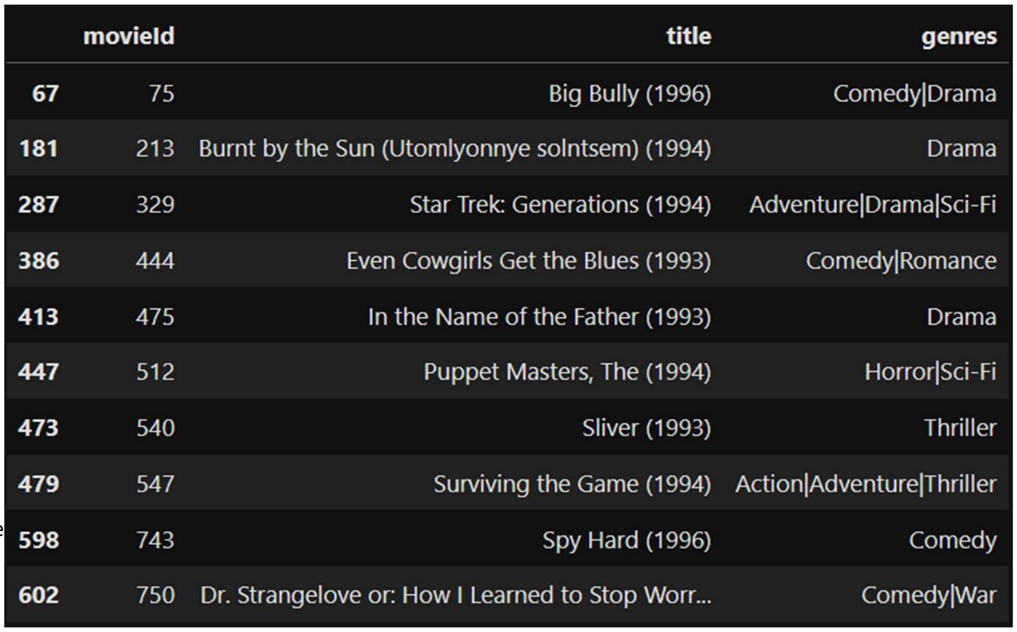
In practical terms, selecting an appropriate *k* depends on the specific application, the desired level of compression, and the importance of retaining information. Experimentation and evaluation based on the specific context can help determine an optimal value for *k*.



	movieId	title	genres
67	75	Big Bully (1996)	Comedy|Drama
248	287	Nina Takes a Lover (1994)	Comedy|Romance
273	314	Secret of Roan Inish, The (1994)	Children|Drama|Fantasy|Mystery
304	346	Backbeat (1993)	Drama|Musical
379	435	Coneheads (1993)	Comedy|Sci-Fi
413	475	In the Name of the Father (1993)	Drama
560	674	Barbarella (1968)	Adventure|Comedy|Sci-Fi
572	700	Angus (1995)	Comedy
598	743	Spy Hard (1996)	Comedy
609	765	Jack (1996)	Comedy|Drama



	movieId	title	genres
67	75	Big Bully (1996)	Comedy|Drama
248	287	Nina Takes a Lover (1994)	Comedy|Romance
273	314	Secret of Roan Inish, The (1994)	Children|Drama|Fantasy|Mystery
304	346	Backbeat (1993)	Drama|Musical
379	435	Coneheads (1993)	Comedy|Sci-Fi
413	475	In the Name of the Father (1993)	Drama
560	674	Barbarella (1968)	Adventure|Comedy|Sci-Fi
572	700	Angus (1995)	Comedy
598	743	Spy Hard (1996)	Comedy
609	765	Jack (1996)	Comedy|Drama



	movieId	title	genres
67	75	Big Bully (1996)	Comedy|Drama
181	213	Burnt by the Sun (Utomlyonnye solntsem) (1994)	Drama
287	329	Star Trek: Generations (1994)	Adventure|Drama|Sci-Fi
386	444	Even Cowgirls Get the Blues (1993)	Comedy|Romance
413	475	In the Name of the Father (1993)	Drama
447	512	Puppet Masters, The (1994)	Horror|Sci-Fi
473	540	Sliver (1993)	Thriller
479	547	Surviving the Game (1994)	Action|Adventure|Thriller
598	743	Spy Hard (1996)	Comedy
602	750	Dr. Strangelove or: How I Learned to Stop Worr...	Comedy|War

Refrences used for this project:

- towardsdatascience.com/simple-svd-algorithms
- ins.sjtu.edu.cn/people/leili/teaching/sjtu
- www.youtube.com
- github.com/RRisto/learning/blob/master/linear_algebra_learn/PCA_SVD/power_method.ipynb

GitHub Repository of this Project:

github.com/SheidaAbedpour/SVD-movie-recommender-system/tree/main