Movie Recommendation System Using Singular Value Decomposition (SVD)

1 Background

The rise of user-centric applications such as movie recommendation systems has significantly transformed how users discover new content. Recommender systems aim to suggest items (movies, books, music) that a user might find appealing. The challenge lies in selecting the most suitable movies for users with minimal historical data and predicting what a user would like based on other users' preferences.

One popular technique for addressing this challenge is Singular Value Decomposition (SVD). SVD reduces the dimensionality of large datasets and reveals underlying relationships between users and movies. In this report, we describe how SVD can be applied to a movie recommendation system to predict ratings and recommend movies for both existing and new users.

2 Problem Statement

We aim to build a movie recommendation system using a given dataset (e.g., the MovieLens dataset). The system will:

- Predict movie ratings for users based on the historical user-movie ratings matrix.
- Provide recommendations to new users by leveraging both past ratings and predicted ratings from the SVD algorithm.
- Visualize the difference between actual and predicted ratings for users.

The MovieLens dataset, comprising movies, ratings, and tags, is used as input for this system. The solution involves performing SVD on the user-movie ratings matrix, generating recommendations, and visualizing results.

3 Introduction to SVD

Singular Value Decomposition (SVD) is a mathematical technique used to factorize a matrix into three distinct matrices, providing insights into the structure

of the data. For a given matrix A of dimensions $m \times n$, SVD can be represented as:

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

Where:

- U is an $m \times m$ orthogonal matrix whose columns are the left singular vectors.
- Σ is an $m \times n$ diagonal matrix containing the singular values, which are non-negative and ordered from largest to smallest.
- V^T is an $n \times n$ orthogonal matrix whose rows are the right singular vectors.

3.1 Matrix Components

Matrix U:

- Each column of U represents a left singular vector, corresponding to the original users in the dataset.
- It reflects how much each user relates to the latent features.
- Dimensions: $m \times k$ (where k is the number of latent features).

Matrix Σ:

- Diagonal elements of Σ are the singular values, denoted as $\sigma_1, \sigma_2, \ldots, \sigma_k$. These values quantify the importance of each latent feature.
- The singular values are always non-negative, and they are arranged in descending order: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k$.
- Dimensions: $k \times k$.

Matrix V^T:

- Each row of V^T corresponds to a right singular vector, representing the movies in the dataset.
- It indicates how each movie relates to the latent features.
- Dimensions: $k \times n$.

3.2 Latent Features

Latent features are derived from the SVD by identifying patterns in user and item interactions that are not explicitly visible in the original ratings. In the context of movie ratings:

- Each latent feature captures an underlying characteristic of users' preferences or movies' attributes. For example, one latent feature may correspond to a preference for action movies, while another may capture a preference for romance.

The computation of latent features is achieved through the matrix multiplication of U, Σ, and V^T. The predicted ratings for a user can be expressed as:

$$\hat{A} = U_k \Sigma_k V_k^T$$

Where:

- U_k contains the first k columns of U.
- Σ_k is the top $k \times k$ diagonal matrix with the top k singular values.
- V_k^T contains the first k rows of V^T.

3.3 Choise of (k)

The choice of k is often empirical, based on balancing:

- **Model Complexity:** A higher k can lead to a more complex model that captures more variance and detail in the data but can also result in overfitting.
- **Computational Efficiency:** SVD with a lower k reduces computation time and memory usage while still capturing the essential patterns in the data.
- **Data Characteristics:** The selection of k should consider the size of the dataset and the inherent structure. Cross-validation can help determine the optimal k.
- **T**he Optimal value of k is calculated for the current code, which is provided in the Github link.

3.4 Conclusion

In summary, SVD provides a powerful mechanism for reducing dimensionality and uncovering latent structures in large datasets. By factorizing the user-movie ratings matrix into U, Σ, and V^T, we can make predictions about unrated movies for users, enabling personalized recommendations. The choice of k should be carefully considered based on the specific application, ensuring a balance between model complexity and predictive performance.

4 Movie Recommendation based on Past and Predicted Ratings - Code Explanation

This section explains the core implementation of the movie recommendation system using Singular Value Decomposition (SVD) for matrix factorization. The code is accessible through the provided GitHub link.

1. Data Loading and Preparation

The program begins by loading three datasets: `Movie_Data.csv`, `Ratings_Data.csv`, and `Tags_Data.csv`. These contain metadata on movies, user ratings, and related tags. The ratings data is used to create a user-movie ratings matrix, where rows correspond to users, columns correspond to movies, and each cell represents a user's rating of a particular movie. Missing ratings are replaced with zeros using `fillna(0)` to create a dense matrix that includes all user-movie combinations:

```
user_movie_ratings = ratings.pivot(index='userId', columns='movieId',
values='rating').fillna(0)
```

To efficiently perform operations on large matrices, the code converts the ratings matrix into a sparse matrix representation using `csr_matrix`, which optimizes memory usage by storing only non-zero entries.

2. Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a mathematical technique used to factorize the user-movie ratings matrix R into three smaller matrices:

$$R \approx U \cdot \Sigma \cdot V^T$$

Where:

- U is an $m \times k$ matrix, where m is the number of users and k is the number of latent features.
- Σ is a diagonal matrix containing the singular values. These values indicate the importance of each latent feature.
- V^T is the transpose of an $n \times k$ matrix, where n is the number of movies.

In the code, the `svds` function from `scipy.sparse.linalg` is used to compute SVD. The number of latent factors (k) is set to 50:

```
U, sigma, Vt = svds(user_movie_ratings_sparse, k=40)
sigma = np.diag(sigma)
```

The matrix Σ contains the singular values that are arranged in descending order of significance. These latent features represent hidden relationships between users and movies, such as preferences for genres, actors, or directors.

3. Reconstructing the Ratings Matrix

The code uses the matrices U, Σ, and V^T to approximate the original ratings matrix R, allowing for predictions of unrated movies:

$$R_{\text{reconstructed}} = U \cdot \Sigma \cdot V^T$$

This reconstructed matrix provides estimated ratings for all user-movie pairs, filling in missing values (i.e., where users did not rate certain movies). These predicted ratings are stored in a new DataFrame for easy access and comparison with the original ratings.

4. Movie Recommendations

The function `recommend_movies_with_history` generates personalized movie recommendations for a given user. It combines the user's past ratings with predicted ratings obtained from the SVD model:

- **Original Ratings**: Movies that the user has already rated are retrieved and ranked.
- **Predicted Ratings**: Movies that the user has not rated are sorted based on the predicted ratings from the reconstructed matrix.
- The system gives preference to highly rated movies from the user's history and augments this list with top predicted movies.

The function outputs a list of recommended movies along with their corresponding predicted ratings.

5. Visualization of Original vs Predicted Ratings

To provide insight into the accuracy of the predictions, the `plot_user_ratings` function generates a plot comparing the original and predicted ratings for a specific user. This allows us to visually assess how closely the model's predictions align with the user's actual ratings.

```
plt.plot(original_ratings.index, original_ratings, label='Original Ratings',
         marker='o', color='blue', alpha=0.6)
plt.plot(predicted_ratings.index, predicted_ratings, label='Predicted Ratings',
         marker='o', color='red', alpha=0.6)
```

6. Analysis of Singular Values

The singular values in the matrix Σ reflect the significance of each latent feature. Larger singular values correspond to more important latent features that capture major patterns in user preferences. The function plots these singular values to provide insight into how the latent features contribute to the model.

7. Star Ratings and Display of Recommendations

To enhance the user experience, the code includes functions to display movie recommendations in a user-friendly format. The `colored_stars` function converts numeric ratings into a star-based format (with full stars and empty stars), while `bold_text` makes the movie titles bold.

The top recommended movies are displayed in the format:

```
{bold_title}: {stars} (Rating: {rating:.1f})
```

8. Conclusion

This recommendation system effectively leverages SVD to predict user ratings based on hidden patterns in the user-movie interactions. By decomposing the ratings matrix into latent features, the model can provide personalized recommendations even for movies that a user has not rated. SVD plays a vital role in capturing these underlying patterns and ensuring that the recommendations are aligned with user preferences.

9. Mechanism of Recommendation

To recommend movies to a new user:

1. Simulate new user ratings where all movie ratings are initially zero.
2. Identify the unseen movies (i.e., movies rated as 0 by the new user).
3. Use the predicted ratings from the reconstructed matrix to recommend the top N unseen movies to the user.

Explanation of the Movie Recommendation System with Tags(User with Preferences)

The code presented here implements a movie recommendation system using Singular Value Decomposition (SVD) on user-movie rating data. Additionally, the system allows for recommendations based on user preferences such as genres or specific movie tags (e.g., "fantasy", "action"). Below is a detailed explanation of the key sections of the code:

1. Loading the Data

2. Preparing the User-Movie Ratings Matrix

3. Applying Singular Value Decomposition (SVD)

To perform matrix factorization, the user-movie ratings matrix is decomposed using SVD into three matrices U, Σ, and V^T, where:

- U contains the user feature vectors.
- Σ is a diagonal matrix with singular values.
- V^T contains the movie feature vectors.

The optimal number of latent features, k, is chosen based on previous analysis or using methods such as the Elbow method:

```
U, sigma, Vt = svds(user_movie_ratings_sparse, k=100)
```

This decomposed matrix is then used to reconstruct the original ratings matrix.

4. Recommending Movies with User-Specified Preferences (Tags)

The code allows for movie recommendations that take into account the user's preferences for specific tags (e.g., genres or keywords like "fantasy", "action"). The key steps involved are:

- First, the system retrieves the unrated movies for the user by filtering the movies that have not been rated by the user.
- It then predicts ratings for these unrated movies using the SVD results.
- If the user has specified any preferences (such as tags), the system filters the movies based on these preferences using the `tags` dataset:

  ```
  matching_movies = tags[tags['tag'].isin(preferred_tags)]
            ['movieId'].unique()
  ```

  This selects only the movies that are associated with the specified tags.
- Finally, the system returns a list of recommended movies sorted by the predicted rating.

5. Displaying Recommendations

The recommended movies are displayed, where each movie title is shown along with a star rating (out of 5), which represents the predicted rating based on the user's preferences.

Emphasis on Tags Usage

The system leverages the `tags` dataset to incorporate the user's preferences for certain genres or themes into the recommendation process. By specifying a list of preferred tags (e.g., "fantasy", "action"), the recommendation engine filters the predicted ratings to include only those movies that match the user's desired tags. This enhances the personalization of recommendations, ensuring that users receive suggestions that align with their specific interests.

Conclusion

In summary, this code implements a robust movie recommendation system using matrix factorization (SVD) and allows for personalized recommendations based on user-specified tags. This feature makes it possible to provide more relevant recommendations by taking into account the user's preferences for specific genres or themes.

5 Visualization and Results

To better understand the effectiveness of our recommendation system, we compare the original and predicted ratings for a random user.

5.1 Original vs Predicted Ratings for a Specific User

Figure 1 shows the actual and predicted ratings for User. This helps to visualize the difference between actual and estimated ratings, thereby demonstrating the accuracy of the model.

5.2 SVD Matrices

The matrices resulting from the Singular Value Decomposition of the user-movie ratings matrix are presented in Figure 2. These matrices are crucial for understanding the latent structures in the data.

5.3 Singular Values

The Figure 3 plot displays the singular values obtained from the SVD, providing insight into the importance of each latent feature in the dataset.

5.4 Recommended Movies with Star Ratings

The Figure 4 shows the top recommended movies for the user, along with their predicted star ratings. This visualization aids in understanding how the recommendations are derived.

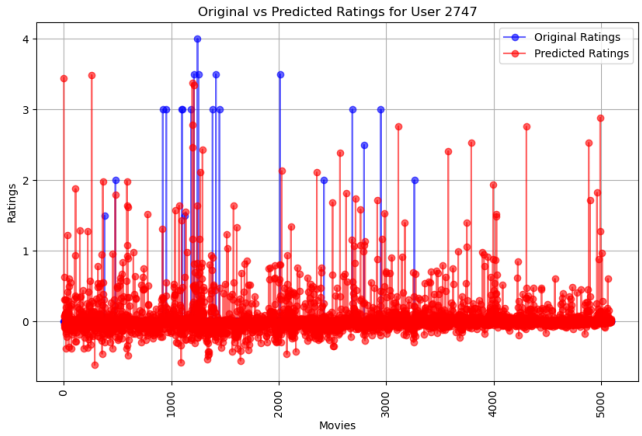

Figure 1: Original vs Predicted Ratings for User

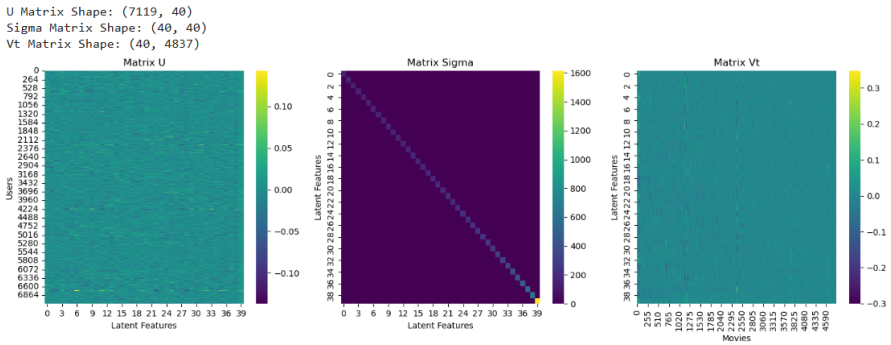

Figure 2: SVD Matrices: U, Σ, and V^T

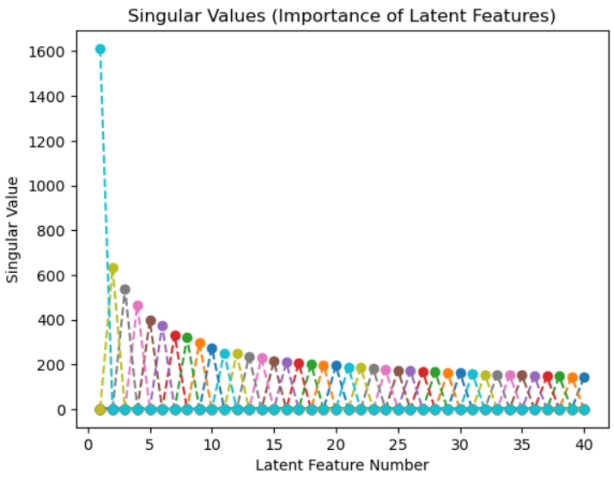

Figure 3: Singular Values (Importance of Latent Features)

```
Beautiful Girls (1996): ★★★★★ (Rating: 5.0)
Dances with Wolves (1990): ★★★★★ (Rating: 5.0)
Braveheart (1995): ★★★★★ (Rating: 5.0)
East of Eden (1955): ★★★★★ (Rating: 5.0)
Some Like It Hot (1959): ★★★★★ (Rating: 5.0)
Breakfast at Tiffany's (1961): ★★★★★ (Rating: 5.0)
Silence of the Lambs, The (1991): ★★★★★ (Rating: 5.0)
Fargo (1996): ★★★★★ (Rating: 5.0)
Schindler's List (1993): ★★★★★ (Rating: 5.0)
Last Action Hero (1993): ★★★★★ (Rating: 5.0)
```

Figure 4: Top Recommended Movies for User with Star Ratings

5.5 Recommended Movies with Ratings and Tags

The Figure 5 shows the top recommended movies for the user, along with their predicted star ratings along with Preference Tags. This visualization aids in understanding how the recommendations are derived.

6 Conclusion and Scope

In this report, we demonstrated the use of SVD in a movie recommendation system. By leveraging matrix factorization techniques, we successfully predicted movie ratings for both existing and new users. The application of SVD allows us to capture latent factors, thus providing meaningful recommendations even with sparse data.

Future scope includes:

- Incorporating more sophisticated models, such as hybrid collaborative filtering, which combines SVD with content-based filtering.
- Adding personalized recommendations based on user profiles and behavioral data.
- Experimenting with different values of latent factors k to optimize recommendations.

7 Code Availability

The complete code for the movie recommendation system is available on GitHub. You can access the repository using the following link:

GitHub Link

Figure 5: Top Recommended Movies for User with Star Ratings