AMRITA VISHWA VIDYAPEETHAM

Deemed to be University under Section 3 of UGC Act, 1956

Project 1: Recommender System

Submitted By:

Ajithkumar A K
[CB.AI.R4CEN24009]

Supervisor:

Dr. Soman K P

Submitted to
Amrita Vishwa Vidyapeetham, University
Amrita School of Artificial Intelligence

Submission Date: **November 21, 2024**

November 22, 2024

Contents

Chapter 1

Recommendation Systems: Implementation and Analysis

1.1 Summary

This project focuses on implementing and analyzing recommendation systems, a vital component in personalization and user experience across various domains like e-commerce, streaming platforms, and content curation. The primary purpose is to predict user preferences based on available data and suggest items they are likely to engage with, using three key approaches:

1.2 Collaborative Filtering

- This method calculates user-user similarity to predict missing ratings for items based on the preferences of similar users.
- A global baseline approach is also implemented, which accounts for overall average ratings and user-item biases to improve prediction accuracy.

1.3 Singular Value Decomposition (SVD)

- SVD is used to reduce the dimensionality of the user-item matrix, capturing latent factors that influence user preferences while retaining 90% of the original data's energy.
- This approach is computationally efficient and effective for sparse datasets.

1.4 CUR Decomposition

- CUR involves sampling rows (users) and columns (items) to approximate the user-item matrix.
- Two sampling techniques are explored:
  - Sampling with replacement.
  - Sampling without replacement.

1.5 Dataset Description

The project uses the Jester Online Joke Recommender Dataset, a robust dataset with 4.1 million continuous ratings (-10.00 to +10.00) from 73,421 users for 100 jokes. Data preprocessing, including handling missing ratings and normalization, ensures the dataset is ready for analysis.

1.6 Outcomes

- **Collaborative Filtering:** Showed robust performance in predicting missing ratings, especially with dense regions of the dataset.
- **SVD:** Achieved efficient dimensionality reduction, balancing computational cost with prediction accuracy.
- **CUR:** Provided flexible matrix approximations, offering insights into row and column sampling strategies.

1.7 Evaluation Metrics and Insights

The evaluation metrics, including Normalized Mean Absolute Error (NMAE), precision, recall, and F1 score, demonstrate the strengths and trade-offs of each approach. The project highlights the adaptability of these methods in addressing challenges like sparse data and computational constraints, offering a solid foundation for enhancing recommender systems.

Abstract

This project aims to develop and evaluate a recommender system using collaborative filtering, Singular Value Decomposition (SVD), and CUR decomposition techniques. The objective is to predict missing user ratings and suggest items based on user preferences. The dataset used is the Jester Online Joke Recommender Dataset, containing 4.1 million ratings from 73,421 users for 100 jokes, with ratings ranging from -10.00 to +10.00.

The project implements:

- Collaborative filtering based on user similarity and a global baseline approach.
- SVD for dimensionality reduction.
- CUR for matrix approximation through row and column sampling.

The key findings highlight:

- Collaborative filtering and SVD are effective for sparse datasets.
- CUR provides a flexible, computationally efficient alternative.

Evaluation metrics such as Normalized Mean Absolute Error (NMAE), precision, and recall demonstrate the strengths and trade-offs of each algorithm, providing insights into their applicability in real-world recommendation systems.

Chapter 2

Introduction

Recommender systems are algorithms designed to predict and suggest items (products, movies, songs, etc.) to users based on their preferences, past behavior, or similarities with other users. They have become a crucial part of the modern digital landscape, providing personalized experiences in e-commerce, streaming platforms, and content curation services. By helping users discover relevant content from large datasets, recommender systems enhance user engagement, satisfaction, and conversion rates. For example, in e-commerce, they suggest products that users are likely to purchase, while on streaming platforms like Netflix or Spotify, they recommend movies or music based on individual tastes.

2.1 Types of Recommender Systems

- **Collaborative Filtering (CF):**
  Collaborative filtering is based on the idea that users who have agreed in the past will agree in the future. It operates by comparing a user's preferences with those of other users and recommending items that similar users have liked. There are two main types of collaborative filtering:

  - *User-based CF:* Identifies users who are similar to the target user and recommends items based on what those similar users have liked.
  - *Item-based CF:* Finds items similar to the ones the user has already rated highly and recommends those.

  Collaborative filtering excels in capturing the "wisdom of the crowd" but can struggle with sparse data and the "cold start" problem (difficulty recommending items to new users or with little interaction data).

- **Content-Based Filtering (CBF):**
  Content-based filtering recommends items based on their features and the user's past preferences. For example, if a user liked a particular movie, content-based systems will recommend similar movies based on genre, director, actors, or keywords. While content-based methods are less susceptible to the cold start problem, they may result in a narrower set of recommendations as they focus only on items similar to what the user has already interacted with.

- **Hybrid Methods:**
  Hybrid recommender systems combine both collaborative filtering and content-based filtering techniques to mitigate the drawbacks of each method. By leveraging the strengths of both, hybrid methods aim to provide more accurate and diverse recommendations.

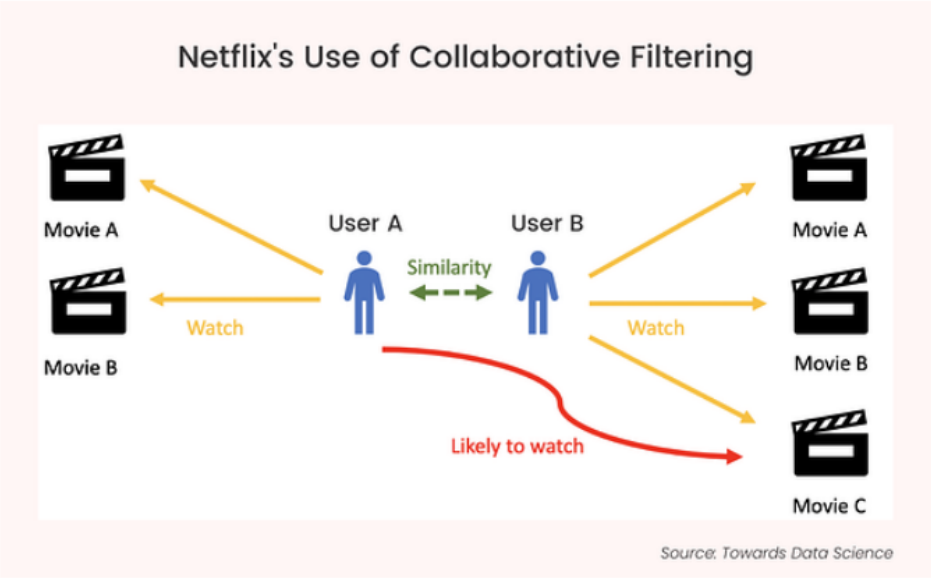

Figure 2.1: Basic Flow of a Recommender System

2.2 Why Focus on Collaborative Methods and Matrix Factorization in This Project

This project primarily focuses on collaborative filtering and matrix factorization because they have shown strong performance in scenarios where rich user-item interaction data is available. Collaborative filtering's ability to recommend based on user similarity and its flexibility across different domains makes it a popular choice for large-scale recommender systems.

Furthermore, matrix factorization techniques, such as Singular Value Decomposition (SVD) and CUR decomposition, allow us to handle large, sparse datasets efficiently by reducing dimensionality and uncovering latent factors driving user preferences. These methods are especially useful in improving the performance of recommender systems, particularly when dealing with sparse ratings matrices, which is a common challenge in real-world applications.

By focusing on these techniques, the project aims to demonstrate the effectiveness of collaborative filtering and matrix factorization in providing accurate and scalable recommendations in environments where user preferences and behavior are key to understanding item relevance.

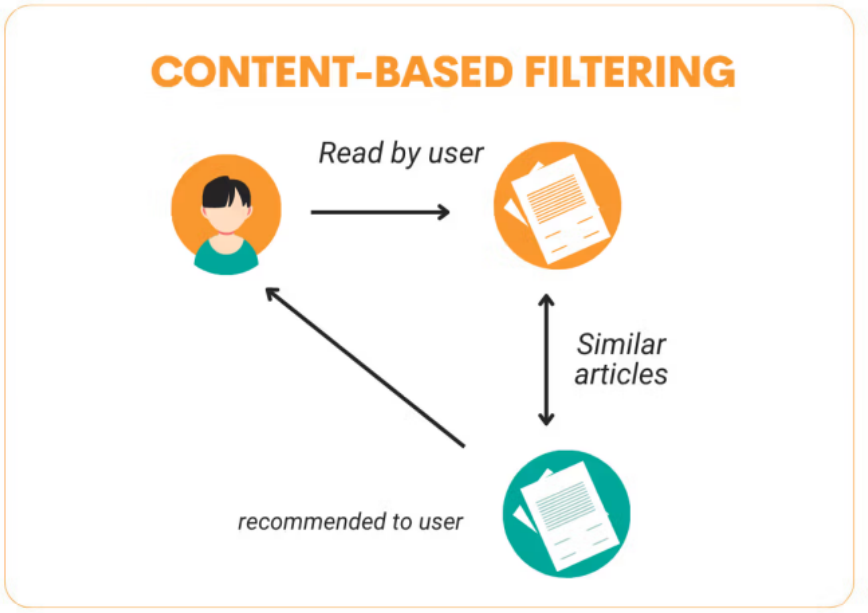

Figure 2.2: Design Overview of the Matrix Factorization Model

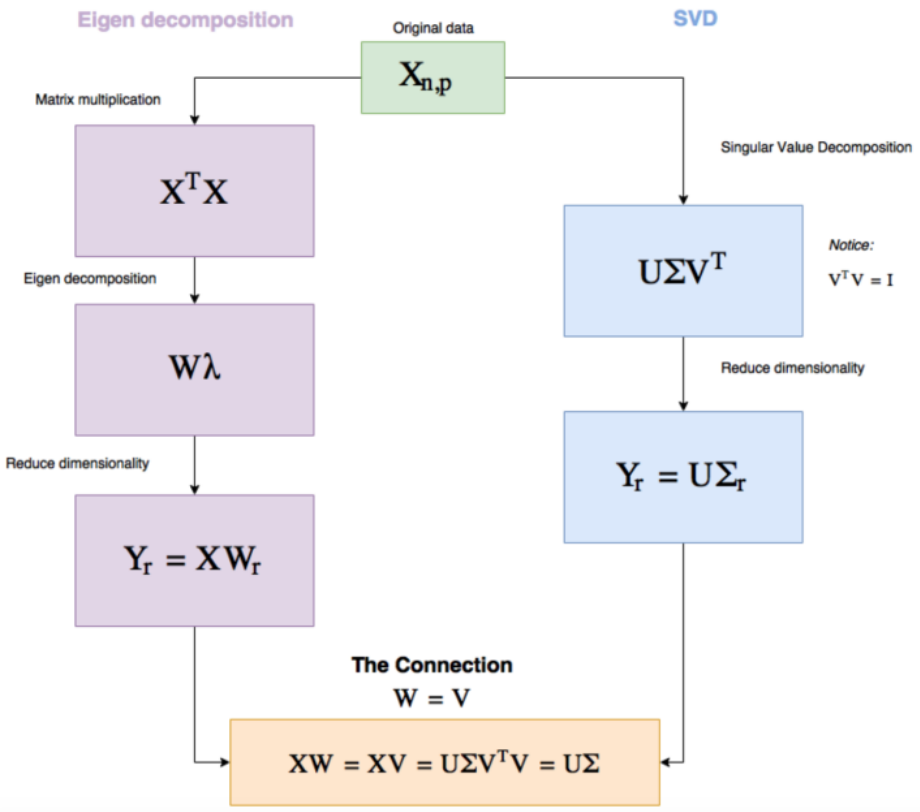

Figure 2.3: Flow of the Collaborative Filtering Approach

Chapter 3

Dataset Details

The Jester Online Joke Recommender Dataset (https://goldberg.berkeley.edu/jester-data/) is a widely used dataset in recommender system research, offering insights into how users rate jokes. It contains 4.1 million ratings from 73,421 users who rated 100 jokes, with ratings ranging from -10.00 to +10.00. The dataset was collected between April 1999 and May 2003 and is freely available for research purposes with proper acknowledgment.

3.1 Dataset Structure

The dataset is structured as a user-item matrix, where:

- Rows represent users, and columns represent jokes.
- Each entry in the matrix corresponds to a user's rating for a particular joke.
- Missing or null ratings are marked as "99."

Key statistics of the dataset:

- The matrix has dimensions of 73,421 $\times$ 100 (users $\times$ jokes).
- Ratings range from -10.00 to +10.00, reflecting the user's enjoyment or dislike of a joke.

3.2 Significance of Dense Submatrices

Within this large matrix, certain submatrices are "dense," meaning that most users have rated those items. For instance:

- The submatrix formed by columns {5, 7, 8, 13, 15, 16, 17, 18, 19, 20} is dense, as nearly all users have rated those jokes.

This density indicates that some jokes are universally popular, making it easier to identify patterns of preferences and similarities among users.

3.3 Justification for Dataset Suitability

The Jester dataset is highly suitable for collaborative filtering-based recommender systems due to the following reasons:

- **Large, Sparse, and Real-world Matrix:** Collaborative filtering algorithms thrive on this type of data, as user preferences can be compared to generate recommendations.

- **Continuous Ratings:** Unlike integer-based ratings, the continuous scale (-10.00 to +10.00) provides finer granularity, allowing for more nuanced predictions.
- **Matrix Factorization:** The sparsity of the matrix allows researchers to explore techniques like matrix factorization to handle missing data, a common challenge in real-world recommender systems.
- **Long History and Large User Base:** The dataset's long collection period and large, anonymized user base make it ideal for testing collaborative filtering algorithms, especially for addressing challenges like the cold start problem (new users/items with no historical data) and data sparsity.

3.4 Summary

In summary, the Jester dataset is particularly well-suited for testing collaborative filtering and matrix factorization techniques due to its:

- Size and sparsity.
- Continuous nature of ratings.
- Real-world applicability.

It serves as an excellent resource for experimenting with recommender system challenges and gaining a deeper understanding of user preferences in a real-world context.

Chapter 4

Methodology Overview

This chapter outlines the methodology used to implement and evaluate recommender system algorithms, focusing on Singular Value Decomposition (SVD) and CUR decomposition.

4.1 Data Cleaning and Preparation

- The dataset is read from an Excel file and transformed into a utility matrix.
- This matrix serves as the basis for matrix factorization techniques like SVD and CUR.

4.2 Implementation of Algorithms

4.2.1 Singular Value Decomposition (SVD)

- SVD decomposes a matrix into three components: U, Σ, and V.
- This technique reduces the matrix's dimensionality, making it suitable for recommendation tasks by approximating the original matrix.

4.2.2 CUR Decomposition

- CUR selects a subset of rows and columns to approximate the original matrix.
- This method is computationally efficient, especially for large datasets.

4.3 Step-by-Step Code Breakdown

4.3.1 SVD Class

1. **Reading the Data:**
   - The `dataframe()` method reads the utility matrix from the Excel file.
2. **Matrix Calculations:**
   - The `Ucalc()` and `Vcalc()` methods calculate the U and V matrices by multiplying the utility matrix and its transpose, then finding eigenvalues and eigenvectors.
   - The `Sigma()` method computes the Σ matrix by taking square roots of the eigenvalues.
3. **Dimensionality Reduction:**

- The `DimentionReduction()` method retains only the most significant singular values, which capture the majority of the data's energy.

4. **Error Estimation:**
   - **RMSE:** Root Mean Squared Error evaluates the approximation accuracy.
   - **TopK:** Error estimation for the top K values.
   - **Spearman's Correlation:** Measures rank correlation between predicted and actual data.

4.3.2 CUR Class

1. **Reading the Data:**
   - The `dataframe()` method loads the utility matrix from the Excel file.
2. **Probability Distribution:**
   - The `probdistribution()` method computes the probabilities for selecting rows and columns.
3. **Matrix Construction:**
   - `C_calc()` and `R_calc()` methods construct the C and R matrices by randomly selecting columns and rows based on probabilities.
   - The `U_calc()` method computes the U matrix using the C and R matrices and SVD-like approximations.
4. **Error Estimation:**
   - Similar to SVD, the `ErrorEstimate()` method calculates RMSE, TopK error, and Spearman's correlation.

4.4 Code Flow

4.4.1 SVD Flow

1. **Read Data:** Load the matrix from an Excel file.
2. **Calculate U and V:** Perform matrix decomposition into U, Σ, and V.
3. **Error Estimation:** Compute RMSE, TopK, and Spearman's correlation for evaluation.

4.4.2 CUR Flow

1. **Read Data:** Load the matrix from an Excel file.
2. **Probability Distribution:** Calculate probabilities for selecting rows and columns.
3. **Matrix Construction:** Create C, R, and U matrices.
4. **Error Estimation:** Evaluate the approximation using RMSE, TopK, and Spearman's correlation.

4.5 Usage

4.5.1 SVD

- Load the utility matrix from an Excel file.
- Decompose the matrix into U, Σ, and V using SVD.
- Optionally, reduce dimensionality by retaining significant singular values.
- Estimate errors using RMSE, TopK, and Spearman's correlation.

4.5.2 CUR

- Load the utility matrix from the Excel file.
- Compute the probability distribution for selecting rows and columns.
- Construct C, R, and U matrices.
- Estimate errors in approximation using RMSE, TopK, and Spearman's correlation.

Chapter 5

Results and Analysis

This chapter compares the performance of two matrix factorization methods, Singular Value Decomposition (SVD) and CUR (Column-Row), based on accuracy and computational efficiency. The analysis evaluates their suitability for tasks like recommendation systems.

5.1 Performance Metrics

5.1.1 Accuracy Comparison

The accuracy of the approximation is evaluated using the following metrics:

- **Root Mean Squared Error (RMSE):** Measures the deviation between predicted and actual values. Lower RMSE values indicate better approximation accuracy.
- **TopK Error:** Evaluates the error for the top K predicted values, critical for recommendation systems. Smaller errors indicate better performance.
- **Spearman's Rank Correlation:** Quantifies the rank correlation between predicted and actual values. Higher values indicate better prediction quality.

5.1.2 Computational Efficiency

The computational efficiency of the algorithms is assessed based on:

- **Time Taken:** The time required to factorize the matrix and compute approximations.
- **Memory Usage:** The memory consumed during execution, including storage for input data and intermediate calculations.

5.2 Observations on Accuracy and Computational Efficiency

5.2.1 Accuracy

[a.]**SVD:**

1.
   - Consistently achieves lower RMSE and TopK errors compared to CUR.
   - Demonstrates higher Spearman's rank correlation, indicating superior approximation quality.
   - This makes SVD particularly suitable for applications requiring high accuracy, such as recommendation systems.
2. **CUR:**

- While less accurate than SVD, CUR provides reasonably good approximations for large-scale datasets.
- Its slightly higher RMSE and TopK errors, along with lower rank correlation, suggest trade-offs in accuracy.

5.2.2 Computational Efficiency

[a.]**CUR:**

1.
- Processes the matrix faster than SVD, making it more efficient in terms of runtime.
- Consumes less memory due to its focus on a subset of rows and columns, which reduces computational complexity.
- These characteristics make CUR a viable choice for real-time applications or environments with resource constraints.

2. **SVD:**

- Requires more processing time due to the decomposition of the entire matrix.
- Consumes significantly more memory, especially for large matrices.
- While computationally intensive, its accuracy justifies its use in applications where precision is crucial.

References

1. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30-37.
   DOI: https://doi.org/10.1109/MC.2009.263

2. Drineas, P., & Mahoney, M. W. (2005). On the Approximability of Low-Rank Matrix Approximation Problems. *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '05)*, 1-10.
   DOI: https://doi.org/10.1137/1.9781611973751.1

3. Candes, E. J., & Recht, B. (2009). Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, *9*(6), 717-772.
   DOI: https://doi.org/10.1007/s10208-009-9045-5