Data612-Project 3

Natalie Kalukeerthie & Anna Moy

2025-06-20

```
# Load libraries
library(recommenderlab)
library(Matrix)
library(dplyr)
library(ggplot2)
```

What is Singular Value Decomposition (SVD)?

A method used in linear algebra to decompose a matrix into 3 simpler matrices which makes it easier to analyze and manipulate the data.

he Singular Value Decomposition (SVD) of a matrix R is:

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

Where:

- R is the user–item rating matrix,
- U is the matrix of user latent features, User's preference
- Σ (or D) is the diagonal matrix of singular values, Importance table
- V^T is the transpose of the item latent feature matrix. Traits

Singular Value Decomposition(SVD) is used on the MovieLens dataset to uncover latent patterns in user preferences and movie characteristics. Each row in the dataset represents a user and each column represents a movie, and the entries of the ratings the user has for each movie. There are missing ratings for some of these movies.

Here, we load the built-in MovieLense dataset from the recommenderlab package to create a basic recommender system.For computational ease, subset the first 500 rows to reduce runtime and memory usage.

```
# Load MovieLens data (built-in dataset in recommenderlab)
data(MovieLense)

# For computational ease, use first 500 users (rows) and all movies (columns)
ml_subset <- MovieLense[1:500, ]
```

First we'll convert the realRatingMatrix object to a dense numeric matrix with user ratings, calculate the average rating for each user (row-wise while ignoring missing values). And then replace all missing ratings (NAs) in each user's row with their average rating. This step will fill the sparse matrix to make it suitable for SVD, which requires a complete matrix.

```r
# Convert to rating matrix (dense)
rating_mat <- as(ml_subset, "matrix")

# Impute missing values by user's mean rating (mean-centering)
user_means <- rowMeans(rating_mat, na.rm = TRUE)

# Create a complete matrix with NAs replaced by user's mean rating
rating_mat_imputed <- rating_mat

for(i in 1:nrow(rating_mat_imputed)){
  rating_mat_imputed[i, is.na(rating_mat_imputed[i, ])] <- user_means[i]
}
```

Subtract each user's mean rating from their imputed ratings to center the data around zero.

Mean-centering helps remove user-specific rating biases, which improves the quality of matrix factorization.

```r
# Optional: center the data by subtracting user means to remove bias
rating_mat_centered <- rating_mat_imputed - user_means
```

While performing Singular Value Decomposition on the centered rating matrix, we'll keep only the top k=20 singular values and vectors to reduce dimensionality and capture the most important features.

This factorization breaks down the rating matrix into user and item latent factor matrices as well. This step improves efficiency and makes a prediction.

```r
# Perform SVD on centered matrix
# We can limit the number of singular values (k) to say 20
k <- 20
svd_result <- svd(rating_mat_centered, nu = k, nv = k)
```

Next, we'll rebuild the rating matrix from the reduced matrices (u, d, v) keeping only the top k components and then add back the user mean ratings to reverse the centering.

We kept the ratings to be a scale of 1 to 5 to maintain realistic rating values.

```r
# Reconstruct the approximate ratings matrix using top k singular values
approx_ratings <- svd_result$u %*% diag(svd_result$d[1:k]) %*% t(svd_result$v)

# Add back the user means to reverse centering
approx_ratings <- approx_ratings + user_means

# Clip predictions to rating scale (1 to 5)
approx_ratings[approx_ratings > 5] <- 5
approx_ratings[approx_ratings < 1] <- 1
```

Using evaluationScheme, we split data into training (80%) and testing (20%) sets.given = 10 is used here so each user keeps 10 known ratings for training; the rest are hidden for testing.

Then we split the training data and parts of the test data that are known and unknown to the model for evaluation.

```r
# Now evaluate predictions against known ratings in a test set
# We'll create a train/test split first:

set.seed(123)
split <- evaluationScheme(ml_subset, method = "split", train = 0.8, given = 10, goodRating = 4)

train_data <- getData(split, "train")
test_known <- getData(split, "known")
test_unknown <- getData(split, "unknown")
```

We'll Convert the training data to a matrix and repeat the imputation and mean-centering on the training set to make sure the factorization is based on known data only.

```r
# Convert train data matrix for SVD again:
train_mat <- as(train_data, "matrix")

# Repeat imputation on training data for SVD (same steps)
train_user_means <- rowMeans(train_mat, na.rm = TRUE)
train_mat_imputed <- train_mat
for(i in 1:nrow(train_mat_imputed)){
  train_mat_imputed[i, is.na(train_mat_imputed[i, ])] <- train_user_means[i]
}
train_mat_centered <- train_mat_imputed - train_user_means
```

Here factorization is used on the centered training matrix using SVD, limited to top k features.

```r
# SVD on train data
svd_train <- svd(train_mat_centered, nu = k, nv = k)

# Reconstruct full rating prediction matrix
pred_mat <- svd_train$u %*% diag(svd_train$d[1:k]) %*% t(svd_train$v)
pred_mat <- pred_mat + train_user_means

# Clip to rating range
pred_mat[pred_mat > 5] <- 5
pred_mat[pred_mat < 1] <- 1
```

Finally we extract actual ratings from the unknown test set and corresponding predicted ratings from the reconstructed matrix, while calculating RMSE, MSE, and MAE on only those test entries where actual ratings exist.

```r
# Now evaluate predictions only on test_unknown ratings:
test_unknown_mat <- as(test_unknown, "matrix")

# Get predicted ratings for the test unknown entries
pred_ratings <- pred_mat

# Calculate evaluation metrics: RMSE, MAE, MSE
# Only on entries where test_unknown_mat has actual ratings

actuals <- test_unknown_mat[!is.na(test_unknown_mat)]
preds <- pred_ratings[!is.na(test_unknown_mat)]
```

```r
rmse <- sqrt(mean((preds - actuals)^2))
mse <- mean((preds - actuals)^2)
mae <- mean(abs(preds - actuals))

cat(sprintf("SVD Matrix Factorization Results:\nRMSE: %.4f\nMSE: %.4f\nMAE: %.4f\n", rmse, mse, mae))
```

```
## SVD Matrix Factorization Results:
## RMSE: 1.2962
## MSE: 1.6802
## MAE: 1.0362
```

Summary

The SVD matrix factorization approach achieved an RMSE of about 1.30, an MSE of 1.68, and an MAE of 1.04 on the test set. These metrics indicate that the model is reasonably effective at predicting user ratings for movies it has not yet seen, though there is still some prediction error.

The RMSE of 1.30 suggests that on average, the predicted ratings deviate from actual user ratings by about 1.3 points on a 1–5 rating scale. While this level of error shows the model captures general user preferences well, it also highlights the challenge of predicting individual user tastes perfectly.

By using SVD, the model leverages latent factors derived from user-item interactions, enabling it to uncover underlying patterns in the data beyond explicit features like genres. However, some loss of accuracy may stem from imputing missing values and the basic mean-centering approach, which may not capture all complex user behaviors.

Overall, SVD matrix factorization offers a powerful dimensionality reduction technique that applies both complexity and prediction performance. Future improvements could involve more advanced imputation strategies, incorporating additional side information, or experimenting with other matrix factorization methods such as Alternating Least Squares (ALS) to further improve recommendation accuracy.

Potential issues

Issues in which we have to watch out for in our model is imputing missing ratings using the mean which could create bias predictions towards the average preferences since SVD is not able to handle missing data. We did use mean centering to remove user bias but does not address item biases for movies which higher or lower ratings. In our model we used K = 20 but we may want to find out the most optimal without losing important information or overfitting the model. SVD is not scalable if we are using this for a large dataset and may want to use Alternative Least Squares.

Possible Improvements or considerations

Ways we can improve our model is using the Alternative Least Squares(ALS) and see how it compares to Singular Value Decomposition. ALS has a built in regularization to prevent the data from being overfitted and does not require any imputation of missing data compared to SVD. Another option is using the baseline predictor and capture the user and item bias before we conduct factorization on the data. Adding regularization in the model can avoid it from being overfitted.