Data 612 - Project 4

Anna Moy & Natalie Kalukeerthie

2025-06-29

```r
# Load libraries
library(recommenderlab)
library(Matrix)
```

The Jester 5k dataset contains 5k users and 100 jokes and the ratings ranges from -10 to 10.

We load the built- in Jester 5k data set from the recommederlab package to create a basic recommender system.

```r
# Load the built-in Jester5k dataset (5,000 users × 100 jokes)
data(Jester5k, package = "recommenderlab")
rrm <- Jester5k
```

Using evaluation Scheme, we split data into training (80%) and testing (20%) sets.given = 10 is used here so each user keeps 10 known ratings for training and the rest is "unknown: and used for test predictions.

```r
# 80-20 split, 10 observed ratings per user retained for "known"
set.seed(123)
scheme <- evaluationScheme(rrm, method = "split", train = 0.8, given = 10, goodRating = 3)
```

We train two models which is the user-user collaborative filtering (UBCF) and Singluar Value Decomposition (SVD).

User based collaborative filtering (UBCF) - recommends items liked by user with similar tastes Singular Value Decomposition (SVD) - matrix factorization that finds hidden patterns in user preferences

```r
# User-User Collaborative Filtering (UBCF)
rec_UBCF <- Recommender(getData(scheme, "train"), method = "UBCF")

# SVD-based model
rec_SVD <- Recommender(getData(scheme, "train"), method = "SVD")
```

Predicts the rating for both the UBCF and SVD and compared the Root Mean Square Error (RSME), Mean Absolute Error (MAE) and Mean Squared Error (MSE) to determine which one has a better prediction.

```r
# Predict ratings
pred_UBCF <- predict(rec_UBCF, getData(scheme, "known"), type = "ratings")
pred_SVD  <- predict(rec_SVD,  getData(scheme, "known"), type = "ratings")

# Evaluate accuracy (RMSE/MAE)
acc <- rbind(
  UBCF = calcPredictionAccuracy(pred_UBCF, getData(scheme, "unknown")),
  SVD  = calcPredictionAccuracy(pred_SVD,  getData(scheme, "unknown"))
)
print(acc)
```

```
##          RMSE      MSE      MAE
## UBCF 4.883778 23.85128 3.807471
## SVD  4.556161 20.75860 3.559910
```

Overall the results of Singular Value Decomposition (SVD) outperforms the User based collaborative filtering(UBCF) as the RMSE, MSE and MAE was the lowest. It's lower RMSE and MAE suggest it not only reduces large errors but also more reliable and consistent recommendation across users. RMSE measures the average error which is 4.556161 and MAE indicates on average the predictions were off by 3.559910 points. SVD uses matrix factorization, which captures hidden relationships between users and items. It is able to handle sparsity by generalizing but at the same time with higher accuracy there is a risk that we are recommending items that are very similar. UBCF relies on direct overlap between users and sparse data is challenging for it to handle. UBCF is more prone to overfitting in spare dataset since it relies heavily on overlap in user profiles. We can prevent overfitting by adding regularization and cross validation to the model.

```
#Diversity by Shuffling
# Top 5 recommendation from original SVD model
topN_SVD <- predict(rec_SVD, getData(scheme, "known"), type = "topNList", n = 5)
original_items <- as(topN_SVD, "list")

# Shuffle top-5 recommendations to simulate a more diverse rerank
set.seed(123)
diverse_list <- lapply(original_items, function(x) {
  sampled_chars <- sample(x, length(x))            # shuffle the joke IDs (characters)
  matched_indices <- match(sampled_chars, colnames(rrm))  # convert to integer indices
  return(matched_indices)
})

# Rebuild new top5List object
pred_diverse <- new("topNList",
                    items = diverse_list,
                    itemLabels = colnames(rrm),
                    n = as.integer(5))
```

We introduced diversity shuffling in the data. Taking the top 5 recommendations for the original SVD model and shuffled the data. Then reconstructed a new object wit the shuffled recommendation. By conducting shuffling we reduce accuracy and increase serendipity and novelty.

```
# Gives you precision and recall of the original SVD top 5 listed.
# Evaluate original SVD top-N recommendations
eval_orig <- evaluate(scheme, method = "SVD", type = "topNList", n = 5)
```

```
## SVD run fold/sample [model time/prediction time]
##   1  [0.059sec/0.059sec]
```

```
avg_prec_orig <- avg(eval_orig, "prec")
avg_rec_orig <- avg(eval_orig, "rec")

cat(sprintf("Original Precision: %.3f, Recall: %.3f\n", avg_prec_orig, avg_rec_orig))
```

```
## Original Precision: 2.895, Recall: 2.895
##  Original Precision: 2.105, Recall: 2.105
##  Original Precision: 22.603, Recall: 22.603
##  Original Precision: 62.367, Recall: 62.367
##  Original Precision: 89.970, Recall: 89.970
##  Original Precision: 0.579, Recall: 0.579
##  Original Precision: 0.145, Recall: 0.145
##  Original Precision: 0.145, Recall: 0.145
##  Original Precision: 0.031, Recall: 0.031
##  Original Precision: 5.000, Recall: 5.000
```

```
# print some approximation or note on diverse recommendations accuracy here
```

```
# Approximate precision for diverse SCD recommendations:
# Check how many recommended items appear in the unknown/test data for each user

unknown_data <- getData(scheme, "unknown")
diverse_lists <- as(pred_diverse, "list")

precision_diverse <- mean(sapply(1:length(diverse_lists), function(u) {
  recommended <- diverse_lists[[u]]
  relevant <- which(!is.na(unknown_data[u, ]))  # items user rated in test
  if (length(recommended) == 0) return(NA)
  length(intersect(recommended, relevant)) / length(recommended)
}), na.rm = TRUE)

cat(sprintf("Approximate Diverse Recommendations Precision: %.3f\n", precision_diverse))
```

```
## Approximate Diverse Recommendations Precision: 0.000
```

```
#Compares the original SVD and Diverse SVD
```

```
unknown_data <- getData(scheme, "unknown")
diverse_lists <- as(pred_diverse, "list")

# Precision
precision_diverse <- mean(sapply(seq_along(diverse_lists), function(u) {
  recommended <- diverse_lists[[u]]
  relevant <- which(!is.na(unknown_data[u, ]))
  if (length(recommended) == 0) return(NA)
  length(intersect(recommended, relevant)) / length(recommended)
}), na.rm = TRUE)

# Recall
recall_diverse <- mean(sapply(seq_along(diverse_lists), function(u) {
  recommended <- diverse_lists[[u]]
```

```r
  relevant <- which(!is.na(unknown_data[u, ]))
  if (length(relevant) == 0) return(NA)
  length(intersect(recommended, relevant)) / length(relevant)
}), na.rm = TRUE)

unknown_data <- getData(scheme, "unknown")
diverse_lists <- as(pred_diverse, "list")

# Precision
precision_diverse <- mean(sapply(seq_along(diverse_lists), function(u) {
  recommended <- diverse_lists[[u]]
  relevant <- which(!is.na(unknown_data[u, ]))
  if (length(recommended) == 0) return(NA)
  length(intersect(recommended, relevant)) / length(recommended)
}), na.rm = TRUE)

# Recall
recall_diverse <- mean(sapply(seq_along(diverse_lists), function(u) {
  recommended <- diverse_lists[[u]]
  relevant <- which(!is.na(unknown_data[u, ]))
  if (length(relevant) == 0) return(NA)
  length(intersect(recommended, relevant)) / length(relevant)
}), na.rm = TRUE)

library(proxy)

item_sim <- similarity(rrm, method = "cosine", which = "items")
item_sim <- as.matrix(item_sim)  # <-- convert to matrix

intra_list_similarity <- function(item_list, sim_matrix) {
  if (length(item_list) < 2) return(0)
  pairs <- combn(item_list, 2)
  sims <- apply(pairs, 2, function(pair) sim_matrix[pair[1], pair[2]])
  mean(sims)
}

original_items_int <- lapply(original_items, function(x) match(x, colnames(rrm)))

ils_values <- sapply(original_items_int, function(x) intra_list_similarity(x, sim_matrix = item_sim))

avg_ils_orig <- mean(ils_values)
cat("Average intra-list similarity (original):", avg_ils_orig, "\n")
```

```
## Average intra-list similarity (original): 0.7772745
```

```r
cat("Original SVD diversity (intra-list similarity): ", avg_ils_orig, "\n")
```

```
## Original SVD diversity (intra-list similarity):  0.7772745
```

```r
avg_ils_diverse <- mean(sapply(diverse_list, function(x) intra_list_similarity(x, sim_matrix = item_sim
cat("Diverse SVD diversity (intra-list similarity): ", avg_ils_diverse, "\n")
```

```
## Diverse SVD diversity (intra-list similarity):  0.7772745
```

Summary

It is taking the diverse (shuffled) recommendation match the actual items rated by users in the test dataset.Shuffling the recommendation is a good way to increase diversity and novelty and avoiding redundancy. The results shows the precision is 0.00 which indicates there is no overlap between diverse list and the test item. For the top 5 recommendation accuracy we looked at precision and recall. The SVD model shows high variability since precision/recall scores changes from .031 to 89.97 depending on the evaluation fold or user. This indicates some users received high relevant recommendation which others did not.

The original SVD intra-list similarity is .7779104 which means it has low diversity and most of the recommendations tend to be similar jokes. In the Diverse SVD is because there are some recommendation which has fewer than two items and it did not calculate property.

Errors

Potential errors with using shuffling is that it destroys relevance completely. And in most causes this will show the precision as 0 since the random items are not going to match what the users like.

Improvements

When we conduct shuffling of items it does not introduce new or different items but recommending it in different orders. Replacing it with items that are less similar or less popular jokes may increase diversity. We received NA on our results for the diverse SVD which we can improve by ignoring the NA values when we are calculating the average.

In an online environment, we could also evaluate user engagement metrics such as click-through rate (CTR), dwell time, or conversion rates to assess the true effectiveness of diverse recommendations. Additional experiments could include A/B testing of standard vs. diversified recommendation lists to observe changes in real user behavior. A reasonable design would involve randomly assigning users to different recommendation strategies and logging their interactions over time. This would allow us to measure both short-term satisfaction (e.g., CTR) and long-term retention or novelty discovery. Moreover, metrics such as serendipity (how surprising and useful a recommendation is) and novelty (how dissimilar a recommended item is from a user's history) could be incorporated if we could access full browsing/rating histories online.