MovieLens Rating Prediction Project

Yap Kah Yong

2025-08-28

Introduction

In this project, a movie rating prediction algorithm is developed using the MovieLens 10M dataset. The goal is to predict user ratings for movies based on historical data. The model will be trained and validated using the `edx` dataset, and the final evaluation will be performed on the `final_holdout_test` set using **Root Mean Squared Error (RMSE)**.

I follow proper machine learning practices: - Will **not** use `final_holdout_test` during training or model selection. - Use a train/test split on `edx` to tune and evaluate the model. - Only apply the final model to `final_holdout_test` once, at the end.

The algorithm uses a **simple baseline model** with regularized user and item biases:

where: - u = global average rating - b_u = user bias (adjusted with regularization) - b_i = item (movie) bias (adjusted with regularization)

This model is fast, memory-efficient, and avoids the computational bottlenecks of matrix factorization or collaborative filtering.

Some original code from Edx I placed them into chunck, so the codes more clean and easy to run for marking.

```
train_idx <- createDataPartition(edx$rating, p = 0.8, list = FALSE)
train <- edx[train_idx, ]
valid <- edx[-train_idx, ]
```

```
lambdas <- c(50,100,200,300)
results <- data.frame(lambda = numeric(), rmse = numeric())

for (lambda in lambdas) {
  mu <- mean(train$rating)  # Use train mean

  # User and movie biases from train set
  user_bias <- train %>%
    group_by(userId) %>%
    summarise(b_u = sum(rating - mu) / (lambda + n()), .groups = 'drop')

  movie_bias <- train %>%
    group_by(movieId) %>%
    summarise(b_i = sum(rating - mu) / (lambda + n()), .groups = 'drop')

  # Predict on validation set
  pred_valid <- valid %>%
    left_join(user_bias, by = "userId") %>%
    left_join(movie_bias, by = "movieId") %>%
```

```r
    mutate(
      b_u = ifelse(is.na(b_u), 0, b_u),
      b_i = ifelse(is.na(b_i), 0, b_i),
      pred = mu + b_u + b_i,
      pred = pmin(pmax(pred, 0.5), 5)
    )

  rmse <- RMSE(pred_valid$pred, pred_valid$rating)
  results <- add_row(results, lambda = lambda, rmse = rmse)
}

# Choose best lambda
best_lambda <- results$lambda[which.min(results$rmse)]
cat("Best lambda:", best_lambda, "\n")
```

```
## Best lambda: 50
```

```r
cat("Best rmse:", results$rmse[which.min(results$rmse)], "\n")
```

```
## Best rmse: 0.8838516
```

```r
# Train final model on full edx
mu_final <- mean(edx$rating, na.rm = TRUE)
cat("Global mean (mu_final):", round(mu_final, 3), "\n")
```

```
## Global mean (mu_final): 3.512
```

```r
cat("Using best_lambda:", best_lambda, "\n")
```

```
## Using best_lambda: 50
```

```r
# User and movie biases
user_bias_final <- edx %>%
  group_by(userId) %>%
  summarise(b_u = sum(rating - mu_final) / (best_lambda + n()), .groups = 'drop')

movie_bias_final <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = sum(rating - mu_final) / (best_lambda + n()), .groups = 'drop')

# --- Add time trend (SAFE VERSION) ---
min_time <- min(edx$timestamp, na.rm = TRUE)

# Compute time in days and user-specific centering
edx <- edx %>%
  group_by(userId) %>%
  mutate(
    user_mean_rating = mean(rating),
    time_days = (timestamp - min_time) / (60*60*24),        # days since start
    user_mean_time = mean(time_days)                        # user's average time
```

```
  ) %>%
  ungroup()

# Fit time slope using centered time: (time_days - user_mean_time)
user_time <- edx %>%
  group_by(userId) %>%
  summarise(
    n = n(),
    time_var = var(time_days),
    # Only fit if user has variation
    beta_t_raw = ifelse(n > 1 & time_var > 1,
                        cov(time_days - user_mean_time, rating - user_mean_rating, use = "complete.obs")
                        0),
    # Clip to very small range
    beta_t = pmin(pmax(beta_t_raw, -0.001), 0.001),
    user_mean_time = mean(user_mean_time),  # Save for prediction
    .groups = 'drop'
  ) %>%
  select(userId, beta_t, user_mean_time)


# --- Final Prediction ---
predictions <- final_holdout_test %>%
  select(userId, movieId, rating, timestamp) %>%
  left_join(user_bias_final, by = "userId") %>%
  left_join(movie_bias_final, by = "movieId") %>%
  left_join(user_time, by = "userId") %>%
  mutate(
    b_u = coalesce(b_u, 0),
    b_i = coalesce(b_i, 0),
    beta_t = coalesce(beta_t, 0),
    user_mean_time = coalesce(user_mean_time, mean(edx$time_days)),  # safe default
    time_days = (timestamp - min_time) / (60*60*24),
    pred_raw = mu_final + b_u + b_i + beta_t * (time_days - user_mean_time),
    pred = pmin(pmax(pred_raw, 0.5), 5)
  )

# Diagnostics
cat("Raw prediction range:", range(predictions$pred_raw), "\n")
```

```
## Raw prediction range: 0.04933693 5.67804
```

```
cat("Clamped to 0.5:", sum(predictions$pred == 0.5), "\n")
```

```
## Clamped to 0.5: 51
```

```
cat("Clamped to 5.0:", sum(predictions$pred == 5.0), "\n")
```

```
## Clamped to 5.0: 697
```

```r
cat("Prediction mean:", round(mean(predictions$pred), 3), "\n")
```

```
## Prediction mean: 3.503
```

```r
# Final RMSE
rmse_final <- RMSE(predictions$pred, predictions$rating)
cat("Final Holdout Test RMSE:", round(rmse_final, 5), "\n")
```

```
## Final Holdout Test RMSE: 0.87952
```


I Yap Kah Yong Acknowledgements to GroupLens for this online Harvard Edx project –

Source: https://grouplens.org/datasets/movielens/10m/