Collaborative Filtering in R with **{recommenderlab}**

recommenderlab : supported algorithms

- User-based collaborative filtering (UBCF)
- Item-based collaborative filtering (IBCF)
- SVD with column-mean imputation (SVD)
- Funk SVD (SVDF)
- Association rule-based recommender (AR)
- Popular items (POPULAR)
- Randomly chosen items for comparison (RANDOM)
- Re-recommend liked items (RERECOMMEND)
- Hybrid recommendations (HybridRecommender)

recommenderlab Evaluation methods

- Train/test split
- Cross-validation
- Repeated bootstrap sampling

recommenderlab Evaluation measures

- Rating errors: MSE, RMSE, MAE
- Top-N recommendations: TPR/FPR (ROC), precision and recall

Case study with the MovieLense Data

Rating data sets from the MovieLens web site (http://movielens.org).

Load the package

```
library("recommenderlab")
```

Load the data

```
data("MovieLense")
```

Note : MovieLense is an object of class "realRatingMatrix"

Look at the data structure

```
str(MovieLense)
slotNames(MovieLense)
str(as(MovieLense, "data.frame"))
```

Visualise the data

```
image(MovieLense [1:25, 1:25])
```

Examine a few records

```
head(as(MovieLense, "data.frame"))
```

Check the number of ratings per user

```
hist(rowCounts(MovieLense))
```

Check the number of ratings per movie

```
hist(colCounts(MovieLense))
```

Average user ratings

```
hist(rowMeans(MovieLense))
```

Retrieve the number of users

```
nusers=dim(MovieLense)[1]
nusers
```

Retrieve the number of movies

```
nmovies=dim(MovieLense)[2]
nmovies
```

Investigate the number of movies rated by users

```
summary(rowCounts(MovieLense))
```

Visualise a part of the data

```
image(MovieLense[sample(nusers,25),sample(nmovies,25)])
```

Check how the movies have been rated

```
vector_ratings <- as.vector(MovieLense@data)
```

Check what are the unique values of the ratings

```
unique(vector_ratings)
```

Check the count for each rating value

```
table_ratings <- table(vector_ratings)
table_ratings
barplot(table_ratings)
```

Repeat after removing the un-rated items

```
vector_ratings2 <- vector_ratings[vector_ratings != 0]
table_ratings2 <- table(vector_ratings2)
table_ratings2
barplot(table_ratings2)
```

Check out the available Recommender algorithms

```
recommenderRegistry$get_entries(dataType = "realRatingMatrix")
```

Examine the similarity of a few users

```
similarity_users <- similarity(MovieLense[1:4, ], method = "cosine",
which = "users")
as.matrix(similarity_users)
```

Examine the similarity of a few items

```
similarity_items <- similarity(MovieLense[, 1:4], method = "cosine",
which = "items")

as.matrix(similarity_items)
```

Create an evaluation scheme by splitting the data and specifying other parameters

```
evlS <- evaluationScheme(MovieLense, method="split", train=0.9,
given=12); evlS

trg <- getData(evlS, "train"); trg

test_known <- getData(evlS, "known"); test_known

test_unknown <- getData(evlS, "unknown"); test_unknown
```

Create UBCF recommender model with the training data

```
rcmnd_ub <- Recommender(trg, "UBCF")
```

Create predictions for the test users

```
pred_ub <- predict(rcmnd_ub, test_known, type="ratings"); pred_ub
```

Evaluate model accuracy for the unknown set of test users

```
acc_ub <- calcPredictionAccuracy(pred_ub, test_unknown)

as(acc_ub,"matrix")
```

Compare the results

```
as(test_unknown, "matrix")[1:8,1:5]

as(pred_ub, "matrix")[1:8,1:5]
```

Repeat the process with a IBCF model

```
rcmnd_ib <- Recommender(trg, "IBCF")

pred_ib <- predict(rcmnd_ib, test_known, type="ratings")

acc_ib <- calcPredictionAccuracy(pred_ib, test_unknown)

acc <- rbind(UBCF = acc_ub, IBCF = acc_ib); acc
```

Get the top recommendations

```
pred_ub_top <- predict(rcmnd_ub, test_known); pred_ub

movies <-as(pred_ub_top, "list")

movies[1]
```

Extras: *Pre-Processing*

Read the data into R

```
movies <-
read.csv("c:/r_data/movies/movies.csv",stringsAsFactors=FALSE)

ratings <- read.csv("c:/r_data/movies/ratings.csv")
```

Take a look at the data

```
head(movies)

head(ratings)
```

Create a user-item rating matrix

```
      MovieMatrx <- as.matrix(dcast(ratings, userId~movieId, value.var =
"rating", na.rm=FALSE))[, -1]
```

Convert rating matrix into recommenderlab matrix

```
MovieRating <- as(MovieMatrx, "realRatingMatrix")
```