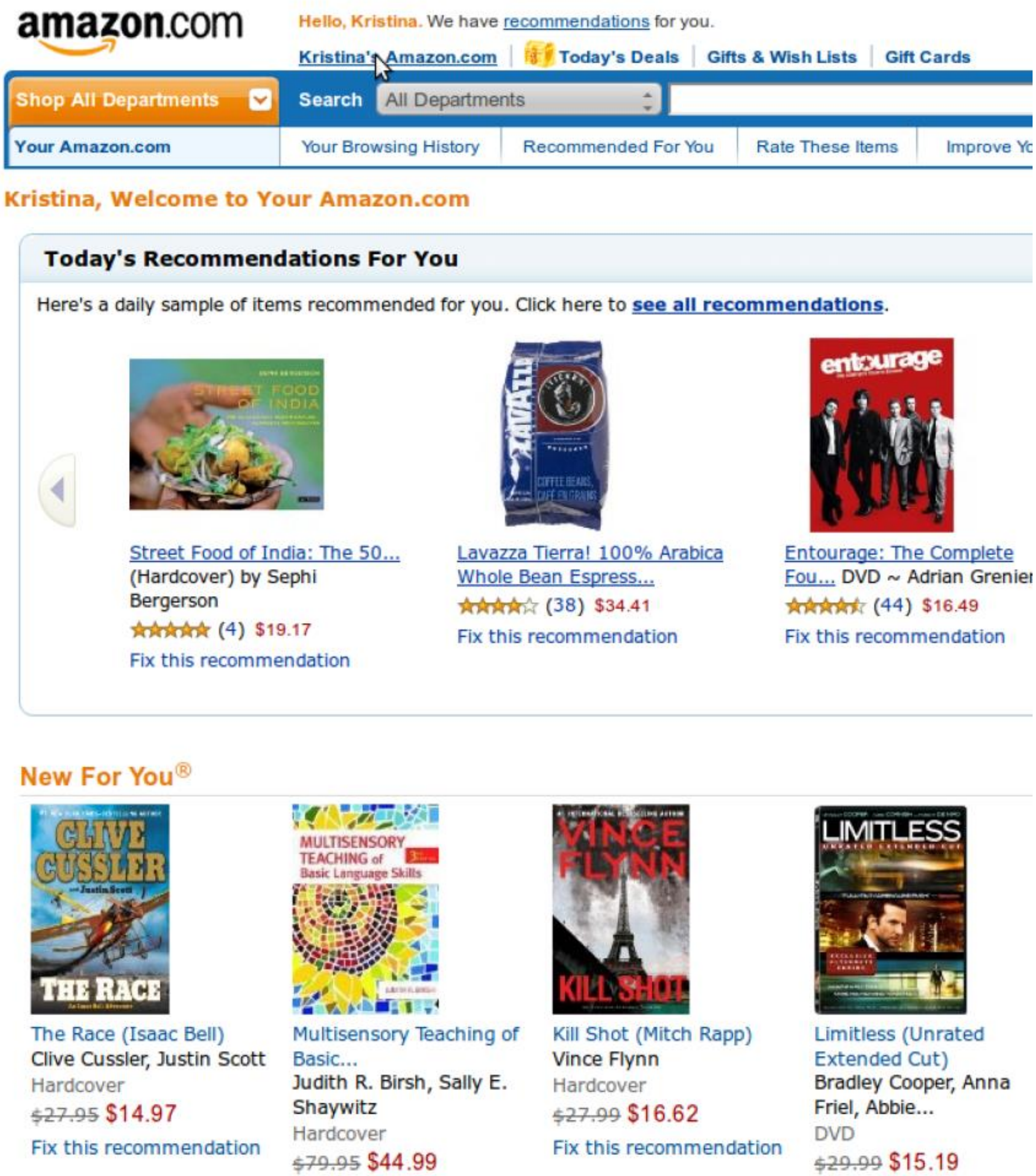
recommender system

Data Mining Practicum

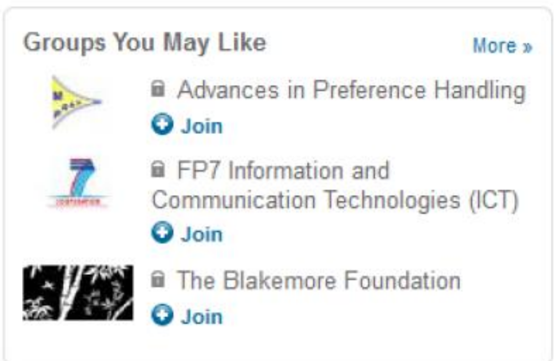
Presented by Hyebong Choi



amazon.com
Hello, Kristina. We have recommendations for you.
Kristina's Amazon.com | Today's Deals | Gifts & Wish Lists | Gift Cards
Shop All Departments
Search All Departments
Your Amazon.com
Your Browsing History
Recommended For You
Rate These Items
Improve Yo
Kristina, Welcome to Your Amazon.com
Today's Recommendations For You
Here's a daily sample of items recommended for you. Click here to see all recommendations.
Street Food of India: The 50... (Hardcover) by Sephi Bergerson
(4) $19.17
Fix this recommendation
Lavazza Tierra! 100% Arabica Whole Bean Espress...
(38) $34.41
Fix this recommendation
Entourage: The Complete Fou... DVD ~ Adrian Grenier
(44) $16.49
Fix this recommendation
New For You®
The Race (Isaac Bell)
Clive Cussler, Justin Scott
Hardcover
$27.95 $14.97
Fix this recommendation
Multisensory Teaching of Basic...
Judith R. Birsh, Sally E. Shaywitz
Hardcover
$79.95 $44.99
Kill Shot (Mitch Rapp)
Vince Flynn
Hardcover
$27.99 $16.62
Fix this recommendation
Limitless (Unrated Extended Cut)
Bradley Cooper, Anna Friel, Abbie...
DVD
$29.99 $15.19


NETFLIX
Suggestions (1141)
Suggestions by Genre
Rate Movies
Rate Genres
Movies You've Rated (262)
Movies You'll Love
You have 1141 Suggestions from 262 ratings.
The Fugitive (1993)
Wrongfully convicted of murdering his wife, Dr. Richard Kimble (Harrison Ford) escapes custody after a ferocious train accident (one of the most thrilling wrecks ever filmed). While Kimble tries to find the true murderer, gung-ho U.S. Marshal Samuel Gerard (Tommy Lee Jones, in an Oscar-winning performance) is hot on Kimble's trail, pulling out all stops to put him back behind bars.
Starring: Harrison Ford, Tommy Lee Jones
Director: Andrew Davis
Genre: Action & Adventure
MPAA: PG-13
4.7 Our best guess for Michael
4.1 Customer Average
Recommended based on 8 ratings
The Fugitive
Because you enjoyed:
Patriot Games
Indiana Jones and the Last Crusade
Die Hard
Add
Play
+ All
Not Interested
Not Interested
Spacehunter
RoboCop:

In the Social Web



RECOMMENDER SYSTEMS

- Recommender systems apply statistical and knowledge discovery techniques to the problem of making product recommendations (Sarwar et al., 2000).

Advantages of recommender systems (Schafer et al., 2001):

✓ Improve conversion rate: Help customers find a product she/he wants to buy.

✓ Cross-selling: Suggest additional products.

✓ Improve customer loyalty: Create a value-added relationship.

✓ Improve usability of software!

TYPES OF RECOMMENDER SYSTEMS

- ✓ Content-based filtering: Consumer preferences for product attributes.
- ✓ Collaborative filtering: Mimics word-of-mouth based on analysis of

(Ansari et al., 2000)

CONTENT-BASED APPROACH

1. Analyze the objects (documents, video, music, etc.) and extract attributes/features (e.g., words, phrases, actors, genre).
2. Recommend objects with similar attributes to an object the user likes.

MUSIC GENOME PROJECT

Musical Attributes	Low =====>=====>=====> High										
Level of vibrato in Lead Vocal	0	1	2	3	4	5	6	7	8	9	10
Lead Vocal sound: Nasal	0	1	2	3	4	5	6	7	8	9	10
Lead Vocal sound: Thickness	0	1	2	3	4	5	6	7	8	9	10
Prominence of Percussion	0	1	2	3	4	5	6	7	8	9	10
Prominence of Horn Section	0	1	2	3	4	5	6	7	8	9	10
Use of Woodwinds (Saxes etc..)	0	1	2	3	4	5	6	7	8	9	10
Prominence of vocal harmony	0	1	2	3	4	5	6	7	8	9	10
Vocal Backups gender male -to- female		1	2	3	4	5	6	7	8	9	10
Use of Vocal call-and-response harmony	0	1	2	3	4	5	6	7	8	9	10
Amount of distortion on the electric guitar	0	1	2	3	4	5	6	7	8	9	10
Prominence of Electric Piano	0	1	2	3	4	5	6	7	8	9	10
Song form: Number of distinct sections	0	1	2	3	4	5	6	7	8	9	10
Amount of rhythmic syncopation	0	1	2	3	4	5	6	7	8	9	10

“The Music Genome Project is an effort to capture the essence of music at the fundamental level using almost 400 attributes to describe songs and a complex mathematical algorithm to organize them."

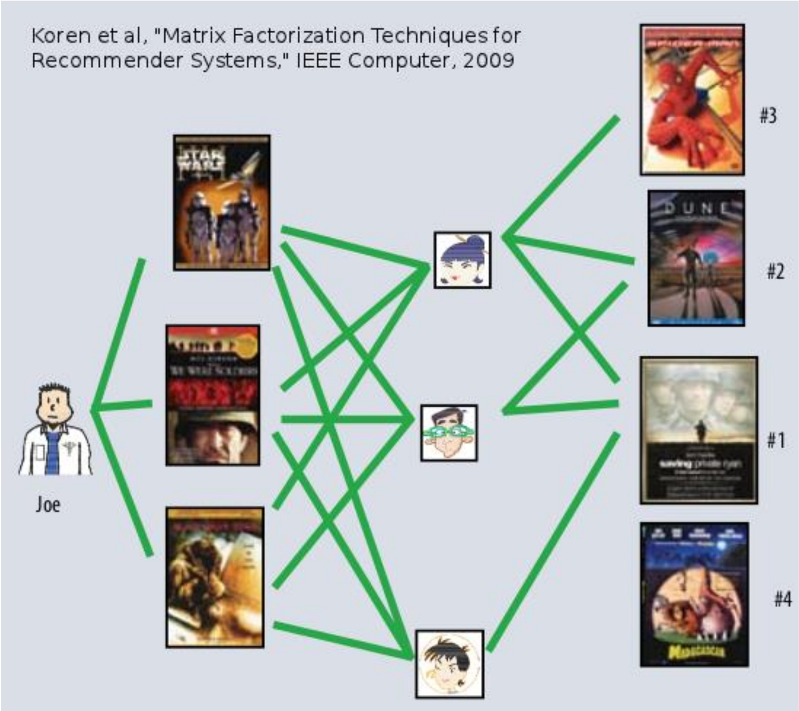
http://en.wikipedia.org/wiki/Music_Genome_Project

Limitation

- Need to encode contents into some meaningful features
  - Which represent user’s taste
- Quality judgement
  - Content is not the only reason to prefer certain item other others
- Limit the chance to expose new diverse item to users
  - No surprises

COLLABORATIVE FILTERINF (CF)

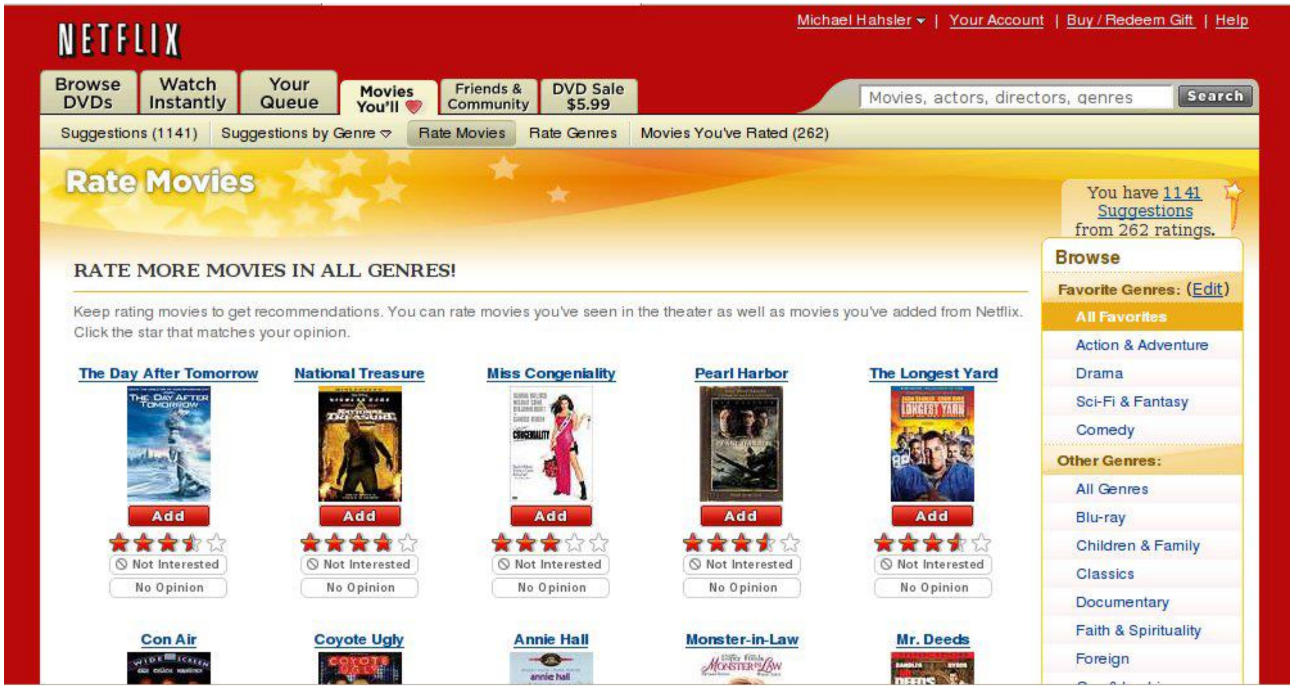
- **Memory-based CF**
- **Model-based CF**



Make automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many other users (collaboration).

Assumption: those who agreed in the past tend to agree again in the future.

DATA COLLECTION



Data sources:

- ✓ Explicit: ask the user for ratings, rankings, list of favorites, etc.
- ✓ Observed behavior: clicks, page impressions, purchase, uses, downloads, posts, tweets, etc.

What is the incentive structure?

DATA COLLECTION

Predicted rating of unrated movies (Breese et al., 1998)

A top-N list of unrated (unknown) movies ordered by predicted rating/score (Deshpande and Karypis, 2004)

Example of User-rating Matrix

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You
A	2		2	4	5	
B	5		4			1
C			5		2	
D		1		5		4
E			4			2
F	4	5		1		

TYPES OF CF ALGORITHMS

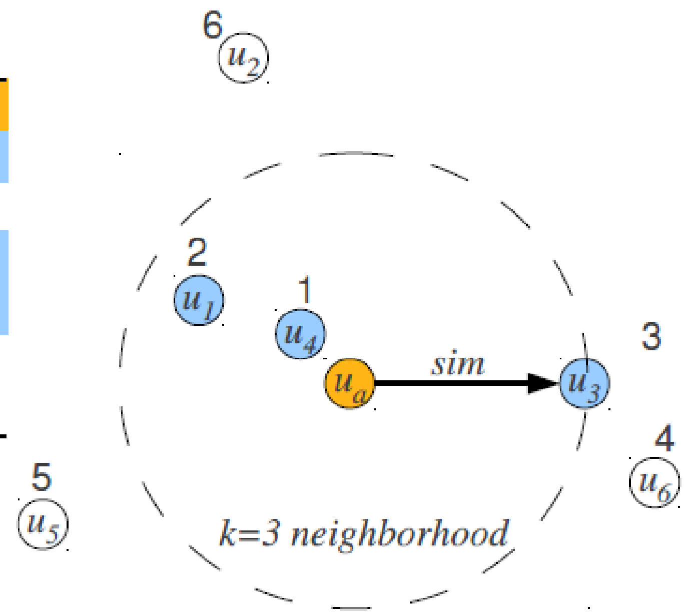
- Memory-based: Find similar users (user-based CF) or items(item-based CF) to predict missing ratings.
- Model-based: Build a model from the rating data (clustering, latent semantic structure, etc.) and then use this model to predict missing ratings.

USER-BASED CF (UCBF)

- Produce recommendations based on the preferences of similar users (Goldberg et al., 1992; Resnick et al., 1994; Mild and Reutterer, 2001).

	i_1	i_2	i_3	i_4	i_5	i_6
u_a	?	?	4.0	3.0	?	1.0
u_1	?	4.0	4.0	2.0	1.0	2.0
u_2	3.0	?	?	?	5.0	1.0
u_3	3.0	?	?	3.0	2.0	2.0
u_4	4.0	?	?	2.0	1.0	1.0
u_5	1.0	1.0	?	?	?	?
u_6	?	1.0	?	?	1.0	1.0
	3.5	4.0			1.3	

Recommendations: i_2, i_1



1. Find k nearest neighbors for the user in the user-item matrix.
2. Generate recommendation based on the items liked by the k nearest neighbors. E.g., average ratings or use a weighting scheme.

SIMILARITY MEASURES

$$u_{ik} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}}$$

Pearson Correlation

$$\cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^{m} v_{ik}^2 \sum_{k=1}^{m} v_{jk}^2}}$$

Cosine Similarity

User-based CF

```
load('ratings.RData')

ratings

##   The Avengers Sherlock Transformer Matrix Titanic Me Before You
## A            2       NA           2      4       5           NA
## B            5       NA           4     NA      NA            1
## C           NA       NA           5     NA       2           NA
## D           NA        1          NA      5      NA            4
## E           NA       NA           4     NA      NA            2
## F            4        5          NA      1      NA           NA

pearsonCor <- function(x, y){
  x_mean <- mean(x, na.rm = T)
  y_mean <- mean(y, na.rm = T)
  idx <- !is.na(x) & !is.na(y)
  if(sum(idx) == 0) return(NA)
  x_new <- x[idx]
  y_new <- y[idx]
  sum((x_new- x_mean) * (y_new -y_mean)) /
    sqrt( sum( (x_new - x_mean)**2) * sum( (y_new-y_mean) **2) )
}
```

User-based CF

```
u <- ratings['E', ]
sim <- apply(ratings, 1, function(x) {
  pearsonCor(u, x)
})
sim
```

```
##          A          B          C          D          E          F
## -1.0000000  0.8741573  1.0000000 -1.0000000  1.0000000         NA
```

```
k = 2
library(doBy)
k_neighbors <- setdiff(which.maxn(sim, k+1), 5) ## delete user 'E' itself
k_recommend <- apply(ratings[k_neighbors,], 2, function(x) { mean(x, na.rm = T)})

k_recommend
```

```
##   The Avengers       Sherlock    Transformer         Matrix        Titanic
##            5.0            NaN            4.5            NaN            2.0
## Me Before You
##            1.0
```

```
k_recommend_final <- k_recommend[is.na(u)]

sort(k_recommend_final, decreasing = T)
```

```
## The Avengers      Titanic
##            5            2
```

Practice

- Make recommendation for other users, A, B, C, D, and F

USER-BASED CF (UCBF)

- Pearson correlation coefficient:

$$\mathrm{sim}_{\mathrm{Pearson}}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i \in I} x_i y_i - I\bar{\mathbf{x}}\bar{\mathbf{y}}}{(I-1)s_x s_y}$$

- Cosine similarity:

$$\mathrm{sim}_{\mathrm{Cosine}}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$$

- Jaccard index (only binary data):

$$\mathrm{sim}_{\mathrm{Jaccard}}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

where $\mathbf{x} = b_{u_x,\cdot}$ and $\mathbf{y} = b_{u_y,\cdot}$ represent the user's profile vectors and X and Y are the sets of the items with a 1 in the respective profile.

Problem

Memory-based. Expensive online similarity computation.

ITEM-BASED CF (ICBF)

- Produce recommendations based on the relationship between items in the user-item matrix (Kitts et al., 2000; Sarwar et al., 2001)

S	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
i_1	-	0.1	0	**0.3**	**0.2**	**0.4**	0	0.1
i_2	0.1	-	**0.8**	**0.9**	0	**0.2**	0.1	0
i_3	0	**0.8**	-	0	**0.4**	0.1	0.3	**0.5**
i_4	**0.3**	**0.9**	0	-	0	**0.3**	0	0.1
i_5	**0.2**	0	**0.7**	0	-	**0.2**	0.1	0
i_6	**0.4**	**0.2**	0.1	**0.3**	0.1	-	0	0.1
i_7	0	**0.1**	**0.3**	0	0	0	-	0
i_8	**0.1**	0	**0.9**	**0.1**	0	0.1	0	-
	-	0	4.56	2.75	-	2.67	0	-

$k=3$

$u_a=\{i_1, i_5, i_8\}$

$r_{ua}=\{2, ?,?,?,4,?,?, 5\}$

Recommendation: i_3

1. Calculate similarities between items and keep for each item only the values for the k most similar items.
2. Use the similarities to calculate a weighted sum of the user's ratings for related items.

$$\hat{r}_{ui} = \sum_{j \in s_i} s_{ij} r_{uj} / \sum_{j \in s_i} |s_{ij}|$$

Regression can also be used to create the prediction.

ITEM-BASED CF (ICBF)

Similarity measures:

- Pearson correlation coefficient, cosine similarity, jaccard index
- Conditional probability-based similarity (Deshpande and Karypis, 2004):

$$\mathrm{sim}_{\mathrm{Conditional}}(x, y) = \frac{\mathrm{Freq}(xy)}{\mathrm{Freq}(x)} = \hat{P}(y|x)$$

  where x and y are two items, $\mathrm{Freq}(\cdot)$ is the number of users with the given item in their profile.

Properties

- Model (reduced similarity matrix) is relatively small $(N \times k)$ and can be fully precomputed.
- Item-based CF was reported to only produce slightly inferior results compared to user-based CF (Deshpande and Karypis, 2004).
- Higher order models which take the joint distribution of sets of items into account are possible (Deshpande and Karypis, 2004).
- Successful application in large scale systems (e.g., Amazon.com)

MODEL-BASED CF

- **There are many techniques:**
  - Cluster users and then recommend items the users in the cluster closest to the active user like.
  - Mine association rules and then use the rules to recommend items (for binary/binarized data)
  - Define a null-model (a stochastic process which models usage of independent items) and then find significant deviation from the null-model.
  - Learn a latent factor model from the data and then use the discovered factors to find items with high expected ratings.

COLD START PROBLEM

What happens with new users where we have no ratings yet?

- ✓ Recommend popular items
- ✓ Have some start-up questions (e.g., "tell me 10 movies you love")

What do we do with new items?

- ✓ Content-based filtering techniques.
- ✓ Pay a focus group to rate them.

RECOMMENDERLAB: READING DATA

Movie Rating Data from [Kaggle Competition]

```
library(recommenderlab)

library(dplyr)

library(tidyr)

library(tibble)
ratings.df <- read.csv('train_v2.csv')
```

	ID	user	movie	rating
1	610739	3704	3784	3
2	324753	1924	802	3
3	808218	4837	1387	4
4	133808	867	1196	4
5	431858	2631	3072	5
6	895320	5410	2049	4
7	290408	1733	157	2
8	578446	3536	736	4
9	121351	780	379	3
10	538170	3312	1028	4

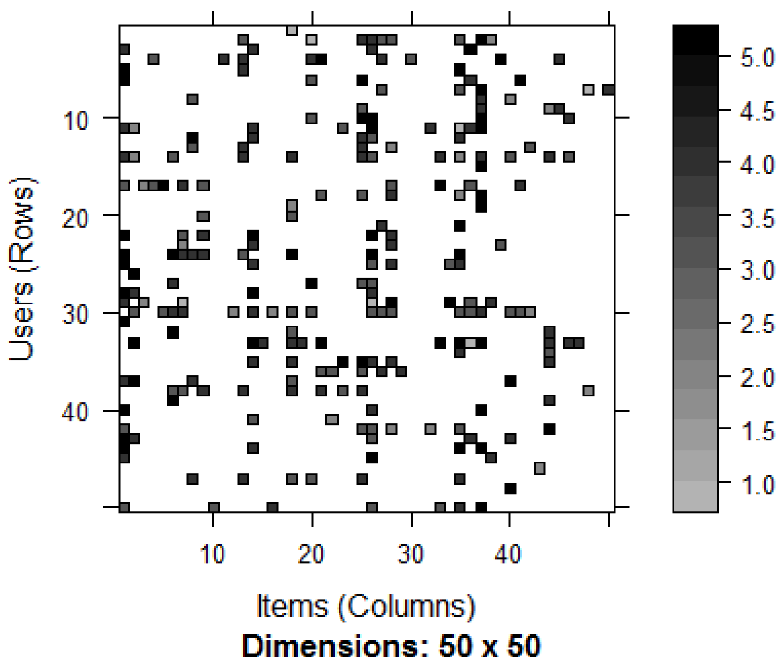
```r
rating_matrix <- ratings.df %>% select(-ID) %>%
  spread(movie, rating) %>%
  remove_rownames() %>%
  column_to_rownames(var = 'user')


rating_rrm <- as(as(rating_matrix, 'matrix'), 'realRatingMatrix')
rating_rrm <- rating_rrm[rowCounts(rating_rrm) > 50,
                         colCounts(rating_rrm) > 100]

# normalization of rating matrix
rating_rrm_norm <- normalize(rating_rrm)

image(rating_rrm[1:50, 1:50])

image(rating_rrm_norm[1:50, 1:50])
```

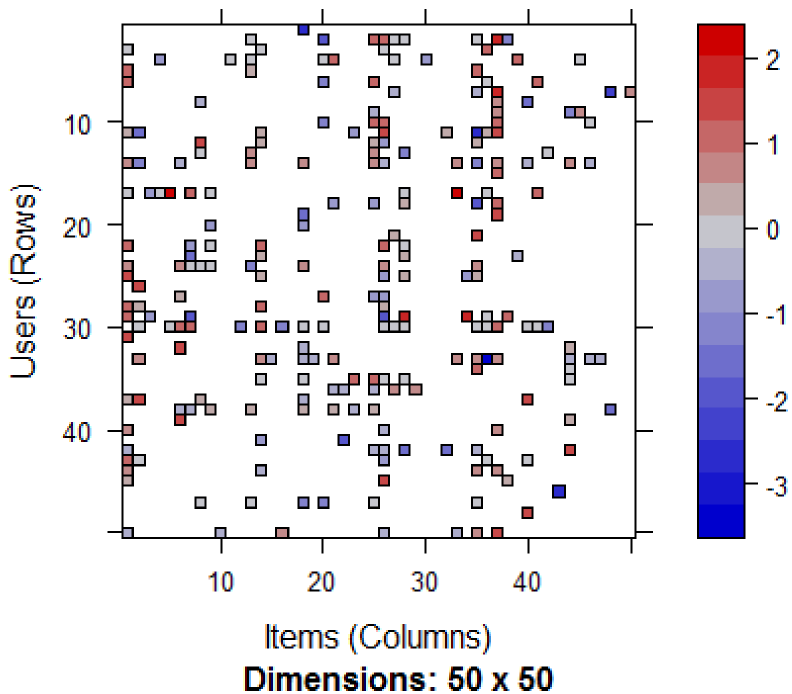
```r
# size comparison
print(object.size(rating_matrix), units = "auto")
```

```
## 85.1 Mb
```

```r
print(object.size(rating_rrm), units = "auto")
```

```
## 7.5 Mb
```





AVAILABLE METHOD IN THE PACKAGE

```
# see avaiable method
recommenderRegistry$get_entries(dataType = "realRatingMatrix")

## $ALS_realRatingMatrix
## Recommender method: ALS for realRatingMatrix
## Description: Recommender for explicit ratings based on latent factors, calculated by
alternating least squares algorithm.
## $IBCF_realRatingMatrix
## Recommender method: IBCF for realRatingMatrix
## Description: Recommender based on item-based collaborative filtering.
## Reference: NA
## Parameters:
##    k   method normalize normalize_sim_matrix alpha na_as_zero
## 1 30 "Cosine"  "center"                FALSE   0.5      FALSE
## $POPULAR_realRatingMatrix
## Recommender method: POPULAR for realRatingMatrix
## $RANDOM_realRatingMatrix
## Recommender method: RANDOM for realRatingMatrix
## Description: Produce random recommendations (real ratings).
## $SVD_realRatingMatrix
## Recommender method: SVD for realRatingMatrix
## Description: Recommender based on SVD approximation with column-mean imputation.
## $SVDF_realRatingMatrix
## Recommender method: SVDF for realRatingMatrix
## Description: Recommender based on Funk SVD with gradient descend.##
## $UBCF_realRatingMatrix
## Recommender method: UBCF for realRatingMatrix
## Description: Recommender based on user-based collaborative filtering.
```

RECOMMENDERLAB: CREATING RECOMMENDATIONS

```
ubcf_model <- Recommender(rating_rrm, method = 'UBCF')

recom <- predict(ubcf_model, rating_rrm[1:2, ])
recom

## Recommendations as 'topNList' with n = 10 for 2 users.

as(recom, 'list')

## $`2`
##  [1] "1258" "968"  "1333" "1073" "253"  "1387" "1219" "1278" "3543" "260"
##
## $`5`
##  [1] "260"  "1028" "919"  "1197" "1097" "1210" "1276" "1"    "912"  "3671"

recom <- predict(ubcf_model, rating_rrm[1:2, ], type = 'ratings')
recom

## 2 x 1790 rating matrix of class 'realRatingMatrix' with 3364 ratings.

as(recom, 'matrix')[,11:20]

##         14       15       16       17       18       19       20       21
## 2 3.731959 3.731959 3.731959 3.712457 3.731959 3.735959 3.731959       NA
## 5 3.067227 3.067227       NA 3.098053 2.970675 3.067227 3.067227 3.067227
##         22       24
## 2 3.736857 3.731959
## 5 3.067227       NA
```

RECOMMENDERLAB: COMPARE ALGORITHMS

```
## comparison
e <- evaluationScheme(rating_rrm, method="split", train=0.8, given=10)
r1 <- Recommender(getData(e, "train"), "UBCF")
r2 <- Recommender(getData(e, "train"), "IBCF")

p1 <- predict(r1, getData(e, "known"), type="ratings")
p2 <- predict(r2, getData(e, "known"), type="ratings")

error <- rbind(
  calcPredictionAccuracy(p1, getData(e, "unknown")),
  calcPredictionAccuracy(p2, getData(e, "unknown"))
)

rownames(error) <- c("UBCF","IBCF")

error

##          RMSE      MSE       MAE
## UBCF 1.031282 1.063542 0.8173145
## IBCF 1.729325 2.990565 1.3411870
```

REFERENCES

- Michael Hahsler, "recommenderlab: A Framework for Developing and Testing Recommendation Algorithms", Nov. 2011.
- R package document, https://cran.r-project.org/web/packages/recommenderlab/recommenderlab.pdf
- Hyebong Choi. (2016.9). An Artificial Neural Network for Local Library's Book Recommender System. Journal of Korean Institute of Information Technology, 14(9), 109-118.