MODEL-BASED COLLABORATIVE FILTERING

DEPARTMENT OF ECONOMICS
AND BUSINESS ECONOMICS
AARHUS UNIVERSITY

MORTEN BERG JENSEN
ASSOCIATE PROFESSOR

OUTLINE

A few extra comments to collaborative filtering

What is SVD (Singular value decomposition) and how can it be used for recommender systems?

Matrix Factorization

The SVD and SVD-FUNK approach to collaborative filtering

How to in Recommenderlab

EXPLICIT RATINGS

Explicit ratings

- Most commonly used (1 to 5, 1 to 7 Likert response scales)
- Typically only one rating per user and item, including time-stamp

Some research topics

- Data sparsity
  - Users not always willing to rate many items
  - How to stimulate users to rate more items?
- Which items have (not) been rated?
  - Ratings not missing at random
- Optimal granularity of scale
  - Indication that 10-point scale is better accepted in movie domain
  - An even more fine-grained scale was chosen in the Jester joke recommender

COLD START AND COLLABORATIVE FILTERING

How to recommend new items? What to recommend to new users?

A problem even on large platforms

New items

- Use of content-based recommendation could be a solution in the initial phase
    - This of course assumes that a new item will be already described by its attributes, which is not always the case

New user

- Ask/force new users to rate a set of items to build an initial profile
- Default voting: assign default values to items that only one of the two users to be compared has rated if we have reasonable criteria

DATA SPARSITY

In general a big problem to calculate reliable similarities between users or items

One “solution” is to restrict the data

```
#Training and test set At least 30 items evaluated or at least 100 users for each item
rates <- MovieLense[rowCounts(MovieLense) > 30, colCounts(MovieLense) > 100]
rates1 <- rates[rowCounts(rates) > 30,]
```

A second approach is to use matrix factorization and latent factor models by reducing the dimensionality

LATENT FACTOR MODELS

Explain ratings by characterizing both users and items on 20 to 100 factors inferred from the rating patterns.

For movies the discovered factors might measure obvious dimensions like

- Comedy
- Drama
- Amount of action
- Orientation to children
- and less well-defined dimensions or completely uninterpretable dimensions

DATA STRUCTURE

								k=3	
	movie 1	movie 2				movie m	dim 1	dim2	dim3
user 1	r11								
user 2	r21								
user n						rnm			

$$R_{nxm} \approx U_{nxk} V'_{kxm}$$

THE COMPONENTS

U_{nx3}

	dim1	dim2	dim3
user 1			
user 2			
user n			

V'_{3xm}

	movie 1	movie 2	movie 3		movie m
dim 1					
dim 2					
dim 3					

$$r_{12} \approx \sum_{j=1}^{3} u_{1j} v_{2j}$$

In mathematical terms it is called the dot product (inner product) of i´th user factor and the j´th movie factor and written as $u_i \cdot v_j$

Two hypothetical dimensions for movies

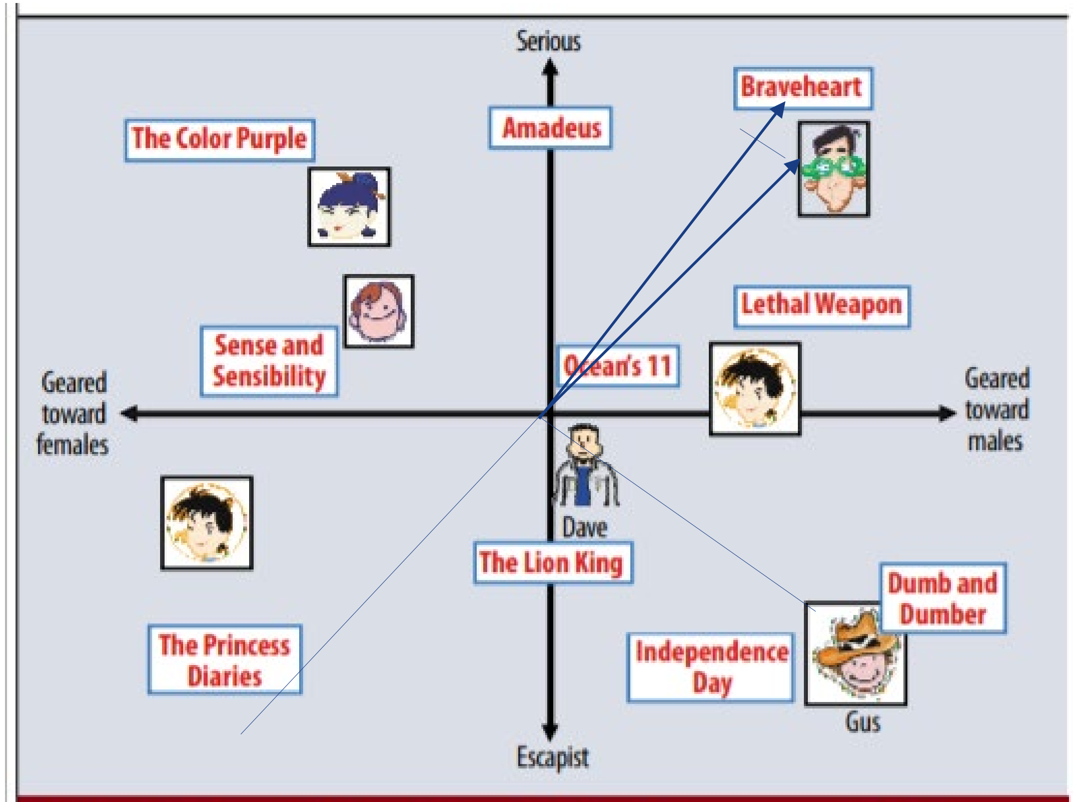


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

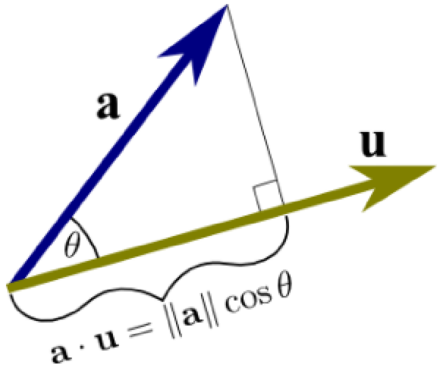
Arrow for Gus

Projection of Amadeus on the arrow for Gus

The inner product or dot products for the vectors for Gus and Amadeus

$$\mathbf{a} \cdot \mathbf{u} = \|\mathbf{a}\| \cos \theta,$$

here θ is the angle between $\mathbf{a}$ and $\mathbf{u}$.



The dot product of **a** with unit vector **u** is defined to be the projection of **a** in the direction of **u**

HOW TO FIND THE U AND V MATRICES?

MATRIX FACTORIZATION
SINGULAR VALUE DECOMPOSITION

SINGULAR VALUE DECOMPOSITION (SVD)

An example borrowed from Leskovec, STANFORD

$$\mathbf{A}_{[m \times n]} = \mathbf{U}_{[m \times r]} \, \Sigma_{[r \times r]} \, (\mathbf{V}_{[n \times r]})^T$$

- **A**: Input data matrix
  - *m* x *n* matrix (e.g., *m* documents, *n* terms)
- **U**: Left singular vectors
  - *m* x *r* matrix (*m* documents, *r* concepts)
- **Σ**: Singular values
  - *r* x *r* diagonal matrix (strength of each ‘concept’
    (*r* : rank of the matrix **A**)
- **V**: Right singular vectors
  - *n* x *r* matrix (*n* terms, *r* concepts)

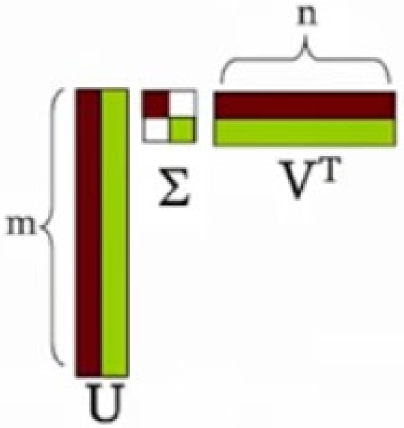
SVD - Properties

It is **always** possible to decompose a real matrix $\boldsymbol{A}$ into $\boldsymbol{A} = \boldsymbol{U} \Sigma \boldsymbol{V}^T$, where

- $\boldsymbol{U}, \Sigma, \boldsymbol{V}$: unique
- $\boldsymbol{U}, \boldsymbol{V}$: column orthonormal
  - $\boldsymbol{U}^T \boldsymbol{U} = \boldsymbol{I}$; $\boldsymbol{V}^T \boldsymbol{V} = \boldsymbol{I}$ ($\boldsymbol{I}$: identity matrix)
  - (Columns are orthogonal unit vectors)
- Σ: diagonal
  - Entries (**singular values**) are positive, and sorted in decreasing order ($\boldsymbol{\sigma_1 \geq \sigma_2 \geq \ldots \geq 0}$)

$A = U\,\Sigma\,V^T$ - example: Users to Movies

$$
\begin{array}{r}
 \\ \\ :i \\ \\ \\ \\ nce \\ \\
\end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\end{array}
=
$$



$A = U \Sigma V^T$ - example: Users to Movies

Sci-fi Romance

Strength of the sci-fi concept

$$
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array}
$$

$$
\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}
=
\begin{bmatrix} \mathbf{0.13} & 0.02 & -0.01 \\ \mathbf{0.41} & 0.07 & -0.03 \\ \mathbf{0.55} & 0.09 & -0.04 \\ \mathbf{0.68} & 0.11 & -0.05 \\ 0.15 & \mathbf{-0.59} & \mathbf{0.65} \\ 0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\ 0.07 & \mathbf{-0.29} & \mathbf{0.32} \end{bmatrix}
\times
\begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix}
\times
\begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\ 0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09 \end{bmatrix}
$$

Sci-fi

- **U**: user-to-concept similarities
- **V**: movie-to-concept similarities
- Σ : strength of each concept

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 \\ 0.41 & 0.07 \\ 0.55 & 0.09 \\ 0.68 & 0.11 \\ 0.15 & -0.59 \\ 0.07 & -0.73 \\ 0.07 & -0.29 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \end{bmatrix}$$

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\approx
\begin{bmatrix}
0.92 & 0.95 & 0.92 & 0.01 & 0.01 \\
2.91 & 3.01 & 2.91 & -0.01 & -0.01 \\
3.90 & 4.04 & 3.90 & 0.01 & 0.01 \\
4.82 & 5.00 & 4.82 & 0.03 & 0.03 \\
0.70 & 0.53 & 0.70 & 4.11 & 4.11 \\
-0.69 & 1.34 & -0.69 & 4.78 & 4.78 \\
0.32 & 0.23 & 0.32 & 2.01 & 2.01
\end{bmatrix}
$$

Frobenius norm:

$\|M\|_F = \sqrt{\Sigma_{ij} M_{ij}^2}$

$\|A-B\|_F = \sqrt{\Sigma_{ij} (A_{ij}-B_{ij})^2}$ is "small"

SVD gives us the best approximation to the rating matrix expressed by the Frobenius norm.

- **SSE** and **RMSE** are monotonically related:
  - $RMSE = \frac{1}{c}\sqrt{SSE}$ Great news: SVD is minimizing RMSE

Now we want to find users that are close to each other (neighbors) in the reduced space.

We assume a customer who likes the movie Matrix and we calculate the coordinates in the reduced space

Compactly, we have:

$$q_{concept} = q\,V$$

E.g.:

SciFi Romr

$$q = \begin{array}{c} \begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\ \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \times \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix} = \begin{array}{c} \text{SciFi-concept} \\ \begin{bmatrix} 2.8 & 0.6 \end{bmatrix} \end{array}$$

movie-to-concept similarities (V)

- **Observation:** User ***d*** that rated (*'Alien'*, *'Serenity'*) will be **similar** to user **q** that rated (*'Matrix'*), although ***d*** and **q** have **zero ratings in common**!

$$\begin{array}{cc} & \begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\ q= & \begin{bmatrix} 0 & 4 & 5 & 0 & 0 \end{bmatrix} \\ d= & \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad \begin{array}{c} \text{SciFi-concept} \\ \begin{bmatrix} 2.8 & 0.6 \end{bmatrix} \\ \begin{bmatrix} 5.2 & 0.4 \end{bmatrix} \end{array}$$

Zero ratings in common

Similarity ≠ 0

THE MISSING VALUE PROBLEM IN THE RATING MATRIX

Imputation

- Substitution with column means and then a traditional SVD

Use the known ratings and construct the U and V matrices based upon this limited information by optimization. Typically with a gradient descent approach

HOW TO FIND U AND V?

R

	movie 1	movie 2	movie 3		movie m
user 1	1		3		5
user 2		3	2		
user n	5	3			1

U_{nx3}

	dim1	dim2	dim3
user 1			
user 2			
user n			

V'_{3xm}

	movie 1	movie 2	movie 3		movie m
dim 1					
dim 2					
dim 3					

$$r_{12} \approx \sum_{j=1}^{3} u_{1j} v_{2j}$$

In mathematical terms it is called the dot product (inner product) of i´th user factor and the j´th movie factor and written as $u_i \cdot v_j$

MATRIX FACTORIZATION AND AN UNCONSTRAINED SOLUTION

In the basic matrix factorization model, the $m \times n$ ratings matrix R is approximately factorized into an $m \times k$ matrix U and an $n \times k$ matrix V, as follows:

$$R \approx UV^T \tag{3.13}$$

We don't require that U and V are orthonormal

From Equation 3.13, it follows that each rating r_{ij} in R can be approximately expressed as a dot product of the ith user factor and jth item factor:

$$r_{ij} \approx \overline{u_i} \cdot \overline{v_j} \tag{3.14}$$

MAIN IDEA – MINIMIZATION OF RMSE ON TRAINING DATA

Squared Frobenius norm if the matrix R was full

Unconstrained factorization

$$\text{Min} \quad J = \tfrac{1}{2}\left\| R - UV^T \right\|^2$$

If only a fraction of the elements in R are known then the minimization problem will only refer to the *known* ratings

If S consists of the set of known ratings, then the minimization problem can be formulated as

$$\text{Minimize } J = \frac{1}{2} \sum_{(i,j)\in S} e_{ij}^2 = \frac{1}{2} \sum_{(i,j)\in S} \left(r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js} \right)^2$$

subject to:

No constraints on U and V

S : observed entities where ratings exist

MAIN IDEA

The basic starting point is that user-item interactions and hence ratings are modeled as inner products in a joint latent factor space

u_{is} i'th user factor

v_{js} j'th item factor

$\hat{r}_{ij} = u_i \cdot v_j$ Estimated rating for an observed rating in the training set

Minimization of the regularized squared error on the set of *known* ratings

$$\min \sum (r_{ij} - u_i \cdot v_j)^2 + \lambda\left(\sum_i \|u_i\|^2 + \sum_j \|v_j\|^2\right)$$

TWO LEARNING ALGORITHMS

Stochastic gradient descent

Simon Funk

Alternating least squares

SVD FUNK

The algorithm loops through all ratings in the training set as described below

For each r(ij) a prediction is made and the error is computed $e_{ij} = r_{ij} - \hat{r}_{ij}$

For a given training case r(ij), we modify the parameters by moving in the opposite direction of the gradient, yielding

$$U - \gamma \nabla U -> U$$
$$V - \gamma \nabla V -> V$$

$$u_i + \gamma(e_{ij} v_j - \lambda u_i) -> u_i$$
$$v_j + \gamma(e_{ij} u_i - \lambda v_j) -> v_j$$

On Netflix data

$$\lambda = 0,02$$
$$\gamma = 0,005$$

EXTENDED WITH CORRECTION FOR USER AND ITEM BIAS

$$\hat{r}_{ij} = \mu + b_i + b_j + \overline{u}_i \cdot \overline{v}_j$$

In order to learn the model parameters ($b_i, b_j, \overline{u}_i, \overline{v}_j$) we minimize the regularised squared error

$$Min \sum (r_{ij} - (\mu + b_i + b_j + u_i \cdot v_j)^2 + \lambda(\sum_i \|u_i\|^2 + \sum_j \|v_j\|^2 + \sum_i \|b_i\|^2 + \sum_j \|b_j\|^2)$$

SVD FUNK

The algorithm loops through all ratings in the training set as described below

For each r(ij) a prediction is made and the error is computed $e_{ij} = r_{ij} - \hat{r}_{ij}$

For a given training case r(ij), we modify the parameters by moving in the opposite direction of the gradient, yielding

$$b_i + \gamma(e_{ij} - \lambda b_i) -> b_i$$

$$b_j + \gamma(e_{ij} - \lambda b_j) -> b_j$$

$$u_i + \gamma(e_{ij} v_j - \lambda u_i) -> u_i$$

$$v_j + \gamma(e_{ij} u_i - \lambda v_j) -> v_j$$

On Netflix data

$$\lambda = 0,02$$

$$\gamma = 0,005$$

THE NETFLIX PRIZE

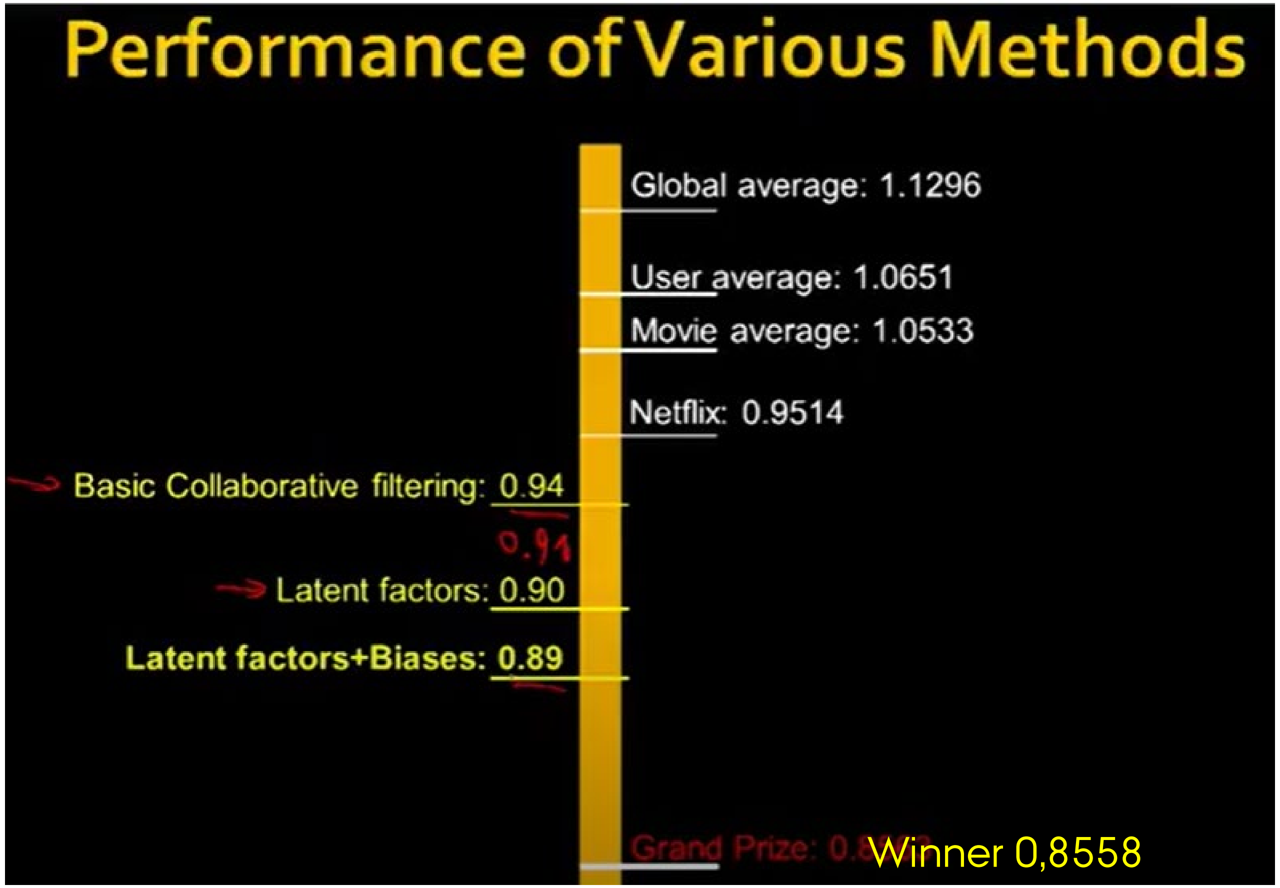
Netflix announced a million dollar prize

- Goal:
  - Beat their own "Cinematch" system by 10 percent
  - Measured in terms of the Root Mean Squared Error

- Effect:
  - Stimulated lots of research

Idea of SVD and matrix factorization picked up again

NETFLIX PRIZE



SVDF – REGULARIZATION
ITERATIVE PROCEDURE – GRADIENT DESCENT

```
$SVDF_realRatingMatrix
Recommender method: SVDF for realRatingMatrix
Description: Recommender based on Funk SVD with gradient descend (https://si
Reference: NA
Parameters:
   k gamma lambda min_epochs max_epochs min_improvement normalize verbose
1 10 0.015  0.001         50        200        0.000001  "center"   FALSE
```

Gamma (learning rate) : regulates how large a weight on the update of a coefficient in U,V

Lambda : is the regularization parameter

Regularization term
Is added to the objective
function to avoid
overfitting:

$$\frac{\lambda}{2}(\|U\|^2 + \|V\|^2)$$

ALTERNATING LEAST SQUARES

$$\min \sum (r_{ij} - u_i \bullet v_j)^2 + \lambda(\|U^2\| + \|V^2\|)$$

Notice that when one of these is taken as a constant the optimization problem is quadratic and can be optimally solved.

Rotate between fixing the u(i)'s to solve for the v(j)'s and vice versa.

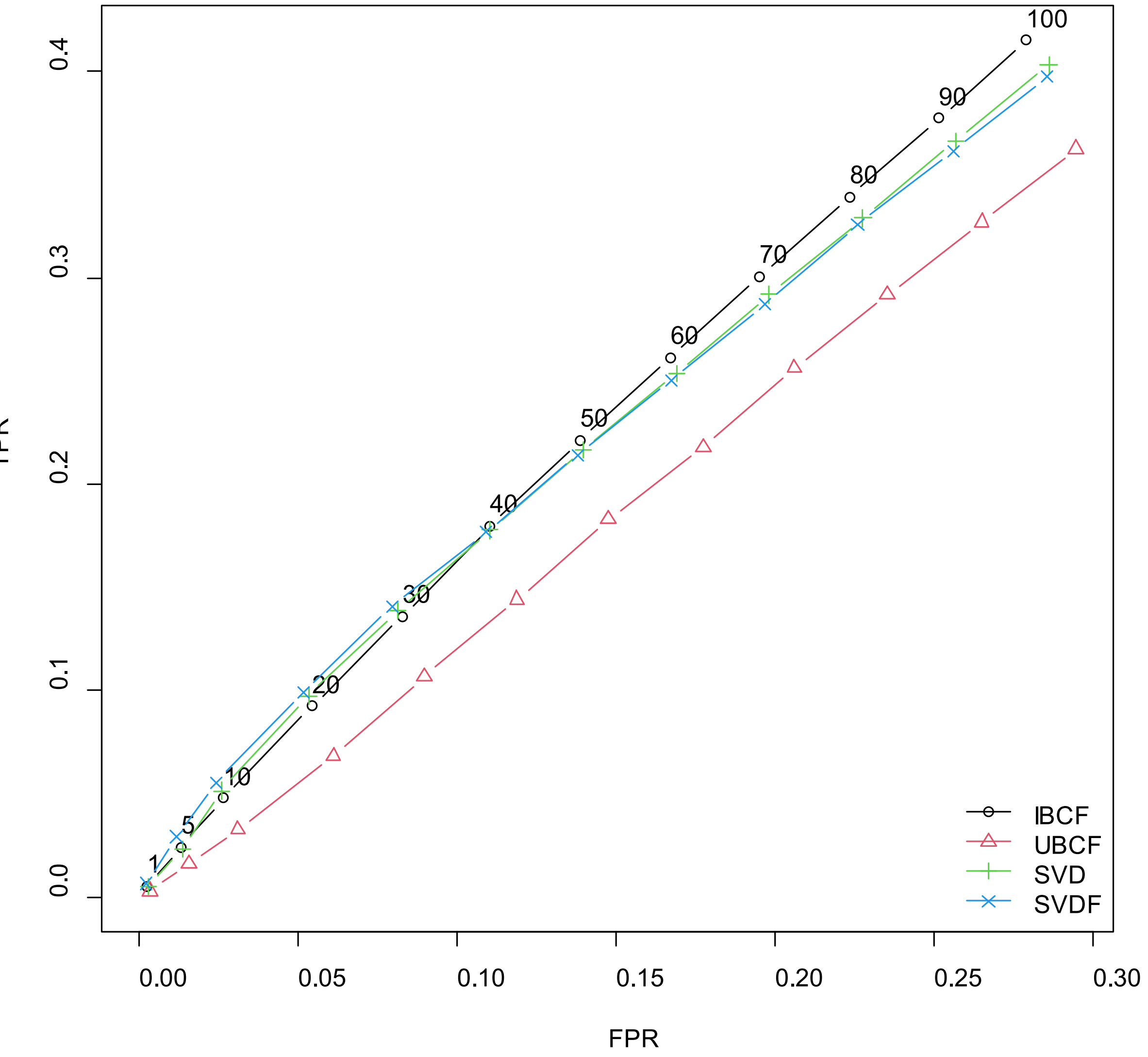
When all u's are fixed then the system recomputes the v's by solving a least-squares problem and vice versa

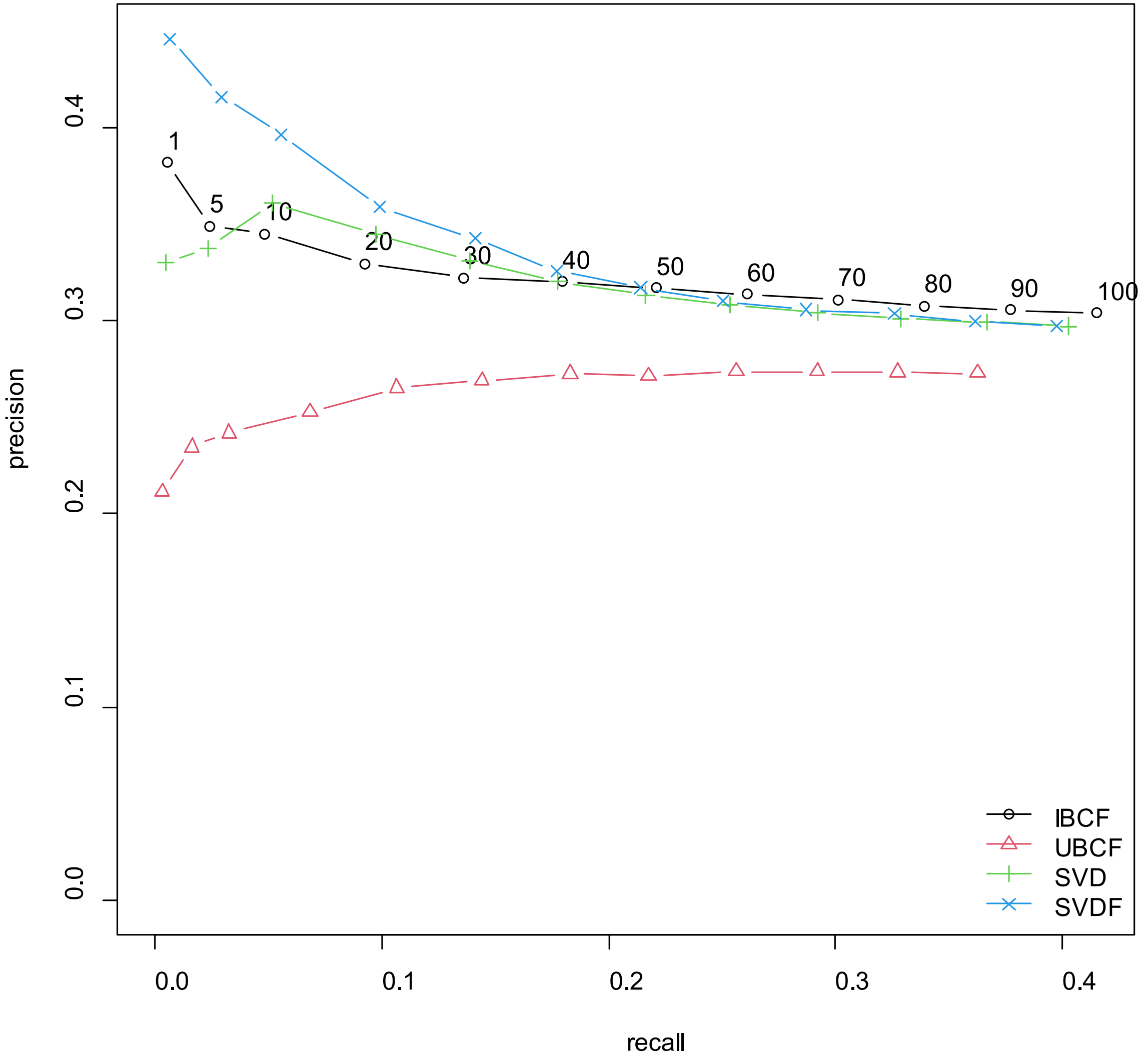
Stochastic gradient descent is easier and faster than Alternating least squares

```r
library(recommenderlab)
library(tidyverse)


data(MovieLense)
class(MovieLense)
help(MovieLense)
dim(MovieLense)



#select only the users who have rated at least 50 movies or movies that had been rated more than 100 times
(ratings_movies <- MovieLense[rowCounts(MovieLense) > 50,
                              colCounts(MovieLense) > 100])



# use the minimum number of items purchased by any user to decide item number to keep
(min(rowCounts(ratings_movies)))

n_fold <- 4
items_to_keep <- 15
rating_threshold <- 3
```

INCLUSION OF SVD ,SVD-FUNK IN COMPARISON, TOP-N

```r
# Use k-fold to validate models
set.seed(1234)
eval_sets <- evaluationScheme(data = ratings_movies, method = "cross-validation",k = n_fold, given = it
                              goodRating = rating_threshold)


models  <- list(

  IBCF=list(name="IBCF",param=list(method = "cosine")),
  UBCF=list(name="UBCF", param=list(method = "pearson")),
  SVD = list(name="SVD", param=list(k = 50)),
  SVDF=list(name="SVDF", param=list(k=50))
)

# varying the number of items we want to recommend to users
n_rec <- c(1, 5, seq(10, 100, 10))

# evaluating the recommendations
results <- evaluate(x = eval_sets, method = models, n= n_rec)

# extract the related average confusion matrices
(avg_matrices <- lapply(results, avg))

plot(results, annotate=TRUE)
plot(results, "prec/rec", annotate = TRUE, main = "Precision-Recall")
```


TPR
FPR
0.0
0.1
0.2
0.3
0.4
0.00
0.05
0.10
0.15
0.20
0.25
0.30
1
5
10
20
30
40
50
60
70
80
90
100
IBCF
UBCF
SVD
SVDF


precision
recall
0.0
0.1
0.2
0.3
0.4
1
5
10
20
30
40
50
60
70
80
90
100
IBCF
UBCF
SVD
SVDF

```r
recommender_ibcf <- Recommender(data = getData(eval_sets, "train"),
                                method = "IBCF",parameter = list(method = "cosine"))

recommender_ubcf <- Recommender(data = getData(eval_sets, "train"),
                                method = "UBCF",parameter = list(method = "pearson"))

recommender_svd <- Recommender(data = getData(eval_sets, "train"),
                               method = "SVD",parameter = list(k=50))

recommender_svdf <- Recommender(data = getData(eval_sets, "train"),
                               method = "SVDF",parameter = list(k=50))

items_to_recommend <- 10


eval_prediction_ibcf <- predict(object = recommender_ibcf, newdata = getData(eval_sets, "known"), n = items_to_recommend, type = "ratings")

eval_prediction_ubcf <- predict(object = recommender_ubcf, newdata = getData(eval_sets, "known"), n = items_to_recommend, type = "ratings")

eval_prediction_svd <- predict(object = recommender_svd, newdata = getData(eval_sets, "known"), n = items_to_recommend, type = "ratings")

eval_prediction_svdf <- predict(object = recommender_svdf, newdata = getData(eval_sets, "known"), n = items_to_recommend, type = "ratings")
# compare RMSE for different models
```

```r
# compare RMSE for different models
######################RANDOM######################

#UBCF

eval_accuracy_ubcf <- calcPredictionAccuracy(
  x = eval_prediction_ubcf, data = getData(eval_sets, "unknown"), byUser = F)

eval_accuracy_ubcf_user <- calcPredictionAccuracy(
  x = eval_prediction_ubcf, data = getData(eval_sets, "unknown"), byUser = TRUE)


head(eval_accuracy_ubcf_user)



#IBCF
eval_accuracy_ibcf <- calcPredictionAccuracy(
  x = eval_prediction_ibcf, data = getData(eval_sets, "unknown"), byUser = F)

eval_accuracy_ibcf_user <- calcPredictionAccuracy(
  x = eval_prediction_ibcf, data = getData(eval_sets, "unknown"), byUser = TRUE)


head(eval_accuracy_ibcf_user)

#SVD
eval_accuracy_svd <- calcPredictionAccuracy(
  x = eval_prediction_svd, data = getData(eval_sets, "unknown"), byUser = F)

eval_accuracy_svd_user <- calcPredictionAccuracy(
  x = eval_prediction_svd, data = getData(eval_sets, "unknown"), byUser = TRUE)


head(eval_accuracy_svd_user)

#SVDF
eval_accuracy_svdf <- calcPredictionAccuracy(
  x = eval_prediction_svdf, data = getData(eval_sets, "unknown"), byUser = F)

eval_accuracy_svdf_user <- calcPredictionAccuracy(
  x = eval_prediction_svdf, data = getData(eval_sets, "unknown"), byUser = TRUE)


head(eval_accuracy_svdf_user)
```

COMPARING MODELS ON RMSE

```
> head(eval_accuracy_ubcf_user)
        RMSE       MSE       MAE
2  1.0618458 1.1275166 0.6466249
28 1.0242833 1.0491562 0.8731549
43 0.9782619 0.9569963 0.7361582
52 0.7456760 0.5560326 0.6088942
54 1.0503581 1.1032521 0.7970184
57 0.8185478 0.6700205 0.6890288
```

```
> head(eval_accuracy_ibcf_user)
        RMSE       MSE       MAE
2  1.3009030 1.6923486 0.8790406
28 0.9543325 0.9107505 0.7492338
43 0.9768906 0.9543152 0.6827829
52 0.5519665 0.3046670 0.3323901
54 0.9000073 0.8100132 0.7281628
57 1.6507024 2.7248184 1.3532901
```

```
> head(eval_accuracy_svd_user)
        RMSE       MSE       MAE
2  0.9866971 0.9735713 0.5974063
28 0.9711420 0.9431168 0.8514862
43 0.9246202 0.8549226 0.6791641
52 0.6698990 0.4487646 0.5586869
54 0.9896816 0.9794697 0.7038758
57 0.8486196 0.7201552 0.6979219
```

```
> head(eval_accuracy_svdf_user)
        RMSE       MSE       MAE
2  1.2317286 1.5171555 0.8969386
28 0.9098291 0.8277891 0.7894634
43 0.9540314 0.9101758 0.7142944
52 0.6510177 0.4238241 0.5372401
54 1.0015432 1.0030887 0.7025052
57 0.7652586 0.5856207 0.6366053
```

```
> eval_accuracy_ubcf
     RMSE       MSE       MAE
1.0696169 1.1440803 0.8405173
> eval_accuracy_ibcf
     RMSE       MSE       MAE
1.1199620 1.2543149 0.8339064
> eval_accuracy_svd
     RMSE       MSE       MAE
0.9548824 0.9118004 0.7501329
> eval_accuracy_svdf
     RMSE       MSE       MAE
0.9410858 0.8856424 0.7334548
```

CHARACTERISTICS OF A LATENT FACTOR MODEL SOLUTION

Matrix factorization

- Projecting items and users in the same n-dimensional space
- Each rating r(ij) can approximately be expressed as a dot product of the i'th user factor and the j'th item factor ($r_{ij} \approx \overline{u}_i \cdot \overline{v}_j$). The dot product represents the interaction between items and users in concept space

Prediction quality can increase as a consequence of…

- filtering out some "noise" in the data and
- detecting nontrivial correlations in the data

Depends on the right choice of the amount of data reduction

- number of singular values in the SVD approach or number of factors in the SVD-FUNK

LITERATURE

Koren et al(2009) Matrix Factorization Techniques for recommender systems, *Computer,p.42-49*

DEPARTMENT OF ECONOMICS
AND BUSINESS ECONOMICS
AARHUS UNIVERSITY