# Recommender Systems

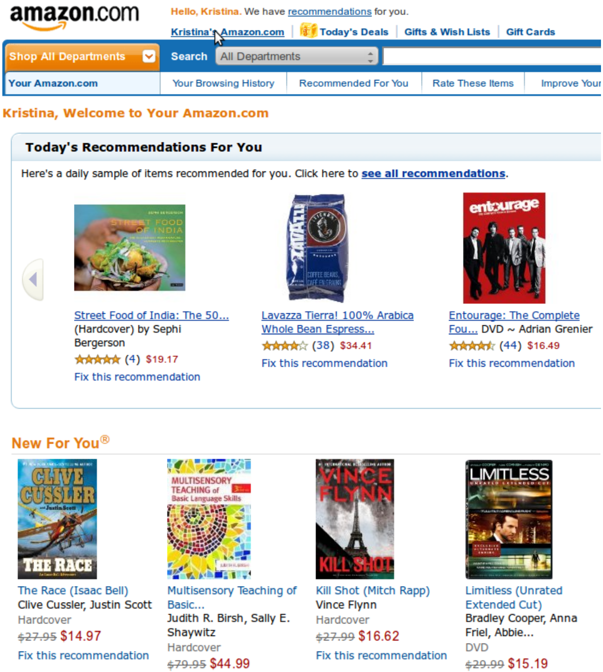
## From Content to Latent Factor Analysis

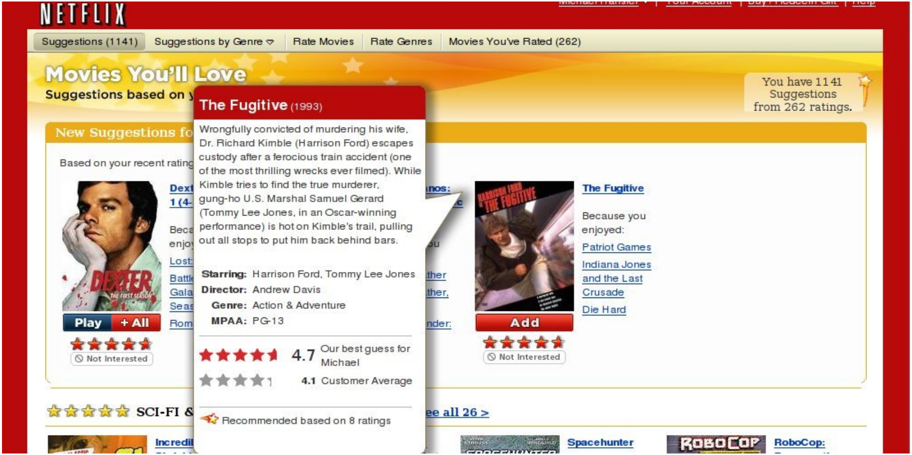
Michael Hahsler

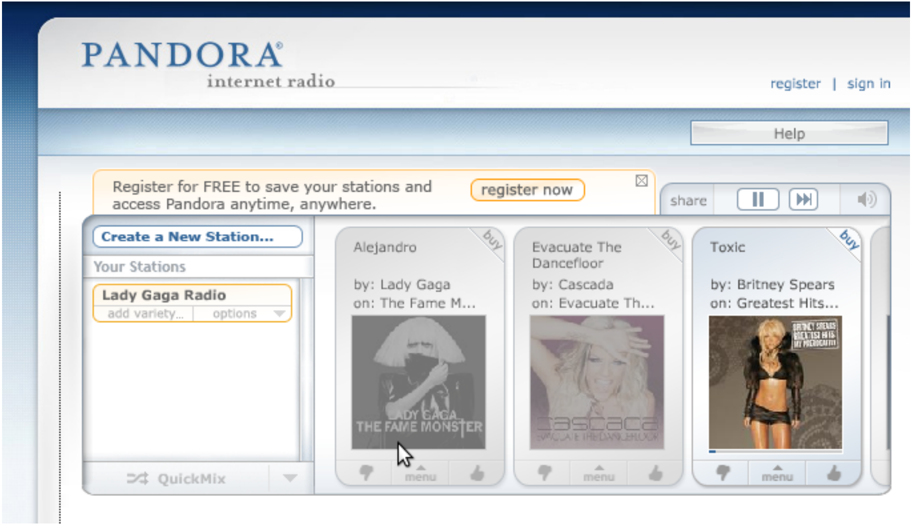
Intelligent Data Analysis Lab (IDA@SMU)
CSE Department, Lyle School of Engineering
Southern Methodist University

CSE Seminar
September 7, 2011

SMU | Bobby B. Lyle School of Engineering


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 Your
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
Play
+ All
Not Interested
The Fugitive
Because you enjoyed:
Patriot Games
Indiana Jones and the Last Crusade
Die Hard
Add
Not Interested
Spacehunter
RoboCop:


PANDORA
internet radio
register | sign in
Help
Register for FREE to save your stations and access Pandora anytime, anywhere.
register now
share
Create a New Station...
Your Stations
Lady Gaga Radio
add variety...
options
QuickMix
Alejandro
by: Lady Gaga
on: The Fame M...
LADY GAGA
THE FAME MONSTER
menu
Evacuate The Dancefloor
by: Cascada
on: Evacuate Th...
menu
Toxic
by: Britney Spears
on: Greatest Hits...
menu
buy

# Table of Contents

# 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 rating/usage/sales data from many users.

(Ansari *et al.*, 2000)

# Table of Contents

# Content-based Approach

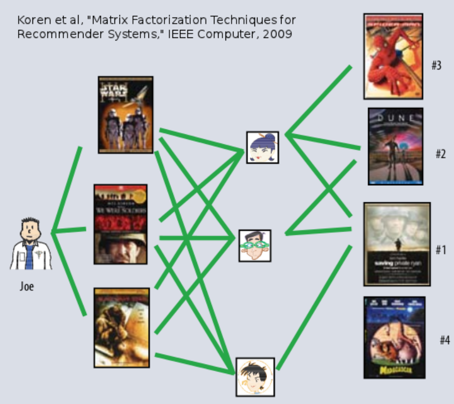

1. Analyze the objects (documents, video, music, etc.) and extract attributes/features (e.g., words, phrases, actors, genre).
2. Recommend objects with similar attributes to an object the user likes.


"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`

# Table of Contents

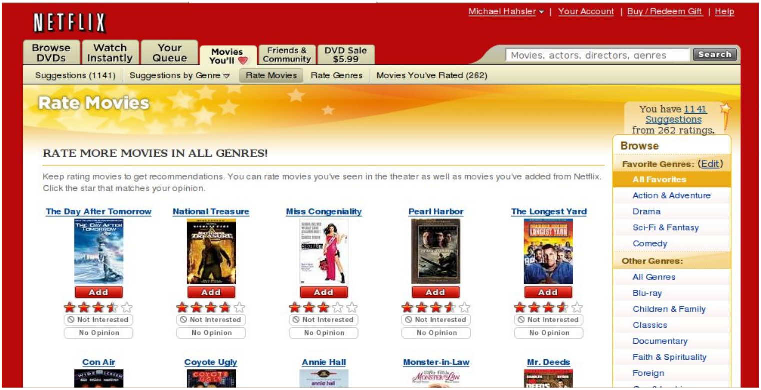
# Collaborative Filtering (CF)




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.

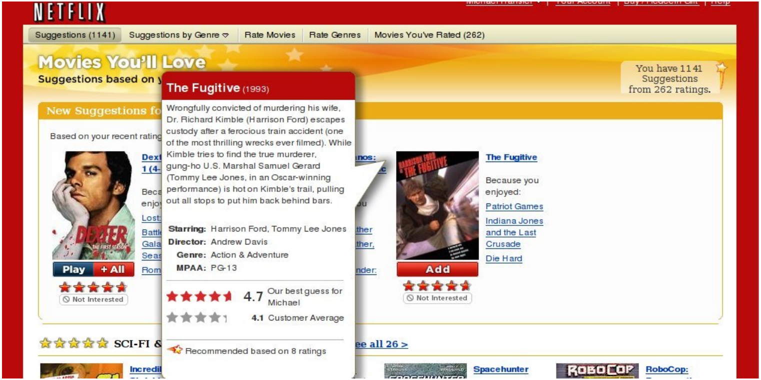
# Data Collection




- 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?

# Output of a Recommender System




- 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)

# 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.

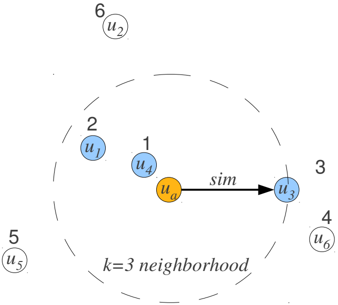
# Table of Contents

# User-based CF

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.

# User-based CF II

- 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

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 II

**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)

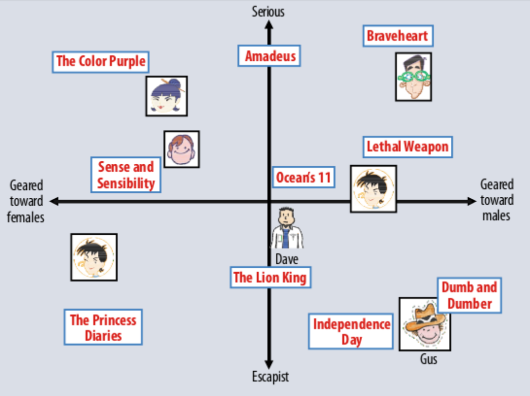
# Table of Contents

# Different Model-based CF Techniques

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.

# Latent Factor Approach




Latent factor approach. Koren et al, "Matrix Factorization Techniques for Recommender Systems," IEEE Computer, 2009

Latent semantic indexing (LSI) developed by the IR community (late 80s) addresses sparsity, scalability and can handle synonyms

⇒ Dimensionality reduction.

# Matrix Factorization

Given a user-item (rating) matrix $M = (r_{ui})$, map users and items on a joint latent factor space of dimensionality $k$.

- Each item $i$ is modeled by a vector $q_i \in \mathbb{R}^k$.
- Each user $u$ is modeled by a vector $p_u \in \mathbb{R}^k$.

such that a value close to the actual rating $r_{ui}$ can be computed. Usually approximated by the dot product of the item and the user vector.

$$r_{ui} \approx \hat{r}_{ui} = q_i^T p_u$$

The hard part is to find a suitable latent factor space.

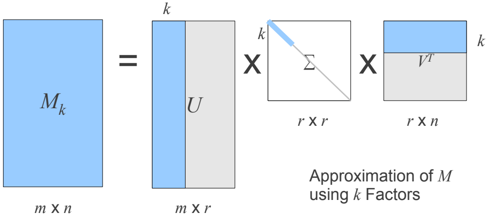
# Singular Value Decomposition (Matrix Factorization)

Linear algebra: Singular Value Decomposition (SVD) to factorize matrix $M$

$$M = U\Sigma V^T$$

$M$ is the $m \times n$ (users $\times$ items) rating matrix of rank $r$.
Columns of $U$ and $V$ are the left and right singular vectors.
Diagonal of $\Sigma$ contains the $r$ singular values.

A low-rank approximation of $M$ using only $k$ factors is straight forward.




The approximation minimizes error $||M - M_k||_F$ (Frobenius norm).

# Challenges (Matrix Factorization)

SVD is $O(m^3)$ and missing values are a problem.

1. Use Incremental SVD to add new users/items without recomputing the whole SVD (Sarwar *et al.*, 2002).
2. To avoid overfitting minimize the regularized square error on only known ratings:

$$\underset{p^*,q^*}{\operatorname{argmin}} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

where $\kappa$ is the $(u,i)$ pairs for which $r$ is known.

Good solutions can be found by stochastic gradient descent or alternating least squares (Koren *et al.*, 2009).

# Prediction (Matrix Factorization)

1. For new user (item) compute $q_i$ ($p_u$).
2. After all $q_i$ and $p_u$ are known, prediction is very fast:

$$\hat{r}_{ui} = q_i^T p_u$$

# Table of Contents

# 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.

# Table of Contents

# Open-Source Implementations

- Apache Mahout: ML library including collaborative filtering (Java)
- C/Matlab Toolkit for Collaborative Filtering (C/Matlab)
- Cofi: Collaborative Filtering Library (Java)
- Crab: Components for recommender systems (Python)
- easyrec: Recommender for Web pages (Java)
- LensKit: CF algorithms from GroupLens Research (Java)
- MyMediaLite: Recommender system algorithms. (C#/Mono)
- RACOFI: A rule-applying collaborative filtering system
- Rating-based item-to-item recommender system (PHP/SQL)
- recommenderlab: Infrastructure to test and develop recommender algorithms (R)

See `http://michael.hahsler.net/research/recommender/` for URLs.

# Table of Contents

# recommenderlab: Reading Data

100k MovieLense ratings data set: The data was collected through the `movielens.umn.edu` from 9/1997 to 4/1998. The data set contains about 100,000 ratings (1-5) from 943 users on 1664 movies.

```
R> library("recommenderlab")
R> data(MovieLense)
R> MovieLense
943 x 1664 rating matrix of class ‘realRatingMatrix’ with
99392 ratings.
R> train <- MovieLense[1:900]
R> u <- MovieLense[901]
R> u
1 x 1664 rating matrix of class ‘realRatingMatrix’ with 124
ratings.
R> as(u, "matrix")[,1:5]
 Toy Story (1995)  GoldenEye (1995) Four Rooms (1995)
                5                NA                NA
Get Shorty (1995)    Copycat (1995)
               NA                NA
```

# recommenderlab: Creating Recommendations

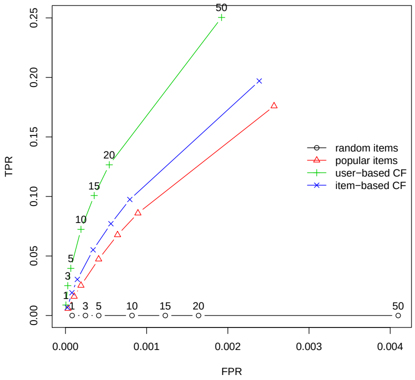
```
R> r <- Recommender(train, method = "UBCF")
R> r
Recommender of type ‘UBCF’ for ‘realRatingMatrix’
learned using 900 users.
R> recom <- predict(r, u, n = 5)
R> recom
Recommendations as ‘topNList’ with n = 5 for 1 users.
R> as(recom, "list")
[[1]]
[1] "Fugitive, The (1993)"
[2] "Shawshank Redemption, The (1994)"
[3] "It’s a Wonderful Life (1946)"
[4] "Princess Bride, The (1987)"
[5] "Alien (1979)"
```

# recommenderlab: Compare Algorithms

```
R> scheme <- evaluationScheme(train, method = "cross", k =
4,
+     given = 10, goodRating=3)
R> algorithms <- list(
+ `random items` = list(name = "RANDOM", param = NULL),
+ `popular items` = list(name = "POPULAR", param = NULL),
+ `user-based CF` = list(name = "UBCF",
+     param = list(method = "Cosine", nn = 50)),
+ `item-based CF` = list(name = "IBCF",
+     param = list(method = "Cosine", k = 50)))
R> results <- evaluate(scheme, algorithms,
+     n = c(1, 3, 5, 10, 15, 20, 50))
```

# recommenderlab: Compare Algorithms II

```
R> plot(results, annotate = c(1, 3), legend = "right")
```




recommenderlab is available at:

http://cran.r-project.org/package=recommenderlab

# References I


Asim Ansari, Skander Essegaier, and Rajeev Kohli. Internet recommendation systems. *Journal of Marketing Research*, 37:363–375, 2000.

John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transations on Information Systems*, 22(1):143–177, 2004.

David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

Brendan Kitts, David Freed, and Martin Vrieze. Cross-sell: a fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 437–446. ACM, 2000.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42:30–37, August 2009.

Andreas Mild and Thomas Reutterer. Collaborative filtering methods for binary market basket data analysis. In *AMT '01: Proceedings of the 6th International Computer Science Conference on Active Media Technology*, pages 302–313, London, UK, 2001. Springer-Verlag.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167. ACM, 2000.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28, 2002.

J. Ben Schafer, Joseph A. Konstan, and John Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1/2):115–153, 2001.

# Thank you!

This presentation can be downloaded from:
`http://michael.hahsler.net/` (under “Publications and talks”)