Matrix Factorisation / Spotify

•••

Simon Kalt & Jannis Fey

Seminar: Music Information Retrieval

Outline

- Recommender Systems
- A Basic Matrix Factorization Model
- Spotify
- Improvements for the Matrix Factorization Model
- Netflix Prize Competition

Recommender Systems

Content Filtering

- Create a profile for each user and a representation for each product
- Match profiles of users with products
- Requires external information → needs to be collected
- Used for Pandora “Music Genome Project”

Collaborative Filtering

- Generate recommendations based on ratings or usage
- No external information necessary
- Relationships between users
- Dependencies between products

→ Associate users with new products

- Problem: Cold Start

Explicit vs. Implicit Feedback

Movies

	?	3	5	?	
	1	?	?	1	
Users	2	?	3	2	
	?	?	?	5	
	5	2	?	4	Chris

Inception

Songs

	1	1	0	0
	0	1	1	1
Users	0	1	0	1
	1	0	1	1
	0	0	1	0

- explicit feedback
  - ○ explicit user input

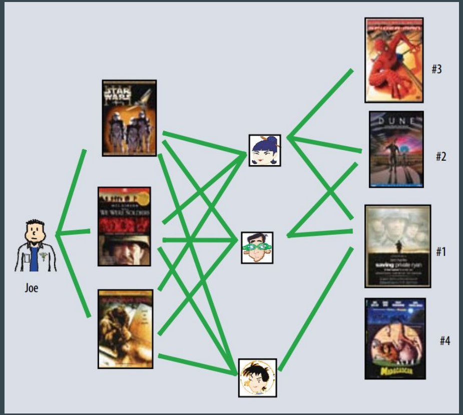
Netflix: 1 – 5 Stars

- implicit feedback
  - ○ observing user behavior

Spotify: 1 if streamed, 0 if not

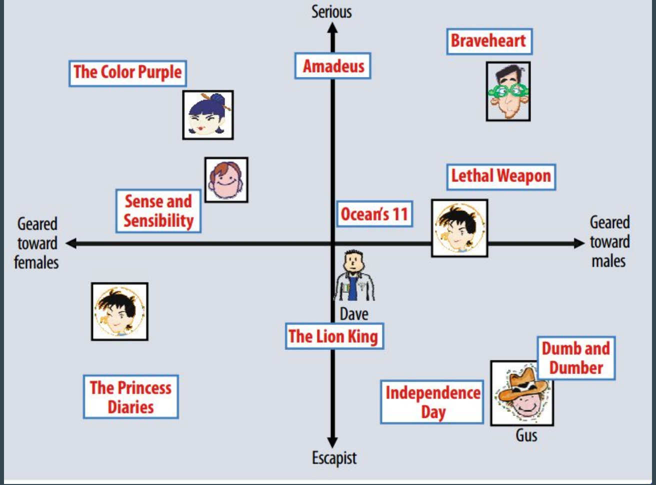
Neighborhood Models

- Relationships between users with similar tastes
- Example:
  - User likes a movie
  - Find users who liked the same movie
  - Find movies a lot of them liked
  - Recommend the movie that has the most “likes”



Latent Factor Models

- Score users and movies in certain “factors”
- Factors measure dimensions like “comedy” or “action”
- Users: how much they like a movie that scores high in this factor



Matrix Factorization

n by m Matrix

$$R = \begin{array}{cccc} R_1 & R_2 & R_3 & R_4 \\ \hline 1 & 2 & 3 & 5 \\ 2 & 4 & 8 & 12 \\ 3 & 6 & 7 & 13 \end{array}$$

→

$$R_1 = 1{*}R_1 + 0{*}R_3$$
$$R_2 = 2{*}R_1 + 0{*}R_3$$
$$R_3 = 0{*}R_1 + 1{*}R_3$$
$$R_4 = 2{*}R_1 + 1{*}R_3$$

n by r

$$P = \begin{array}{cc} R_1 & R_3 \\ \hline 1 & 3 \\ 2 & 8 \\ 3 & 7 \end{array}$$

r by m

$$Q = \begin{array}{cccc} 1 & 2 & 0 & 2 \\ 0 & 0 & 1 & 1 \end{array}$$

→

$$P{*}Q = R$$

A Basic Matrix Factorization Model

What does Matrix Factorization do?

- Characterizes items and users by vectors of factors
- Matrix with two dimension
  - First representing users
  - Second representing items of interest
- Factorize matrix into two matrices, one for users, one for items
- High correspondence between item and user factors

→ recommendation

Example

$$R = \begin{array}{|c|c|c|c|}\hline 5 & 3 & ? & 1 \\ \hline 4 & ? & ? & 1 \\ \hline 1 & 1 & ? & 5 \\ \hline 1 & ? & ? & 4 \\ \hline ? & 1 & 5 & 4 \\ \hline \end{array}$$

- N = 4 User
- M = 5 Items (e.g. movies)
- K = latent features (e.g. genre)
- ? = unknown value (set to 0)

Task:

- find Matrix P and Q such that $R \approxeq P * Q^T$
- R: N x M matrix
- P: N x K matrix
- Q: K x M matrix

Example

$$R = \begin{array}{ccccc} 5 & 3 & 0 & 1 & \\ 4 & 0 & 0 & 1 & \\ 1 & 1 & 0 & 5 & \\ 1 & 0 & 0 & 4 & \\ 0 & 1 & 5 & 4 & p_u \\ & & & q_i & \end{array}$$

$$r_{ui} = q_i^T p_u$$

- each item i is associated with a vector q_i
- each user u is associated with a vector p_u
- r_{ui} represents user’s overall interest in the item’s characteristics

Example

R =

5	3	0	1
4	0	0	1
1	1	0	5
1	0	0	4
0	1	5	4

Goal:

- approximate the matrix R
- minimize the regularized squared error on known ratings

$$\min_{q^*,p^*} \sum_{(u,i)\in\mathcal{K}} \left(r_{ui} - q_i^T p_u\right)^2 + \lambda\left(||q_i||^2 + ||p_u||^2\right)$$

Example

$$R = \begin{array}{|c|c|c|c|}\hline 5 & 3 & 0 & 1 \\ \hline 4 & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 5 \\ \hline 1 & 0 & \boxed{0} & 4 \\ \hline 0 & 1 & 5 & 4 \\ \hline \end{array} \xrightarrow{\text{5000 steps}} \begin{array}{|c|c|c|c|}\hline 4{,}97 & 2{,}98 & 2{,}18 & 0{,}98 \\ \hline 3{,}97 & 2{,}40 & 1{,}97 & 0{,}99 \\ \hline 1{,}02 & 0{,}93 & 5{,}32 & 4{,}93 \\ \hline 1{,}00 & 0{,}85 & \boxed{4{,}49} & 3{,}93 \\ \hline 1{,}36 & 1{,}07 & 4{,}89 & 4{,}12 \\ \hline \end{array}$$

- minimize squared error iteratively
- approximate R step-by-step

Learning Algorithm

- Alternating least squares (ALS)
- q_i and p_u are unknown
  - can not be solved optimally
- rotate between fixing the q_i ’s and fixing the p_u ’s
  - problem becomes quadratic
  - solving a least-squares problem

- favorable if the system can use parallelization

Spotify

Hadoop at Spotify 2009

2014: 700 Nodes in London data center

Improvements for the Matrix Factorization Model

Adding Biases

- Some users generally rate higher
- Some movies generally receive higher ratings
- Baseline prediction b_{ui} for an unknown rating:

$$b_{ui} = \mu + b_u + b_i$$

Adding Biases

- Learn b_u and b_i by solving the least squares problem

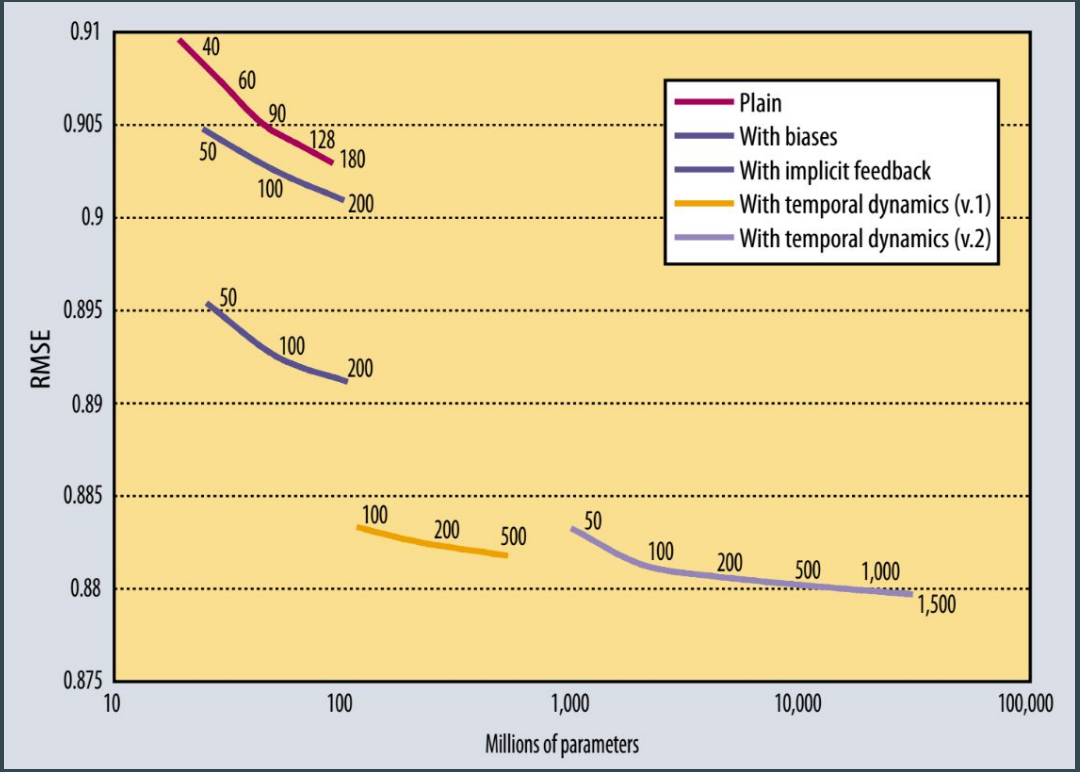
$$\min_{b*,q*,p*} \sum_{(u,i)\in\mathcal{K}} \left(r_{ui} - \mu - b_u - b_i - q_i^T p_u\right)^2$$
$$+ \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2\right)$$

Temporal Dynamics

- Model temporal variation of
  - User preferences: $p_u(t)$
  - Item and user biases: $b_i(t)$, $b_u(t)$
- User's preferences may change
- Movies are more popular at certain times
- User's baseline rating may change
- Time sensitive baseline predictor b_{ui} on a given day t_{ui}

$$b_{ui} = \mu + b_u(t_{ui}) + b_i(t_{ui})$$

Improvements for the Matrix Factorization Model



Netflix Prize Competition

Netflix Prize Competition

- 2006 Netflix announced a contest to improve its recommender system
- Training set: 100 million ratings, 500.000 customers, 17.000 movies

- Teams submit predicted ratings for given test set of 3 million ratings
- Netflix calculates the root-mean-square error (RMSE) on truth ratings

- $1 million for improvement of 10% on Netflix's algorithm
- $50.000 for the first team, if no team reaches 10%

The Winners

- 2007: KorBell
  - RMSE: 0,8723
  - Improvement: 8,42%
- 2008: BellKor in BigChaos
  - RMSE: 0,8624
  - Improvement: 9,27%
- 2009: BellKor's Pragmatic Chaos
  - RMSE: 0,8567
  - Improvement: 10.06%

NETFLIX
2009
DATE: 09.21.09
PAY TO THE ORDER OF: BellKor's Pragmatic Chaos
$ 1,000,000 00
AMOUNT: ONE MILLION 00/100
FOR: The Netflix Prize
Reed Hastings
NETFLIX
NETFLIX

Sources

1. Advances in Collaborative Filtering
    - Yehuda Koren, Robert Bell
2. Matrix Factorization: A Simple Tutorial and Implementation in Python
    - Albert Au Yeung
    - **http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/**
3. Collaborative Filtering with Spark
    - Christopher Johnson (Spotify)
    - https://www.youtube.com/watch?v=3LBgiFch4_g