

Recommender systems

Victor Kitov

v.v.kitov@yandex.ru

Table of Contents

- 1 Introduction
- 2 Core concepts
- 3 Baseline prediction
- 4 User-user collaborative filtering
- 5 Item-item collaborative filtering
- 6 Matrix factorization methods

Introduction

- Everyday people have to make decisions:
 - what music to listen
 - what movie to see
 - what book to read
 - etc.
- Multitude of choices
 - Netflix: >17000 movies
 - Amazon Kindle store: >400000 books
 - etc
- To narrow down the search people rely on recommendations
 - of their friends
 - of newspaper reviews
- These recommendations are too narrow and standard
 - rely on information of a handful of people
 - give general recommendations not tuned for individual user tastes

Examples of recommender systems

cite	recommends what
youtube.com	video
last.fm	music
amazon	goods
facebook	groups, friends
LinkedIn	groups, friends

Information available

We may have information about:

- user: sex, age, education, tastes, activity logs, social graph between users.
- item: description
- rating matrix $R \in \mathbb{R}^{\text{users} \times \text{items}}$ with user-item interactions
 - R is very big
 - R is extremely sparse, because each user interacted only with several items.

Types of the rating matrix

- binary
 - has purchased / didn't purchase particular good
 - liked a post in social network
 - visited particular site
- ternary
 - like, dislike, no opinion
- integer
 - 0,1,2,3,4,5 stars, no opinion
- real
 - time spent on a particular cite
 - money spent for particular service
 - amount of information downloaded from particular resource

Examples of the rating matrix

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U ₁	1			5		2
U ₂		5			4	
U ₃	5	3		1		
U ₄			3			4
U ₅				3	5	
U ₆	5		4			

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U ₁	1			1		1
U ₂		1			1	
U ₃	1	1		1		
U ₄			1			1
U ₅				1	1	
U ₆	1		1			

Task of recommendation system

- Task of recommender system - fill unknown values of the rating matrix
 - in real business we need to present these recommendations:
 - there should be few of them
 - they should have high accuracy and high informativeness (augment bread with butter-not interesting!)
 - diversity of recommendations also matters (augment different types of the same-not interesting!)

Approaches to recommendation

- content-based
 - match user description and item description
- collaborative filtering
 - use only known values of R to fill missing values
- combined
 - ensembling is always good!

Content-based recommendation

- Text similarity approach:
 - If user tastes description is available:
 - match tastes description and item description
 - typically cosine similarity is used
 - If user tastes not available - generate them by concatenation of descriptions of items, he watched/liked/bought.
- General ML approach: learn prediction function

$$F : (\text{user features, item features}) \rightarrow \textit{rating}$$


























Collaborative filtering

Collaborative filtering

A method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating).

- General CF algorithm:
 - 1 Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
 - 2 Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user
- Main assumption of collaborative filtering: if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue.

Collaborative filtering

- This is more general than giving average (not-specific for user) score for each item.
- Recommender systems, describe any algorithm of predicting scores for items.
 - it may use information from other users as CF
 - it may use description of items
 - it may use item preferences of user
 - user query information

Netflix prize

- Netflix - an online DVD-rental and online video streaming service
- Competition for the best collaborative filtering algorithm to predict user ratings for films
- October 2006 - September 2009
- Prize fund: 1 000 000 \$
- 480,189 users
- 17,770 movies.
- ratings have format: <user, movie, date of grade, grade>.
- Grades: 1,2,3,4,5.

Challenges in collaborative filtering

- Scalability
 - many users and items
- Gray sheep
 - some people may not match any of the group of existing users
- Cold start:
 - new user has not rated many items => hard to predict his preferences
 - new item wasn't rated by many users => hard to predict its characteristics
- Shilling attacks
 - shop representatives may give lots of positive ratings for their own items and negative ratings for their competitors.
- Low diversity
 - only highly rated items are promoted and only these items become more popular

Table of Contents

- 1 Introduction
- 2 Core concepts**
- 3 Baseline prediction
- 4 User-user collaborative filtering
- 5 Item-item collaborative filtering
- 6 Matrix factorization methods

Core concepts

- **Users** give **ratings** to **items**
- **Rating matrix R**: (user, item) \rightarrow rating

	<i>Batman Begins</i>	<i>Alice in Wonderland</i>	<i>Dumb and Dumber</i>	<i>Equilibrium</i>
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?

- **Tasks of CF**:
 - Fill missing values of the ratings matrix
 - Recommend items to user
 - may not be the highest ranked, if there are more considerations: use recommendations of different models, make recommendations more diverse, use description information, use current need of user.

Notation

- U - set of all users
- I - set of all items
- u - some user
- i - some item
- I_u - set of all items, rated by user u
- U_i - set of all users, that rated item i .
- $R = \{r_{u,i}\}_{i \in I, u \in U}$ - rating matrix
- $r_{u,i}$ - rating of item i by user u
- $\hat{r}_{u,i}$ - predicted rating of item i by user u

Table of Contents

- 1 Introduction
- 2 Core concepts
- 3 Baseline prediction**
- 4 User-user collaborative filtering
- 5 Item-item collaborative filtering
- 6 Matrix factorization methods

Baseline prediction

- Simple baselines:

- $\hat{p}_{u,i} = \mu$ ($\mu = \frac{1}{n} \sum_{u,i} r_{u,i}$, $n = |\{(u, i) : r_{u,i} \text{ is specified}\}|$)
- $\hat{p}_{u,i} = \bar{r}_u = \frac{1}{|I_u|} \sum_{i \in I_u} r_{u,i}$
- $\hat{p}_{u,i} = \bar{r}_i = \frac{1}{|U_i|} \sum_{u \in U_i} r_{u,i}$

General baseline

- General baseline

$$\hat{p}_{u,i} = \mu + b_u + b_i$$
$$b_u = \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u,i} - \mu)$$
$$b_i = \frac{1}{|U_i|} \sum_{u' \in U_i} (r_{u',i} - b_{u'} - \mu)$$

- intuition:
 - b_u is how much higher user rates items than average
 - b_i is how much item i is rated higher than average user rating.

General baseline with damping

- General baseline

$$\hat{p}_{u,i} = \mu + b_u + b_i$$

$$b_u = \frac{1}{|I_u| + \alpha} \sum_{i \in I_u} (r_{u,i} - \mu)$$

$$b_i = \frac{1}{|U_i| + \beta} \sum_{u' \in U_i} (r_{u',i} - b_{u'} - \mu)$$

- $\alpha > 0, \beta > 0$ - damping terms, $\alpha = \beta \approx 25$ is advised.
- Intuition:
 - when $|I_u|$ is small information from user rating is sparse and unreliable
 - when $|U_i|$ is small information from item rating is sparse and unreliable
 - α, β penalize sparse information about I_u, U_i
 - resulting forecast is closer to the mean

Motivation for baseline

Motivation:

- compare accuracy with advanced model
- can impute missing values with baseline values
- normalization: predict $r_{u,j} - b_{u,j}$ instead of $r_{u,j}$
 - define $B = \{b\}_{u \in U, i \in I}$
 - define $R' = R - \hat{R}$ (normalized ratings)
 - when predict R' instead of R model can concentrate on more subtle dependencies

Table of Contents

- 1 Introduction
- 2 Core concepts
- 3 Baseline prediction
- 4 User-user collaborative filtering**
 - User similarity measures
- 5 Item-item collaborative filtering
- 6 Matrix factorization methods

User-User CF

- Known also as K-NN CF
- Was the first automated CF

Algorithm

- Define similarity function on users: $s : U \times U \rightarrow \mathbb{R}$.
- Algorithm (with level normalization):
 - 1 For user u using similarity s find a neighbourhood of similar users $N \subseteq U$.
 - 2 Generate rating predictions by averaging over N :

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u, u') (r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u, u')|}$$

Comments:

- \bar{r}_u compensates for average rating behavior of u .
 - “skeptics/optimists normalization”
- when $s(u, u') < 0$ averaging will target deviation with opposite sign.

Algorithm

- Algorithm with level and scale normalization:
 - 1 For user u using similarity s find a neighborhood of similar users $N \subseteq U$.
 - 2 Generate rating predictions by averaging over N :

$$p_{u,i} = \bar{r}_u + \sigma_u \frac{\sum_{u' \in N} s(u, u') (r_{u',i} - \bar{r}_{u'}) / \sigma_{u'}}{\sum_{u' \in N} |s(u, u')|}$$

where σ_u is standard deviation of user u ratings.

Number of nearest neighbours

- Depends on application, in general 20 – 50 recommended.
- Possible approaches:
 - use all: $N = U \setminus \{u\}$
 - use K best
 - use $\{u' : s(u', u) \geq \text{threshold}\}$

- 4 User-user collaborative filtering
 - User similarity measures

Pearson correlation

$$s(u, v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}}$$

- captures linear dependency
- tends to give high similarity for users with few ratings
 - solution: may use $s'(u, v) = s(u, v) \min\{|I_u \cap I_v|/50, 1\}$

Constrained Pearson correlation

$$s(u, v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - r_{neutral})(r_{v,i} - r_{neutral})}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - r_{neutral})^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - r_{neutral})^2}}$$

- $r_{neutral}$ -neutral level of rating
 - 0-bad,1-neutral,2-good=> $r_{neutral} = 1$.
- captures linear dependency
- tends to give high similarity for users with few ratings
 - solution: may use $s'(u, v) = s(u, v) \min\{|I_u \cap I_v|/50, 1\}$

Other possible measures

- Spearman correlation
 - correlation between ranks of $r_{u,i}$, $i \in I_u$ and ranks of $r_{v,i}$, $i \in I_v$
 - captures the level of monotone dependencies
- Cosine similarity

$$s(u, v) = \frac{\sum_{i \in I_u \cap I_v} r_{u,i} r_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v} r_{u,i}^2} \sqrt{\sum_{i \in I_u \cap I_v} r_{v,i}^2}}$$

Table of Contents

- 1 Introduction
- 2 Core concepts
- 3 Baseline prediction
- 4 User-user collaborative filtering
- 5 Item-item collaborative filtering**
- 6 Matrix factorization methods

Item-item algorithm

- 1 For item i find a set of items similar to i and rated by u :
 $S \subseteq I_u$.
- 2 Predict $r_{u,i}$ by averaging over S :

$$\hat{p}_{u,i} = \frac{\sum_{i' \in S} s(i, i') r_{u,i'}}{\sum_{i' \in S} |s(i, i')|} \quad (1)$$

Comments: when $s(i, i')$ may be < 0 to avoid negative ratings use

$$\hat{p}_{u,i} = \text{baseline}_{u,i} + \frac{\sum_{i' \in S} s(i, i') (r_{u,i'} - \text{baseline}_{u,i})}{\sum_{i' \in S} |s(i, i')|}$$

How to generate recommendations for purchased/not purchased data, so that $r_{u,i} \in \{0, 1\}$?

Benefits of item-item algorithm

- We need to generate recommendations for user u in real-time when user is active
- When user is active, his ratings change.
 - User-user CF: after changes in rating we need to recalculate all nearest neighbours of user - long!
 - Item-item CF: for each pair of items we precalculate $s(i, i')$ and find nearest neighbours of each item. Change in user ratings affects only the weights in the summation of $s(i, i')$ in (1).
- Due to possible precomputations, item-item becomes more efficient.
 - improvement became possible because
 - user profile is dynamic
 - item profile is approximately static

Item similarities

- Cosine similarity:

$$s(i, j) = \frac{\langle r_i, r_j \rangle}{\|r_i\| \|r_j\|} = \frac{\sum_{u \in U} r_{u,i} r_{u,j}}{\sqrt{\sum_{u \in U} r_{u,i}^2} \sqrt{\sum_{u \in U} r_{u,j}^2}}$$

- For binary ratings: conditional probability (similar to MI):

$$s(i, j) = P(j = 1 | i = 1) = \frac{\text{Freq}(i \& j)}{\text{Freq}(i)}$$

To compensate for frequently occurring j :

$$s(i, j) = \frac{\text{Freq}(i \& j)}{\text{Freq}(i) (\text{Freq}(j))^\alpha}$$

- Pearson correlation¹.

¹can be used but showed worse accuracy than cosine similarity

Practical considerations

- When ratings matrix not normalized we can get spurious correlations due to
 - user bias
 - item bias
- To avoid this, we should calculate $s(i, i')$ on normalized matrix $R' = \{r_{u,i} - \mu - b_u - b_i\}_{u,i}$.
- Useful heuristic: after $\hat{r}_u \leftarrow \hat{r}_u / \|\hat{r}_u\|$ users, having rated few items will impact items similarity more.
- Precompute $s(i, i')$ and k most similar items for each item.

Table of Contents

- 1 Introduction
- 2 Core concepts
- 3 Baseline prediction
- 4 User-user collaborative filtering
- 5 Item-item collaborative filtering
- 6 Matrix factorization methods**

SVD decomposition

Each matrix $R \in \mathbb{R}^{M \times N}$ may be decomposed as $R = U \Sigma V^T$, where

- $U \in \mathbb{R}^{M \times P}$, $\Sigma \in \mathbb{R}^{P \times P}$, $V \in \mathbb{R}^{N \times P}$, $P = \text{rank } R$
- $U^T U = V^T V = I$, $\Sigma = \text{diag}\{\sigma_1, \sigma_2, \dots, \sigma_P\}$, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_P \geq 0$

Truncated SVD with rank K :

- $\hat{R} = U_K \Sigma_K V_K^T$, where
 - U_K contains first K columns of U
 - $\Sigma_K = \text{diag}\{\sigma_1, \sigma_2, \dots, \sigma_K\}$
 - V_K contains first K columns of V

It satisfies the property:

$$\hat{R} = \arg \min_{A \in \mathbb{R}^{M \times N}, \text{rank } A \leq K} \|R - A\|_{\text{Frobenius}}$$

SVD decomposition

- Intuition:
 - row u of U : user's interest in each of K topics
 - column k of V : relevances of each item to topic k
 - k -th diagonal element of Σ : influence of topic k on users-items relationship

Recalculation for new user

- “Folding-in” process: for new user u calculate low-dimensional representation of tastes p_u , given his high-dimensional rating vector r_u :

$$\begin{aligned} p_u &= \arg \min_p \|r_u - Vp\|^2 = \{\text{OLS solution}\} = (V^T V)^{-1} V^T r_u = \\ &= \{\text{orthonormality of } V\} = V^T r_u \end{aligned}$$

- $p_u = V^T r_u$ is a vector of scalar products $[\langle v_1, r_u \rangle, \dots, \langle v_K, r_u \rangle]^T$
- Backward transformation from low-dimensional representation p_u to high dimensional representation \hat{r}_u :
- $\hat{r}_u = Vp_u$
- Recommendation step: predictions of \hat{r}_u can be used as predictions for unrated items for user u .

SVD decomposition

- To compute SVD we need to fill missing values
 - with b_u , b_i or $b_{u,i}$
- We may apply SVD to normalized $R' = R - \hat{R}$.
 - impute missing values with 0 (not true approximation!)
- **Main disadvantage of SVD: it tries to fit all values equally, both specified and unspecified.**
- We need to estimate matrix decomposition only using specified ratings

Sparse SVD

Usual SVD may be written as (because it optimizes Frobenius norm):

$$\|R - PQ\|_{Frobenius}^2 \rightarrow \min_{P,Q}$$

$P = [p_1, \dots, p_{|U|}]^T \in \mathbb{R}^{|U| \times K}$, $Q = [q_1, \dots, q_{|I|}] \in \mathbb{R}^{K \times |I|}$, prediction $\hat{R} = PQ$.

Define $D = \{(u, i) : r_{ui} \text{ is specified}\}$

Approximate: $\sum_{(u,i) \in D} \left(\underbrace{r_{ui} - b_{ui} - p_u^T q_i}_{\varepsilon_{ui}} \right)^2 \rightarrow \min_{P,Q}$

Repeat: for random $(u, i) \in D$ $\varepsilon_{ui}^2 \rightarrow \min_{p_u, q_i}$:

Sparse SVD

Usual SVD may be written as (because it optimizes Frobenius norm):

$$\|R - PQ\|_{Frobenius}^2 \rightarrow \min_{P,Q}$$

$P = [p_1, \dots, p_{|U|}]^T \in \mathbb{R}^{|U| \times K}$, $Q = [q_1, \dots, q_{|I|}] \in \mathbb{R}^{K \times |I|}$, prediction $\hat{R} = PQ$.

Define $D = \{(u, i) : r_{ui} \text{ is specified}\}$

Approximate: $\sum_{(u,i) \in D} \left(\underbrace{r_{ui} - b_{ui} - p_u^T q_i}_{\varepsilon_{ui}} \right)^2 \rightarrow \min_{P,Q}$

Repeat: for random $(u, i) \in D$ $\varepsilon_{ui}^2 \rightarrow \min_{p_u, q_i}$:

$$\begin{cases} p_u \leftarrow p_u + \eta \varepsilon_{ui} q_i & k \in \{1, 2, \dots, K\} \\ q_i \leftarrow q_i + \eta \varepsilon_{ui} p_u & k \in \{1, 2, \dots, K\} \end{cases}$$

Advantages of sparse SVD

- Uses only defined ratings r_{ui} .
 - no bias for describing baseline imputation.
- Easy to add regularization:

$$\varepsilon_{ui}^2 + \lambda \|p_u\|^2 + \mu \|q_i\|^2 \rightarrow \min_{P,Q}$$

- *How will formula change with L_2 regularization?*
- Easy to add non-negativity constraints (with gradient projection method):

$$p_{tu} \geq 0, q_{ti} \geq 0$$

- Straightforward to recalculate factorization:
 - for new user u
 - for new item i
 - for new rating r_{ui}
- Fast on big data