Knowledge Discovery and Data Mining 1 (VO) (707.003)

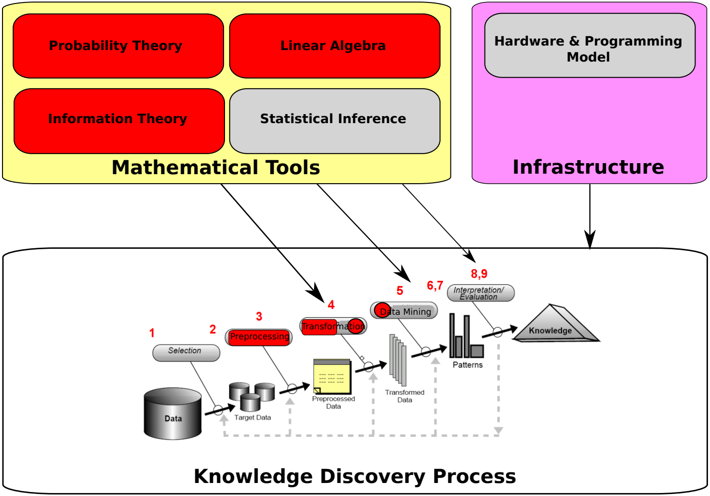
Recommender Systems & Matrix Factorization

Denis Helic

KTI, TU Graz

Nov 27, 2014

Big picture: KDDM



Outline

1. Introduction
2. Content-Based Recommendations
3. Collaborative Filtering
4. Factor Analysis: UV Decomposition

Slides

Slides are partially based on "Mining Massive Datasets" Chapter 9

Recap

Recap

Review of recommender systems

Recap – Recommender systems

- Recommender systems predict user responses to options
- E.g. recommend news articles based on prediction of user interests
- E.g. recommend products based on the predictions of what user might like
- Recommender systems are necessary whenever users interact with huge catalogs of items
- E.g. thousands, even millions of products, movies, music, and so on.

Recap – Formal model: the utility matrix

- In a recommender system there are two classes of entities: users and items

Utility Function

Let us denote the set of all users with U and the set of all items with I.
We define the utility function $u : UxI \to R$, where R is a set of ratings and is a totally ordered set.
For example, $R = \{1, 2, 3, 4, 5\}$ set of star ratings, or $R = [0, 1]$ set of real numbers from that interval.

Recap – The Utility Matrix

- The utility function maps pairs of users and items to numbers
- These numbers can be represented by a utility matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$, where n is the number of users and m the number of items
- The matrix gives a value for each user-item pair where we know about the preference of that user for that item
- E.g. the values can come from an ordered set (1 to 5) and represent a rating that a user gave for an item

Recap – The Utility Matrix

- We assume that the matrix is sparse
- This means that most entries are unknown
- The majority of the user preferences for specific items is unknown
- An unknown rating means that we do not have explicit information
- It does not mean that the rating is low
- Formally: the goal of a recommender system is to predict the blank in the utility matrix

Recap – The Utility Matrix: example

Movie / User	HP1	HP2	HP3	Hobbit	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Recap – Recommender systems: short summary

- Three key problems in the recommender systems:
- Collecting data for the utility matrix
  1. Explicit by e.g. collecting ratings
  2. Implicit by e.g. interactions (users buy a product implies a high rating)
- Predicting missing values in the utility matrix
  1. Problems are sparsity, cold start, and so on.
  2. Content-based, collaborative filtering, matrix factorization
- Evaluating predictions
  1. Training dataset, test dataset, error

Item profiles

- The utility matrix itself does not offer a lot of evidence
- Typically in practice, the utility matrix is a very sparse matrix
- Also, we might think about the utility matrix as a final result of a rating process
- For example, items have some (general) characteristics that users like/dislike and because of these characteristics they rate them in one way or another
- From these characteristics we build **item profiles**
- Thus, an item profile is a **description of an item**

Item profiles: example

- Movies have certain features which describe important characteristics of every movie
- Set of actors
- Director
- Year
- Genre
- Technical characteristics

Item profiles: example

- News articles have different characteristics
- Topic
- Writer
- Length
- Words

Content based recommendations

- Basic idea is to build user profiles from the item profiles that a user liked
- Then we match the user profile against the item catalog and recommend the most similar items to the user profile
- For example, suppose we have the following descriptions of the movies
- We have a set of actors: Julia Roberts, Edward Norton, Martin Scorsese, Ridley Scott, ...
- We have a set of genres: Western, Drama, Thriller, ...

Content based recommendations

- Now we have a user who highly rated two drama movies, both with Julia Roberts
- We will build a user profile from those two movies
- This user profile will have Julia Roberts and Drama as the description
- We will match that user profile with the catalog (calculate the similarity of the user profile with all remaining item profiles)
- We recommend the most similar items to the user: further drama movies with Julia Roberts

Representing profiles

- Let us represent item and user profiles as vectors with features as dimensions
- Let us denote the features with a feature vector $\mathbf{x}$, where each element (dimension) corresponds to a feature
- $x_1 = JuliaRoberts$, $x_2 = EdwardNorton$, ..., $x_7 = Thriller$
- Now we might represent the users and items with vectors $\mathbf{u} \in \mathbb{R}^d$ and $\mathbf{v} \in \mathbb{R}^d$ with corresponding values for each feature

Representing item profiles: example

- E.g. we might represent a movie starring Julia Roberts directed by Martin Scorsese and with a mixture elements of drama and thriller as a vector:

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0.5 \\ 0.5 \end{pmatrix}$$

Representing item profiles: example

- Another thriller movie starring Edward Norton and directed by Ridley Scott:

$$\mathbf{v}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Representing item profiles: example

- A drama movie starring Julia Roberts and directed by Ridley Scott:

$$\mathbf{v}_3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Representing user profiles

- Now, suppose we have a user $\mathbf{u}_i$ who rated movies $\mathbf{v}_1, \ldots, \mathbf{v}_n$
- The simplest way to build a user profile is to calculate the weighted average of the rated item profiles
- Let m_{ij} be the rating of the i-th user for the j-th movie, and n_i is the total number of movies that the user i has rated
- Then the i-th user profile is:

$$\mathbf{u}_i = \frac{1}{n_i} \sum_j m_{ij} \mathbf{v}_j$$

Representing user profiles: example

- Suppose we have a user who rated the first movie with 3 stars and the second movie with 5 stars
- E.g. $m_{1,1} = 3$ and $m_{1,2} = 5$
- Then the user profile is:

$$\mathbf{u}_1 = \frac{1}{2}(3\mathbf{v}_1 + 5\mathbf{v}_2)$$

Representing user profiles: example

$$\mathbf{u}_1 = \frac{1}{2}(3 \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0.5 \\ 0.5 \end{pmatrix} + 5 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}) = \begin{pmatrix} 1.5 \\ 2.5 \\ 1.5 \\ 2.5 \\ 0 \\ 0.75 \\ 3.25 \end{pmatrix}$$

Representing user profiles: example

- Suppose we have another user who rated the first movie with 5 stars and the third movie with 1 star
- E.g. $m_{2,1} = 5$ and $m_{2,3} = 1$
- Then the user profile is:

$$\mathbf{u}_2 = \frac{1}{2}(5\mathbf{v}_1 + \mathbf{v}_3)$$

Representing user profiles: example

$$\mathbf{u}_1 = \frac{1}{2}(5 \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0.5 \\ 0.5 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 2.5 \\ 0.5 \\ 0 \\ 1.75 \\ 1.25 \end{pmatrix})$$

Representing user profiles

- Another possibility to build a user profile is to normalize the user ratings
- The idea here is that some users are more generous than the others and give generally higher ratings
- Thus, we want to have a baseline for each user and that is the user average rating
- Another idea is that low ratings mean that the user did not like the movie
- These are in fact negative ratings

Representing user profiles

- E.g. user 1 rated 5,4,2,1,1
- 2 and 1 would mean that the user did not like the movie
- E.g. user 2 rated 5,5,5,3,3
- Most probably 3 would mean that the user did not like the movie
- How to capture this?

Representing user profiles

- E.g. user 1 rated 5,4,2,1,1
- 2 and 1 would mean that the user did not like the movie
- E.g. user 2 rated 5,5,5,3,3
- Most probably 3 would mean that the user did not like the movie
- How to capture this?
- Move to the center!

Representing user profiles: example

- Back to our example
- First user rated the first movie with 3 stars and the second movie with 5 stars
- E.g. $m_{1,1} = 3$ and $m_{1,2} = 5$
- The user average is 4 and we move to the center: $m_{1,1} = -1$ and $m_{1,2} = 1$
- Then the user profile is:

$$\mathbf{u}_1 = \frac{1}{2}(-1\mathbf{v}_1 + \mathbf{v}_2)$$

Representing user profiles: example

$$\mathbf{u}_1 = \frac{1}{2}(-1 \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0.5 \\ 0.5 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}) = \begin{pmatrix} -0.5 \\ 0.5 \\ -0.5 \\ 0.5 \\ 0 \\ -0.25 \\ 0.25 \end{pmatrix}$$

Representing user profiles: example

- Second user rated the first movie with 5 stars and the third movie with 1 star
- E.g. $m_{2,1} = 5$ and $m_{2,3} = 1$
- The user average is 3 and we move to the center: $m_{1,1} = 2$ and $m_{1,2} = -2$
- Then the user profile is:

$$\mathbf{u}_2 = \frac{1}{2}(2\mathbf{v}_1 - 2\mathbf{v}_3)$$

Representing user profiles: example

$$\mathbf{u}_1 = \frac{1}{2}(2\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0.5 \\ 0.5 \end{pmatrix} - 2\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ -0.5 \\ 0.5 \end{pmatrix})$$

Recommending based on content

- We calculate e.g. cosine similarity of the user vector with all other movies from the catalog
- We recommend the top 5, 10, 15 movies to that user
- Advantages: no need for data on other users
- Recommend to users with unique tastes (user-user similarity over ratings is not a problem)
- Able to recommend new items (item-item similarity over ratings is not a problem)
- Explanations for recommendations are possible

Recommending based on content

- Problems: finding features is very hard
- E.g. difficult to categorize movies to genres because genres overlap
- Overspecialization: you never recommend anything outside user profile
- But people might have multiple interests that they did not express yet
- Unable to exploit the quality judgment of other users
- Cold start problem with new users: what is the user profile for a new user?

Collaborative filtering

- The basic idea is to find set of similar users to a given user
- The set of similar users (neighborhood) are all users whose likes and dislikes are similar to the given user
- In other words it is a set of users with similar ratings to a given users
- Predict the ratings for a given user based on the ratings of users from her neighborhood

Collaborative filtering

- The key is to measure similarity between users
- We already have seen some possibilities
- We represent user ratings as vectors
- We may measure cosine similarity
- We may move to the center and measure cosine, i.e. this is then covariance
- We may normalize to obtain correlation, etc.

Collaborative filtering: example

User \ Movie	HP1	HP2	HP3	Hobbit	SW1	SW2	SW3
A	4			5	2		
B	5	5	2				
C				2	4	5	
D		3			1	1	3

Collaborative filtering

- Users A and B are similar
- Users A and C are dissimilar
- Why is that?

Collaborative filtering

- Users A and B are similar
- Users A and C are dissimilar
- Why is that?
- A and B rated only one movie together, but both gave a high rating
- A and C rated two movies together, but with opposing ratings

Collaborative filtering: cosine similarity

- Let $\mathbf{m}_i$ be the vector rating of the i-th user
- We set zeros in blank spaces

$$\mathbf{m}_1 = \begin{pmatrix} 4 \\ 0 \\ 0 \\ 5 \\ 2 \\ 0 \\ 0 \end{pmatrix} \mathbf{m}_2 = \begin{pmatrix} 5 \\ 5 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \mathbf{m}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \\ 4 \\ 5 \\ 0 \end{pmatrix}$$

Collaborative filtering: Cosine similarity

- The cosine similarity:

$$sim(u_i, u_j) = \frac{\mathbf{m}_i^T \mathbf{m}_j}{||\mathbf{m}_i||_2 ||\mathbf{m}_j||_2}$$

$$\begin{aligned} sim(u_1, u_2) &= 0.40572 \\ sim(u_1, u_3) &= 0.4 \end{aligned}$$

Collaborative filtering: Correlation

- This does not capture our intuition that users 1 and 3 are dissimilar and users 1 and 2 are similar
- The problem is that we treated blanks as zeros and therefore as negative ratings
- Maybe we should treat zeros as average ratings
- Let us move to the center and calculate correlations, i.e. centered cosine similarity
- While calculating the averages and centering we will ignore the blank spaces

$$\mathbf{x}_i = \mathbf{m}_i - \overline{m}_i \mathbf{1}$$

Collaborative filtering: Correlation

$$
\begin{aligned}
\overline{m}_1 &= \frac{11}{3} \\
\overline{m}_2 &= \frac{12}{3} = 4 \\
\overline{m}_3 &= \frac{11}{3}
\end{aligned}
$$

$$
\mathbf{x}_1 = \begin{pmatrix} 1/3 \\ 0 \\ 0 \\ 4/3 \\ -5/3 \\ 0 \\ 0 \end{pmatrix}
\mathbf{x}_2 = \begin{pmatrix} 1 \\ 1 \\ -2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\mathbf{x}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -5/3 \\ 1/3 \\ 4/3 \\ 0 \end{pmatrix}
$$

Collaborative filtering: Correlation

- Now let us calculate correlations, i.e. centered cosine similarity

$$sim(u_i, u_j) = \frac{\mathbf{x}_i^T \mathbf{x}_j}{||\mathbf{x}_i||_2 ||\mathbf{x}_j||_2}$$

$$\begin{aligned} sim(u_1, u_2) &= 0.06299 \\ sim(u_1, u_3) &= -0.59524 \end{aligned}$$

Collaborative filtering: Correlation

- Handles blank spaces as average ratings
- Handles "tough" and "easy" raters
- Some people are just more generous than the others
- It centers the ratings around zero

Collaborative filtering: Correlation

$$\mathbf{sim} = \begin{pmatrix} 1. & 0.06299 & -0.59524 & 0.38576 \\ 0.06299 & 1. & 0. & 0.20412 \\ -0.59524 & 0. & 1. & -0.38576 \\ 0.38576 & 0.20412 & -0.38576 & 1. \end{pmatrix}$$

Collaborative filtering: prediction

- Now we want to predict the rating of the i-th user for the j-th item
- We will select the neighborhood N of the i-th user
- That is k most similar users that also rated item j
- Let $\mathbf{m}_i$ be the vector rating of the i-th user

$$m_{ij} = \frac{1}{k} \sum_{l \in N} m_{lj}$$

Collaborative filtering: prediction

- We want to predict rating of user A for HP2
- $i = 1, j = 2$
- We will use the neighborhood N of size 2
- Two most similar users to user A are B and D, i.e. $N = \{2, 4\}$

$$m_{1,2} = \frac{1}{2} \sum_{l \in \{2,4\}} m_{l,2} = \frac{1}{2}(m_{2,2} + m_{4,2}) = 4$$

Collaborative filtering: example

User \ Movie	HP1	HP2	HP3	Hobbit	SW1	SW2	SW3
A	4	4		5	2		
B	5	5	2				
C				2	4	5	
D		3			1	1	3

Collaborative filtering: prediction

- Then to predict top 5, 10, 15, . . . movies we predict ratings for all movies that the user has not rated and pick the largest

User \ Movie	HP1	HP2	HP3	Hobbit	SW1	SW2	SW3
A	4	4	2	5	2	3	3
B	5	5	2				
C				2	4	5	
D		3			1	1	3

Collaborative filtering: prediction

- Although a toy example we see a possible problem:

Movie / User	HP1	HP2	HP3	Hobbit	SW1	SW2	SW3
A	4	4	2	5	2	3	3
B	5	5	2				
C				2	4	5	
D		3			1	1	3

Collaborative filtering: prediction

- We want to predict the rating of the i-th user for the j-th item
- We will again select the neighborhood N of the i-th user
- That is k most similar users that also rated item j
- Let $\mathbf{m}_i$ be the vector rating of the i-th user
- We then build a similarity weighted average:

$$m_{ij} = \frac{\sum_{l \in N} sim(u_i, u_l) m_{lj}}{\sum_{l \in N} |sim(u_i, u_l)|}$$

Collaborative filtering: prediction

- We want to predict rating of user A for HP2
- $i = 1$, $j = 2$
- We will use the neighborhood N of size 2
- Two most similar users to user A are B and D, i.e. $N = \{2, 4\}$

$$m_{1,2} = \frac{\sum_{l \in \{2,4\}} sim(u_1, u_l) m_{l,2}}{\sum_{l \in \{2,4\}} |sim(u_1, u_l)|} = \frac{0.06299 m_{2,2} + 0.38576 m_{4,2}}{0.06299 + 0.38576} = 3.2807$$

Collaborative filtering: example

Movie / User	HP1	HP2	HP3	Hobbit	SW1	SW2	SW3
A	4	3.2807		5	2		
B	5	5	2				
C				2	4	5	
D		3			1	1	3

Collaborative filtering: prediction

- Then to predict top 5, 10, 15, ... movies we predict ratings for all movies that the user has not rated and pick the largest

User \ Movie	HP1	HP2	HP3	Hobbit	SW1	SW2	SW3
A	4	3.2807	2	5	2	-2.6406	3
B	5	5	2				
C				2	4	5	
D		3			1	1	3

Collaborative filtering

- What we did so far was user-user collaborative filtering
- A dual approach is item-item collaborative filtering
- The idea is the same as before only we switch users and items
- Thus, for an item j we find its neighborhood, which are its most similar items
- Then estimate rating based on the ratings from the similar items
- We can use same approaches and similarities as before

User-user vs. Item-item

- In theory user-user and item-item are dual approaches
- In practice item-item outperforms user-user
- Items are “simpler” than users
- Items belong to a small set of “genres”, user tastes vary greatly
- Users have multiple interests
- Item similarity is more meaningful than user similarity

Collaborative filtering recommendations

- High complexity and therefore we typically pre-compute similarities
- Dimensionality reduction
- Advantage: no feature selection needed
- Problem: Cold start (not enough users to find a match)
- Problem: Sparsity (too many items – some items do not have enough ratings)
- Problem: first rater (can not recommend unrated items)
- Problem: popularity bias (Harry Potter effect)

Collaborative filtering recommendations

- Solutions
- Hybrid approaches
- Combine collaborative filtering with some other methods
- Combine content based method with CF
- Global averages (global average rating, user averages, item averages)
- Dimensionality reduction: Factor analysis

Item profiles

- Let us again think about the utility matrix as a final result of a rating process
- For example, items have some (general) characteristics that users like/dislike and because of these characteristics they rate them in one way or another
- From these characteristics we build **item profiles**
- Thus, an item profile is a **description of an item**
- But, it is difficult to define the features manually
- Since many of these are hidden (latent)

User profiles

- The same set of features might be used to represent the preferences of the users
- We might represent the preferences as the feature weights
- E.g. a feature which the user prefers gets a higher weight
- The final rating of a user for an item might be then the weighted sum of the features from the item profile
- This idea can be used to predict the missing values in the utility matrix

Representing profiles: example

- Let us represent the users as vectors $\mathbf{u}$ with weights for the features representing how much a user liked certain feature
- E.g. a user who highly rated thrillers and Edward Norton might be represented as the following vector

$$\mathbf{u}_1 = \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \end{pmatrix}$$

Representing profiles: example

- Note that with the weights we might express also that a user dislikes a particular feature

$$\mathbf{u}_2 = \begin{pmatrix} -2 \\ 0 \\ 3 \\ 0 \\ 0 \\ 2 \\ 2 \end{pmatrix}$$

Representing profiles: example

- Now, we can calculate a rating that a user gives for a certain movie by calculating $\mathbf{u}^T\mathbf{v}$

$$\mathbf{u}_1^T\mathbf{v}_1 = \begin{pmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0.5 \\ 0.5 \end{pmatrix} = 1$$

Representing profiles: example

$$\mathbf{u}_2^T \mathbf{v}_1 = \begin{pmatrix} -2 & 0 & 3 & 0 & 0 & 2 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0.5 \\ 0.5 \end{pmatrix} = 3$$

Representing profiles: example

$$\mathbf{u}_1^T \mathbf{v}_2 = \begin{pmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = 4$$

Representing profiles: example

$$\mathbf{u}_2^T \mathbf{v}_2 = \begin{pmatrix} -2 & 0 & 3 & 0 & 0 & 2 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = 2$$

Representing users: The $\mathbf{U}$ Matrix

- We can now group all user vectors $\mathbf{u} \in \mathbb{R}^d$ into a matrix $\mathbf{U} \in \mathbb{R}^{n \times d}$
- n is the number of users, and d is the number of features

$$\begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_n^T \end{pmatrix}$$

Representing items: The $\mathbf{V}$ Matrix

- We can now group all item vectors $\mathbf{v} \in \mathbb{R}^d$ into a matrix $\mathbf{V} \in \mathbb{R}^{d \times m}$
- m is the number of items, and d is the number of features

$$\begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots \mathbf{v}_m \end{pmatrix}$$

The product: **UV**

- Now, the utility matrix $\mathbf{M}$ is given by:

$$\mathbf{M} = \mathbf{U}\mathbf{V}$$

- We will decompose $\mathbf{M}$ to obtain $\mathbf{U}$ and $\mathbf{V}$
- We will reduce the dimensions of $\mathbf{M}$
- We can also use $\mathbf{U}$ and $\mathbf{V}$ to predict missing values in $\mathbf{M}$

UV Decomposition

- We start with $\mathbf{M} \in \mathbb{R}^{n \times m}$ and want to find $\mathbf{U} \in \mathbb{R}^{n \times d}$ and $\mathbf{V} \in \mathbb{R}^{d \times m}$
- $\mathbf{UV}$ closely approximates $\mathbf{M}$
- If we are able to find this decomposition than we have established that there are d dimensions that allow us to characterize both users and items closely
- This process is called *UV decomposition*

UV Decomposition: example

$$\begin{pmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{pmatrix} = \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{pmatrix} \times \begin{pmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{pmatrix}$$

Root-Mean-Square-Error

- We approximate $\mathbf{M}$ $\rightarrow$ we need to measure the approximation error
- We can pick among several measures for this error
- A typical choice is the root-mean-square-error (RMSE):
    1. Sum over all nonblank entries in $\mathbf{M}$ the square of the difference between that entry and the corresponding entry in the product $\mathbf{UV}$
    2. Take the average of these squares by dividing by the number of terms in the sum (i.e. the number of nonblank entries in $\mathbf{M}$)
    3. Take the square root of the mean
- Minimizing the sum of the squares is equivalent to minimizing the square root of the average square, thus we can omit the last two steps

Root-Mean-Square-Error: example

- Suppose we start with **U** and **V** with all ones:

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix}$$

Root-Mean-Square-Error: example

$$\begin{pmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{pmatrix} - \begin{pmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 2 & 2 & 1 \\ 1 & -1 & 0 & 2 & -1 \\ 0 & & 1 & -1 & 2 \\ 0 & 3 & 2 & 1 & 3 \\ 2 & 2 & 3 & 2 & \end{pmatrix}$$

Root-Mean-Square-Error: example

- Sum of squares:
  1. Row 1: 18
  2. Row 2: 7
  3. Row 3: 6
  4. Row 4: 23
  5. Row 5: 21
- Total sum: 75
- We can already stop at this point

Incremental computation

- Finding the decomposition with the least RMSE involves starting with some arbitrarily chosen **U** and **V** and iteratively adapting the matrices to make the RMSE smaller
- We consider only adjustments to a single element of **U** or **V**
- In principle we could also make more complex adjustments
- In a typical example we will encounter many *local minima*
- In that case no allowable adjustments to **U** or **V** will make the RMSE smaller

Incremental computation

- Only one of these will be the *global minimum*
- That is the the least possible RMSE
- To increase the chances of finding the global minimum we may start the iteration many times with different starting points
- However, there is no guarantee that we will find the global minimum

Incremental computation: example

- Suppose we start with **U** and **V** with all ones and make a single adjustment (u_{11}):

$$\begin{pmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix}$$

Incremental computation: example

- Sum of squares:
  - Row 1: $(5-(x+1))^2+(2-(x+1))^2+(4-(x+1))^2+(4-(x+1))^2++(3-(x+1))^2$
  - This simplifies to: $(4-x)^2+(1-x)^2+(3-x)^2+(3-x)^2+(2-x)^2$
- We are looking for x that minimizes the sum:

$$\frac{ds}{dx} = 0$$

Incremental computation: example

$$\frac{ds}{dx} = -2((4-x)+(1-x)+(3-x)+(3-x)+(2-x)) = 0$$

- This gives $x = 2.6$

$$\begin{pmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 3.6 & 3.6 & 3.6 & 3.6 & 3.6 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix}$$

Incremental computation: example

- Now we would again make a single adjustment (v_{11}) and repeat the process

$$\begin{pmatrix} 2.6 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \times \begin{pmatrix} y & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2.6y+1 & 3.6 & 3.6 & 3.6 & 3.6 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \\ y+1 & 2 & 2 & 2 & 2 \end{pmatrix}$$

Optimizing an arbitrary element

- How does the general formula look like?
- We denote with $\mathbf{P} = \mathbf{UV}$ the current product of matrices $\mathbf{U}$ and $\mathbf{V}$
- Suppose we want to vary u_{rs} and find the value of this element that minimizes the RMSE
- Note that u_{rs} only affects the elements in the r-th row of $\mathbf{P}$:

$$p_{rj} = \sum_{k=1}^{d} u_{rk} v_{kj} = \sum_{k \neq s} u_{rk} v_{kj} + x v_{sj}$$

- We sum over all nonblank values m_{rj}
- We replaced u_{rs} with x

Optimizing an arbitrary element

- If m_{rj} is a nonblank element then the contribution of this element to RMSE is given by:

$$(m_{rj} - p_{rj})^2 = (m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} + x v_{sj})^2$$

- Now, we can sum over all squares of errors on nonblank entries of $\mathbf{M}$

$$\sum_{j} (m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} + x v_{sj})^2$$

Optimizing an arbitrary element

- We take the derivative with respect to x and set it equal to 0:

$$\sum_{j} -2v_{sj}(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj} + x v_{sj}) = 0$$

- We then solve for x:

$$x = \frac{\sum_j v_{sj}(m_{rj} - \sum_{k \neq s} u_{rk} v_{kj})}{\sum_j v_{sj}^2}$$

Optimizing an arbitrary element

- Similarly, we can derive a formula for element y when we vary v_{rs}

$$y = \frac{\sum_i u_{ir}(m_{is} - \sum_{k \neq r} u_{ik} v_{ks})}{\sum_i u_{ir}^2}$$

The complete algorithm

- Preprocessing: adjusting scales by e.g. subtracting the average in rows and then columns
- Initialization: many different initializations, e.g. the elements that give the product the averages of the elements in the utility matrix
- Optimization: e.g. we always change a single element and pick an order of change (row-by-row, etc)
- Convergence: when the improvements in RMSE fall below a threshold we may stop

Gradient Descent

- This technique for finding the decomposition is an example of *gradient descent*
- We are given some data points: nonblank entries of the utility matrix
- For each data point we find the direction of change that most decreases the RMSE
- If the utility matrix is too large to visit each nonblank point several times
- We might randomly select a fraction of data
- *Stochastic gradient descent*

Overfitting

- One problem that may arise
- We arrive at one local minima that fits very well to the given data
- But it fails to reflect the underlying process that generates the data
- In other words, the RMSE is small on the given data, but it does not do well predicting future data
- This problem is called *overfitting*

Avoid overfitting

- Move the values only a fraction of way towards its optimized value (in the beginning)
- Stop revisiting elements of $\mathbf{U}$ and $\mathbf{V}$ well before the process has converged
- Take several different decompositions and when predicting predict the average of the results of using each decomposition

Factors: interpretation

- We reduced the dimension of the utility matrix to d dimensions
- These dimension are called **factors**
- Our $\mathbf{U}$ matrix connects users with the latent factors, i.e. it is a projection of users to the latent factors space
- Our $\mathbf{V}$ matrix connects movies with the latent factors, i.e. it is a projection of movies to the latent factors space
- These projections bring similar users/movies together

Factors: interpretation

- Connection with SVD ($\mathbf{U\Sigma V}^T$)
- Our $\mathbf{U}$ is the $\mathbf{U}$ matrix from SVD
- Our $\mathbf{V}$ is the $\mathbf{\Sigma V}^T$ from SVD
- The complication is that in the utility matrix we have blank spaces
- For SVD we have to have all elements in the matrix, i.e. we need to make assumption such as set to zero, set to average, etc.

Factors: interpretation

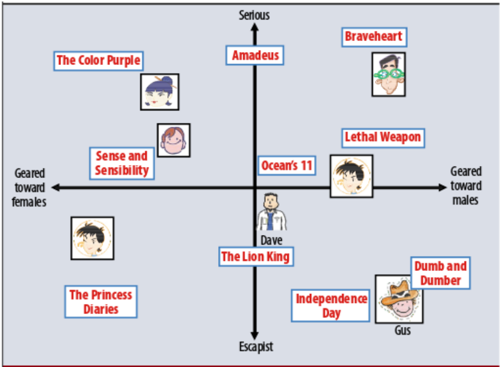


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

- From the winners of the Netflix prize paper: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5197422

Factors: interpretation

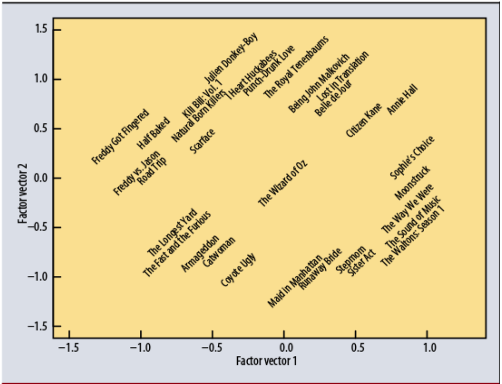


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.