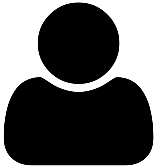
Motivation & Intuition behind Matrix Factorisation

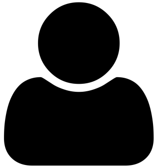
Challenges for Neighbourhood Based Methods

- Synonymy: In real life, different product names can refer to similar objects
  - Similarity based recommender system can't find this hidden association and might treat these objects differently
- Example:



Alice

Recycled Letter pads

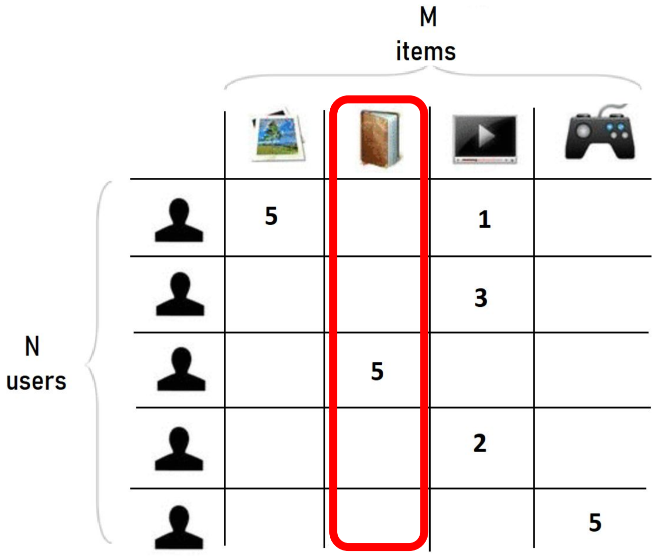


Sam

Recycled Memo pads

Challenges for Neighbourhood Based Methods

- <u>Sparsity:</u> Due to lack of pair of users and items with common ratings, often neighbourhood based methods fail to recommend any item or make predictions



Matrix Factorization

- Objective is to represent user preferences as a combination of
  - User’s interest in an item attribute (e.g. movie genre) and
  - Extent to which the given item is relevant to that attribute

- So using the rating matrix, we want to first calculate the strength of user interest for each user for let’s say a genre
  - Let’s say User Alice is interested in Sci-fi movies
  - Now For a movie ‘Interstellar’ We would find out ‘how sci-fi is this movie’
  - Finally predict rating for interstellar given by Alice based on these 2 values
  - But how do we achieve this mathematically?

Rating Prediction using Matrix Factorization and SVD

Matrix Factorization

- SVD is the factorization of a matrix M into 3 constituent matrices

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

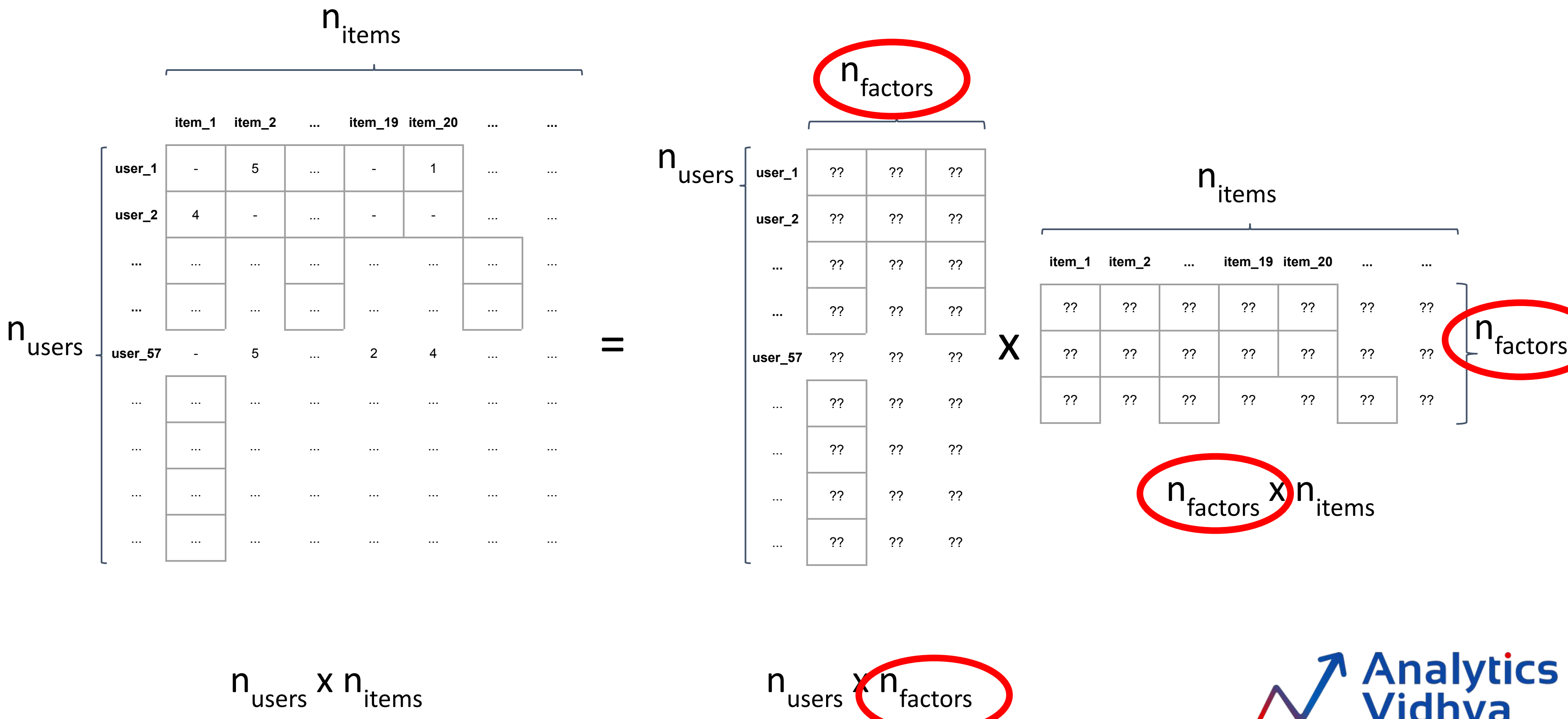
where U and V are called *left* and *right singular vectors* and the values of the diagonal of Σ are called the *singular values*

- We can approximate the full matrix by observing only the most important features – those with the largest singular values
- This can be done by decomposing rating matrix into a user and item matrix using a dimensionality reduction technique

Singular Value Decomposition

- $$R = P\Sigma Q^T$$
- R is *m x n* ratings matrix
- P is *m x k* user-feature affinity matrix
- Q is *n x k* item-feature relevance matrix
- Σ is *k x k* diagonal feature weight matrix
- SVD defines a shared vector space for item and users (feature space)

Matrix Factorization



Interpretation of the User and Item matrices

attribute k an item can possess and a user can be susceptible to

how much user u is susceptible to this attribute k / how much it is important to them.

	user hidden factor k-1	user hidden factor k	user hidden factor k+1
user_1	??	??	??
user_2	??	??	??
...	??	??	??
...	??	??	??
user_u	??	??	??
...	??	??	??
...	??	??	??
...	??	??	??
...	??	??	??

User

	item hidden factor k-1	item hidden factor k	item hidden factor k+1
item_1	??	??	??
item_2	??	??	??
...	??	??	??
item_i	??	??	??
...	??	??	??
...	??	??	??
...	??	??	??

how much item i possesses this attribute k

Item

Example for SVD-based recommendation

- **SVD:** $M_k = U_k \times \Sigma_k \times V_k^{\,T}$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

V_k^T	Terminator	Die Hard	Twins	Eat Pray Love	Pretty Woman
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- **Prediction:** $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

$$= 3 + 0.84 = 3.84$$

What does SVD Achieve?

- SVD captures hidden relationships between users and items

Solving Problem of
Synonymy & Sparsity

- SVD provides lower dimension representation of the original user-item space

Solving Problem of
Scalability

2006 "Funk-SVD" and 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
    - S. Funk (pen name)
        - Use fast gradient descent optimization procedure
        - http://sifter.org/~simon/journal/20061211.html

Algorithm Structure

- Initialize values to train (item/user feature vectors) to arbitrary starting point
  - Must be non-zero
  - Usually must be random
- Try to predict each rating in the dataset
- Use error and update rule to update values for next rating/sample
- Iterate until convergence
  - Stops moving
  - Iterated enough times

Get Rid of Sigma

- Decomposition:

$$\boldsymbol{R} = \boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{Q}^{\boldsymbol{T}}$$

$$\boldsymbol{R} = \boldsymbol{P}\boldsymbol{Q}^{\boldsymbol{T}}$$

- Scoring Rule after dropping Sigma

$$s(i;u) = \hat{r}_{ui} = \sum_f p_{uf} q_{if}$$

Deriving FunkSVD

- Recall our prediction equation

$$s(i;u) = \hat{r}_{ui} = \sum_f p_{uf} q_{if}$$

- We compute Error

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$
$$= r_{ui} - \sum_f p_{uf} q_{if}$$

- We then compute the derivatives $\frac{d}{dp_{uf}} e_{ui}^2$ and $\frac{d}{dq_{if}} e_{ui}^2$

$$\theta = < P, Q > \qquad \theta_n = \theta_{n-1} + \Delta g(\theta_{n-1})$$

Analytics Vidhya

Deriving FunkSVD

- Calculating derivative for puf and qif

$$\frac{d}{dp_{uf}} e_{ui}^2 = 2e_{ui} \frac{d}{dp_{uf}} e_{ui}$$

$$= 2e_{ui} \frac{d}{dp_{uf}} (r_{ui} - \sum_f p_{uf} q_{if})$$

$$= -2e_{ui} q_{if}$$

$$p'_{uf} = p_{uf} - \lambda(-2e_{ui} q_{if})$$
$$q'_{if} = q_{if} - \lambda(-2e_{ui} p_{uf})$$

- Final Equations (add regularization to discourage large values)

$$p_{uf} = p_{uf} + \lambda(e_{ui} q_{if} - \gamma p_{uf})$$
$$q_{if} = q_{if} + \lambda(e_{ui} p_{uf} - \gamma q_{if})$$

Analytics Vidhya

Summary

- Matrix factorization
  - Generate low-rank approximation of matrix
  - Detection of latent factors
  - Projecting items and users in the same n-dimensional space
- Prediction quality can increase as a consequence of…
  - Small & faster model
  - 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
  - Parameters can be determined and fine-tuned only based on experiments

Collaborative Filtering Issues

- Pros:
  - Well-understood,
  - Works well in some domains
  - No knowledge engineering required
- Cons:
  - Requires user community,
  - Sparsity problems
- What is the best CF method?
  - In which situation?
  - Which domain?
- Other Methods
  - Probabilistic Methods
  - Association Rule Mining

Analytics Vidhya