Chapter 1
Introduction

Abstract Representing data in lower dimensional spaces has been used extensively in many disciplines such as natural language and image processing, data mining, and information retrieval. Recommender systems deal with challenging issues such as scalability, noise, and sparsity and thus, matrix and tensor factorization techniques appear as an interesting tool to be exploited. That is, we can deal with all aforementioned challenges by applying matrix and tensor decomposition methods (also known as factorization methods). In this chapter, we provide some basic definitions and preliminary concepts on dimensionality reduction methods of matrices and tensors. Gradient descent and alternating least squares methods are also discussed. Finally, we present the book outline and the goals of each chapter.

Keywords Matrix decomposition · Tensor decomposition

1.1   Recommender Systems

The Web contains more than 4.9 billion pages until the date this book was published and it is growing rapidly day by day. There is a huge amount of information, which burdens people to find what they may need. To overcome this problem, we often rely on recommendations from others who have more experience on a topic. In the Web, this is attained with the help of a collaborative filtering (CF) algorithm, which provides recommendations based on the suggestions of users, who have similar preferences for products. Basically, it is an algorithm for matching people with similar interests under the assumption that similar people like similar products. These kind of recommendations are provided from systems, known as recommender systems. Recommender systems use techniques, which are widely studied in research communities of information retrieval, machine learning, and data mining [16, 36]. These systems have been developed to effectively support the customer's decision-making process mainly for commercial purposes (i.e., what product to buy on Amazon.com, what TV show or movie to rent on Netflix.com, etc.).

In the following, we will discuss an example of a simple recommender system with four users, who have rated four movies, as shown in Fig. 1.1.
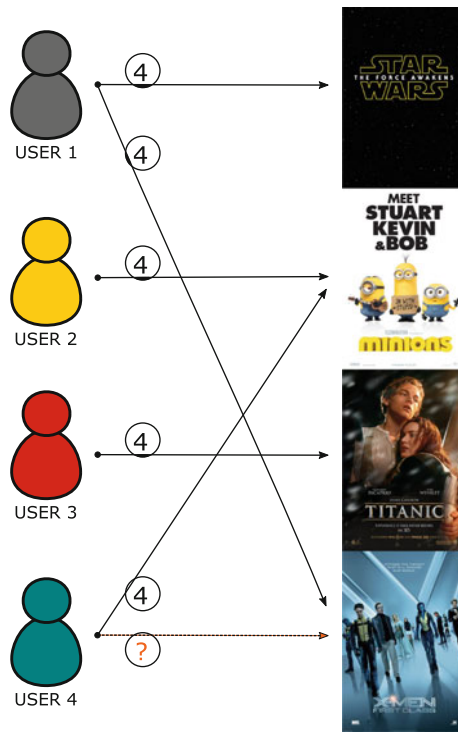

Fig. 1.1 Four users have rated four movies using a [1–5] rating scale

Figure 1.1 presents a weighted user-movie bipartite graph, where each edge between a user and a movie has a weight which indicates the degree of preference of a user for that movie (using a [0–5] rating scale). For reasons of clarity, in Fig. 1.1, we present only "positive" ratings (≥ 4). Let us assume in our running example that we want to predict the rating of user 4 on movie 4 (X-Men). To show this assumption, we use a dotted line for the edge that connects user 4 with movie 4 and the possible rating is shown with a question mark. The challenge of a recommender system is to correctly predict a rating for those items for which a user has not expressed explicitly her preference (with no rating value at all). In case our recommender system predicts this rating as positive (i.e., higher than 4 in the rating scale 0–5), then, it could recommend movie 4 (X-Men) to the target user 4.

Our example of Fig. 1.1 can be represented with a user–item rating matrix A, which is shown in Fig. 1.2a. Please notice that in contrast to Fig. 1.1, Fig. 1.2 also presents ratings with values 0, 1, 2, and 3. In the case that we read matrix A horizontally, Fig. 1.2 represents the ratings that a user U_i gives to a set of movies I_j, where

$j \geq 1$. In the case that we read the matrix A vertically, Fig. 1.2 represents the ratings that one movie (i.e., I_1) receives from several users U_i, where $i \geq 1$.

To ease the discussion, we will use the running example illustrated in Fig. 1.2 where I_{1-4} are items and U_{1-4} are users. As shown, the example data set is divided into training and test sets. The null cells (no rating) are presented as zeros.

(a)

	I_1	I_2	I_3	I_4
U_1	4	1	1	4
U_2	1	4	2	0
U_3	2	1	4	5

(b)

	I_1	I_2	I_3	I_4
U_4	1	4	1	?

Fig. 1.2 **a** Training set (3×4), **b** Test set (1×4)

1.2 Recommender Systems in Social Media

Online social networks (OSNs) contain gigabytes of data that can be mined to provide product recommendations to a target user. Besides explicit friendship relations among users, there are also other implicit relations. For example, users can co-comment on products and can co-rate them. Thus, item recommendation can be provided in such systems based on the suggestions of our friends whom we trust. Recommender systems are widely used by OSNs to stimulate users to extend their personal social graph or for marketing purposes by recommending products that their friends have liked. Moreover, recommender systems can act like filters, trying to provide the right personalized information to each different user. Typically, a recommendation algorithm takes as input the preferences of the user and her friends from their profile and conscripts them to yield recommendations for new ties such as friends, companies, places, products, etc.

In the following, we will discuss a more extended example, where we will try to provide recommendations to user 4 by exploiting both her ratings on movies and her friendship network, as shown in Fig. 1.3, which consists of two layers. The first layer contains the friendship network among users, whereas the second layer shows the movies, which are rated only positively by users.

Our recommender system's task is to predict a rating for user 4 on movie 4. As shown, user 4 is a friend of user 2, who does not like the X-Men movie at all (rated it with 0 in the [0–5] rating scale as shown in Fig. 1.2a). Based on the theory of homophily, which claims that friends may share common characteristics (i.e., beliefs, values, preferences, etc.), we cannot predict a high rating for user 4 on the X-Men movie. This is a simple example of how a social network (i.e., friendship network) can help a recommender system to leverage the quality of its recommendations by exploiting the information from the friendship network.
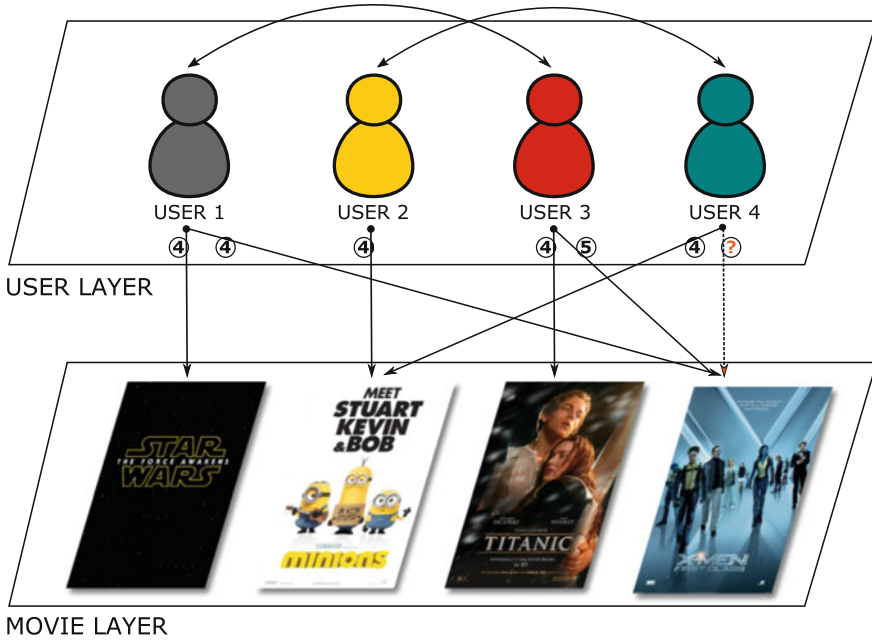
Fig. 1.3 The users' friendship network and the user-movie rating information

1.3 Matrix Factorization

Recommender systems mainly base their suggestions on rating data of two entities (users and items), which are often placed in a matrix with one representing users and the other representing items of interest. For example, Netflix collects ratings for movies using the five-star selection schema, and TiVo users indicate their preferences for TV shows by pressing thumbs-up and thumbs-down buttons. These ratings are given explicitly by users creating a sparse user–item rating matrix, because an individual user is likely to rate only a small fraction of the items that belong to the item set. Another challenging issue with this user–item rating matrix is scalability of data (i.e., the large number of possible registered users or inserted items), which may affect the time performance of a recommendation algorithm.

We can deal with all aforementioned challenges by applying matrix decomposition methods (also known as factorization methods). *Matrix factorization* denotes a process, where a matrix is factorized into a product of matrices. A matrix factorization method is useful for solving plenty of problems, both analytical and numerical; an example of a numerical problem is the solution of linear equations and eigenvalue problems. Its importance relies on the exploitation of latent associations that exist

in the data among participating entities (e.g., between users and items). In a trivial form, the matrix factorization method uses two matrices, which hold the information of correlation between the user-feature and item-feature factors, respectively.

Figure 1.4 shows an example of latent factors, which could be revealed after performing matrix decomposition. As shown, the X′X axis divides both people and movies according to sex (e.g., male or female). When a movie is closer to the female part of X′X axis, it means that this movie is most popular among women rather than
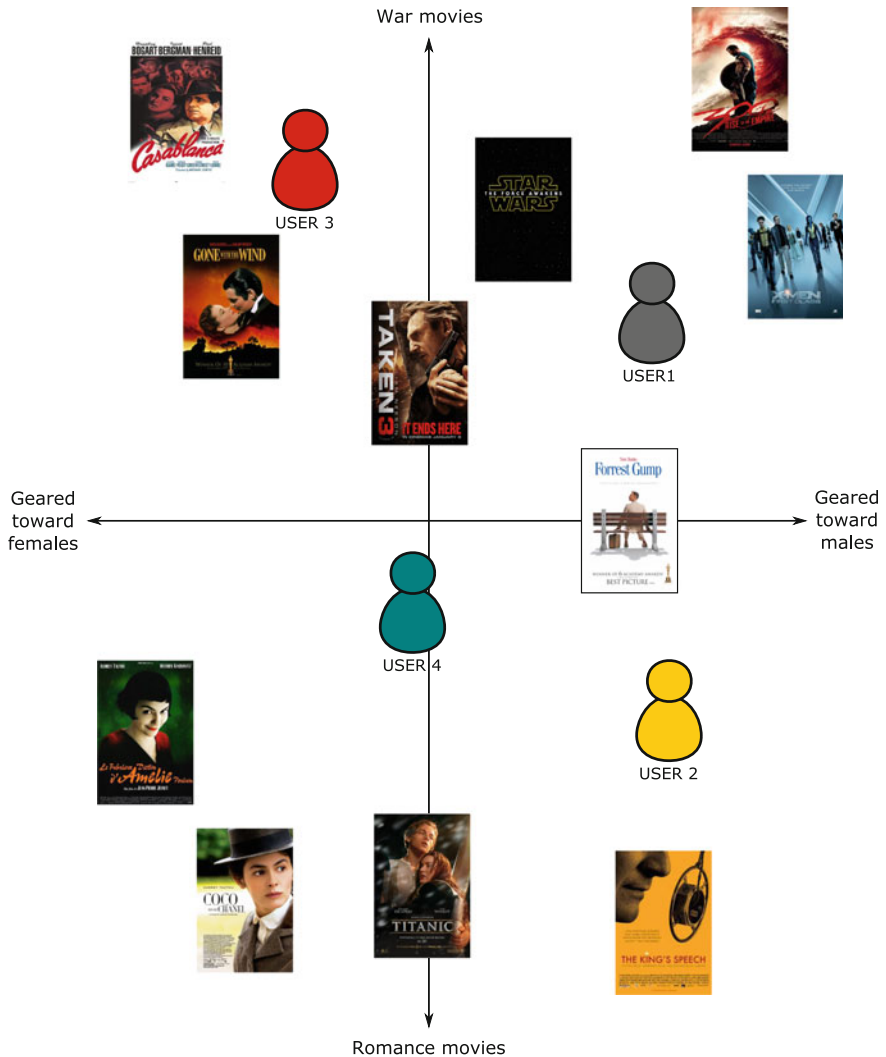

Fig. 1.4 A simplified illustration of the latent factors, which characterizes both users and movies using two axes—male versus female and war-like versus romantic

men. The Y′Y axis divides people and movies as "war-like" and "romantic." A "war-like" viewer is assumed to prefer movies showing blood and deaths. In contrast, a "romantic" viewer chooses movies that present love and passion. To predict a user's rating of a movie, we can compute the dot product of the movie's and user's [x, y] coordinates on the graph. In addition, Fig. 1.4 shows where movies and users might fall on the basic two dimensions. For example, we would expect *user 3* to love "*Casablanca*," to hate "*The King's Speech*," and to rate "*Amelie*" above average. Note that some movies (i.e., "*Taken 3*") and users (i.e., *user 4*) would be characterized as fairly neutral on these two dimensions.

One strong point of matrix factorization is that it also allows the incorporation of additional information. When explicit feedback is not available, recommender systems can infer user preferences using implicit feedback, which indirectly reflects opinions by observing user behavior including purchase history, browsing history, search patterns, or even mouse movements. Implicit feedback usually denotes the presence or absence of an event, so it is typically represented by a densely filled matrix [26].

There are many matrix factorization methods (see Chap. 2). The popularity of these methods and the motivation to study them more in recent years came by their advantage of combining high-performance scalability with good predictive accuracy.

1.4 Tensor Factorization

Standard CF-based algorithms operate on matrices (second-order tensors) representing relations between users and items. However, real-world problems usually consist of more than two participating entities. For example, social tagging systems (STSs) mainly consist of three participating entities (users, items, and tags). Moreover, in location-based social networks (LBSNs), we have also three interacting entities (users, locations, and tags). In LBSNs, users can share location-related information with each other to leverage the collaborative social knowledge. LBSNs consist of a new social structure made up of individuals connected by interdependency derived from their locations in the physical world as well as their location-tagged media content, such as photographs, videos, and texts. As shown in Fig. 1.5, users visit locations in the real world and provide geo-tagged information content (e.g., comments, photographs, videos). In particular, Fig. 1.5 presents three layers, namely user, location, and content. It is obvious that someone could exploit information from each layer independently to leverage recommendations. However, in a more advanced case, we could also exploit ternary relation among entities (i.e., user, location, and content), which goes through all layers.

Because of the ternary relation of data in many cases (e.g., STSs, LBSNs, etc.), many recommendation algorithms originally designed to operate on matrices cannot be applied. Higher order problems put forward new challenges and opportunities for recommender systems. For example, ternary relation of STSs can be represented as a third-order tensor $\mathcal{A} = (a_{u,i,t}) \in \mathbb{R}^{|U| \times |I| \times |T|}$. Symeonidis et al. [44],
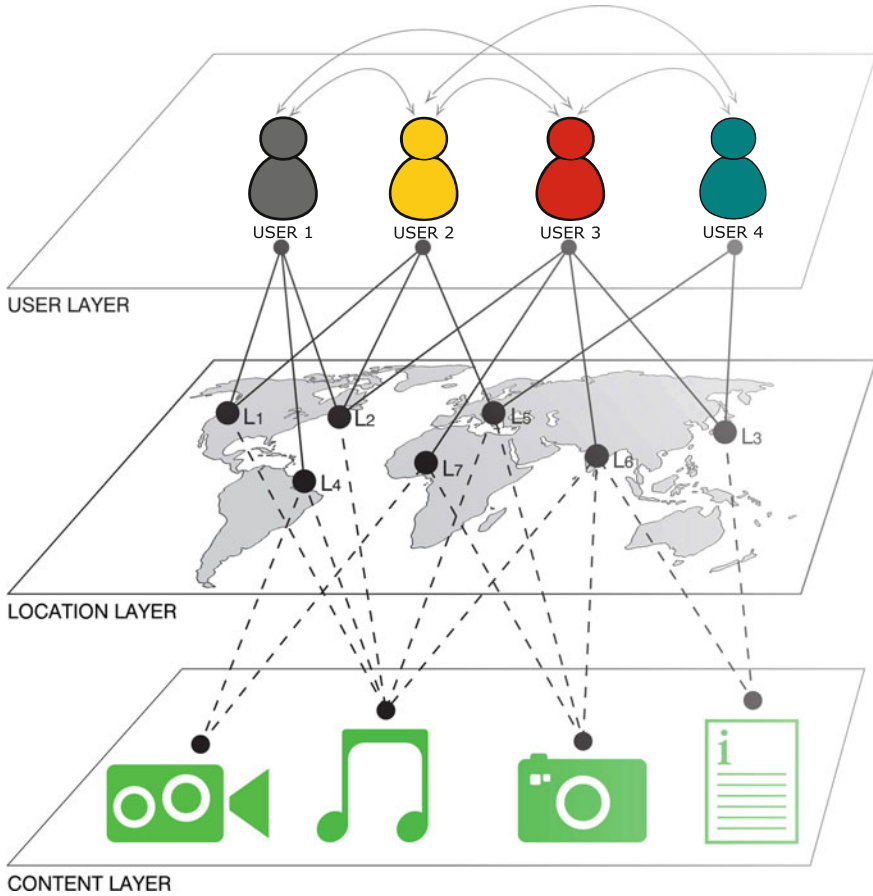
Fig. 1.5 Visual representation of users, locations, and content (i.e., photographs/videos, tags, etc.)

for example, proposed to interpret the user assignment of a tag on an item, as a binary tensor where 1 indicates observed tag assignments and 0 missing values (see Fig. 1.6):

$$a_{u,i,t} := \begin{cases} 1, & \text{if user } u \text{ assigns on item } i \text{ tag } t \\ 0, & \text{otherwise} \end{cases}$$

Tensor factorization techniques can be employed in order to exploit the underlying latent semantic structure in tensor $\mathcal{A}$. While the idea of computing low-rank tensor approximations has already been used in many disciplines such as natural language, image processing, data mining and information retrieval [11, 13, 24, 28, 42, 43, 46], just a few years ago, it was applied to recommendation problems in STSs and

Fig. 1.6 Tensor representation of a STS where positive feedback is interpreted as 1 (i.e., $a_{utr} := 1$) and the rest as 0 (i.e., $a_{utr} := 0$)
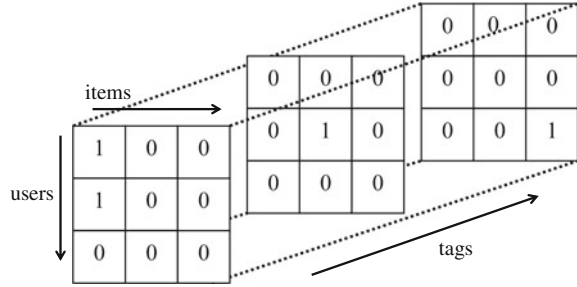
LBSNs. The basic idea is to transform the recommendation problem as a third-order tensor completion problem, by trying to predict nonobserved entries in $\mathcal{A}$.

1.5 Mathematical Background and Notation

In this section, we provide all important notations of variables and symbols, which will be used throughout the book. Moreover, we provide some basic theorems or mathematical definitions as preliminary knowledge to help the reader understand more easily the concepts that will be discussed later.

The notation of every matrix, variable symbol, and any other basic mathematical term is presented in Tables 1.1 and 1.2, respectively.

Linear algebra plays an important role in matrix and tensor decomposition. Therefore, preliminary concepts on matrices drawn from linear algebra are reviewed in this section.

A *diagonal matrix* (also known as *square matrix*) is a matrix in which entries outside the main diagonal are all zero. Thus, a matrix A with $N \times N$ dimensions is diagonal if the following constraint is satisfied:

$$a_{ij} = 0 \quad if \quad i \neq j \quad \forall i, j \in \{1, 2, \ldots, N\} \tag{1.1}$$

The (column) *rank* of a matrix $A \in \mathbb{R}^{N \times M}$ is defined to be the number of linearly independent column vectors. The (row) *rank* of A is defined to be the number of linearly independent row vectors of A.

A square matrix $A \in \mathbb{R}^{N \times N}$ is called *invertible* (or *nonsingular*), if there is a matrix $B \in \mathbb{R}^{N \times N}$ such that:

$$AB = I \quad \text{and} \quad BA = I \tag{1.2}$$

where $I \in \mathbb{R}^{N \times N}$ is the *identity matrix*. A square matrix that is not invertible is called *singular*. A is singular, if its rank is less than N.

Table 1.1 Definition of matrices and variables that will be used throughout this book

Matrices and variables		
I	Identity matrix	An $n \times n$ square matrix with 1s on the main diagonal and 0s elsewhere
$\mathbb{R}$	Real numbers	The set of real numbers
A	User–item rating matrix	This is the user–item rating matrix, which holds ratings of users on movies. Cells with zeros denote the absence of a rating
$\hat{A}$	Prediction matrix	This is a user–item rating matrix, which holds predicted ratings
U	User-latent feature matrix	This matrix holds preferences of users over items on a latent feature space of dimensionality f. In the CUR method, U has another meaning (see Sect. 2.6)
V	Item-latent feature matrix	This matrix expresses how much an item is preferred by users on a latent feature space of dimensionality f
Λ	Eigenvalue matrix	An $r \times r$ diagonal matrix filled with nonzero eigenvalues of matrix A
E	Eigenvector matrix	E ($n \times r$ matrix) stores eigenvectors of A
Γ		Symmetric nonnegative matrix with diagonal elements equal to zero and other elements greater than zero
C		Randomly chosen set of r columns of matrix A
R		Randomly chosen set of r rows of matrix A
F	Friendship matrix	Stores the friendship information among users. 1s declare friendship and 0s no friendship
λ	Eigenvalue	An eigenvalue λ
$\mathbf{e}$	Eigenvector	An eigenvector e
η	eta	This variable controls the size of the step toward minimization of an objective function
β	beta	The coefficient that regularize predicted users' ratings on items
γ	gamma	The coefficient that regulates the contribution of the friendship network

An *eigenvector* of a square matrix $A^{N \times N}$ is a nonzero vector $\mathbf{e} \in \mathbb{R}^N$ that satisfies the following equation:

$$A\mathbf{e} = \lambda\mathbf{e}, \tag{1.3}$$

meaning that the vector $A\mathbf{e}$ follows the direction of $\mathbf{e}$. λ is the *eigenvalue* of A corresponding to the eigenvector $\mathbf{e}$.

Table 1.2 Definition of mathematical symbols that will be used throughout this book

Symbols

i, j	Indices	Denote the position of an element inside a matrix
$\neq$	Not equal	Is not equal to
$\leq$	Inequality	Is less than or equal to
$\geq$	Inequality	Is greater than or equal to
$\approx$	Approximately equal	Is approximately equal to
$B^\top$	Transpose symbol	Reflects matrix B over its main diagonal (which runs from top-left to bottom-right) to obtain $B^\top$
B^{-1}	Inverse symbol	The inverse of a square matrix is a matrix such that $BB^{-1} = I$
$\forall$	For all	
$\in$	Set membership	Is an element of
$\notin$	Set membership	Is not an element of
$\rightarrow$	Material implication	if . . . then
$\longleftrightarrow$	Material equivalence	if and only if
b	Vector	Euclidean vector
$\wedge$	And	
$\vee$	Or	
$\cdot$	Dot product	The dot product of vectors or matrices
$\| \ldots \|$	Norm	Euclidean norm
$\sum$	Sum	The sum of elements from beginning to end
∂	Partial derivative	The partial derivative of a function

A square matrix $A \in \mathbb{R}^{N \times N}$ is called *orthogonal*, if column vectors of A form an orthonormal set in $\in \mathbb{R}^N$. In other words, an *orthogonal matrix* is a square matrix with real numbers as entries, whose columns and rows are orthogonal unit vectors as shown below:

$$AA^\top = A^\top A = I \quad : I \text{ is the identity matrix} \tag{1.4}$$

This leads to equivalent characterization: a matrix A is orthogonal if its transpose is equal to its inverse:

$$A^\top = A^{-1} \tag{1.5}$$

The *Frobenius norm* $||A||$ of a matrix is given by:

$$||A|| = \sum_{i=1}^{N} \sum_{j=1}^{M} {a_{ij}}^2. \tag{1.6}$$

1.6 Book Outline

In the sequel, a brief introduction to each chapter of the book follows:

Chapter 2. **Related Work on Matrix Factorization**
In this chapter, we provide the related work on basic matrix decomposition methods. The first method that we discuss is known as eigenvalue decomposition, which decomposes the initial matrix into a canonical form [1, 20, 34, 35, 48]. The second method is nonnegative matrix factorization (NMF), which factorizes the initial matrix into two smaller matrices with the constraint that each element of the factorized matrices should be nonnegative [3, 7, 8, 14, 18, 22, 29–31, 47]. The third method is probabilistic matrix factorization (PMF), which scales well to large data sets. The PMF method performs well on very sparse and imbalanced data sets using spherical Gaussian priors. The last but one method is probabilistic latent semantic analysis (PLSA) which is based on a mixture decomposition derived from a latent class model. This results in a more principled approach which has a solid foundation in statistics. The last method is CUR decomposition, which confronts the problem of density in factorized matrices (a problem that is faced when handling the SVD method) [15, 32, 33].

Chapter 3. **Performing SVD on Matrices and Its Extensions**
In this chapter, we describe singular value decomposition (SVD), which is applied on recommender systems [2, 5, 9, 12, 19, 27, 40, 41]. We discuss in detail the mathematical background and present (step by step) the SVD method using a toy example of a recommender system. We also describe UV decomposition [38] in detail, which is an instance of SVD, as we have mathematically proven. We minimize an objective function, which captures the error between the predicted and the real value of a user's rating. We also provide a step-by-step implementation of UV decomposition using a toy example, which is followed by a short representation of the algorithm in pseudocode form [4, 17, 49]. Finally, an additional constraint of friendship is added to the objective function to leverage the quality of recommendations [25].

Chapter 4. **Experimental Evaluation on Matrix Decomposition Methods**
In this chapter, we study the performance of described SVD and UV decomposition algorithms, against an improved version of the original item-based CF algorithm [23, 39] combined with SVD. For the UV decomposition method, we will present the appropriate tuning of parameters of its objective function to have an idea of how we can get optimized values of its parameters. We will also answer the question if these values are generally accepted or if they should be different for each data set. The metrics we will use are root-mean-square error (RMSE), precision, and recall. The size of a training set is fixed at 75 %, and we perform a fourfold cross-validation.

Chapter 5. **Related Work on Tensor Factorization**
In this chapter, we provide a preliminary knowledge overview of tensors. Moreover, we provide the related work on tensor decomposition methods. The first method that is discussed is the Tucker Decomposition (TD) method [45], which is the underlying tensor factorization model of Higher Order Singular Value Decomposition [28]. TD decomposes a tensor into a set of matrices and one small core tensor. The second one is the PARAFAC method (PARAllel FACtor analysis) [10, 21], which is the same as the TD method with the restriction that the core tensor should be diagonal. The third one is the Pairwise Interaction Tensor Factorization method [37], which is a special case of the TD method with linear runtime both for learning and prediction. The last method that is analyzed is the low-order tensor decomposition (LOTD). This method has low functional complexity, is uniquely capable of enhancing statistics, and avoids overfitting compared with traditional tensor decompositions such as TD and PARAFAC [6].

Chapter 6. **Performing HOSVD on Tensors and Its Extensions**
In this chapter, we describe tensor decomposition for recommender systems in detail. We will use—as a toy example—a tensor with three dimensions (i.e., user–item–tag). The main factorization method that will be presented in this chapter is higher order SVD (HOSVD), which is an extended version of the Singular Value Decomposition (SVD) method. In this chapter, we will present a step-by-step implementation of HOSVD in our toy example. Then, we will present how we can update HOSVD when a new user is registered in our recommender system. We will also discuss how HOSVD can be combined with other methods for leveraging the quality of recommendations. Finally, we will study limitations of HOSVD and discuss in detail the problem of non-unique tensor decomposition results and how we can deal with this problem. We will also discuss other problems in tensor decomposition, e.g., actualization and scalability.

Chapter 7. **Experimental Evaluation on Tensor Decomposition Methods**
In this chapter, we will provide experimental results of tensor decomposition methods on real data sets in STSs. We will discuss the criteria that we will set for testing all algorithms and the experimental protocol we will follow. Moreover, we will discuss the metrics that we will use (i.e., Precision, Recall, root-mean-square error, etc.). Our goal is to present the main factors that influence the effectiveness of algorithms.

Chapter 8. **Conclusions and Future Work**
In this chapter, we will discuss the main conclusions of the experimental evaluation, limitations of each algorithm, and will provide future research directions.

References

1. Bensmail, H., Celeux, G.: Regularized gaussian discriminant analysis through eigenvalue decomposition. J. Am. Stat. Assoc. **91**(436), 1743–1748 (1996)

2. Berry, M., Dumais, S., O'Brien, G.: Using linear algebra for intelligent information retrieval. SIAM Rev. **37**(4), 573–595 (1994)
3. Berry, M.W., Browne, M., Langville, A.N., Paul Pauca, V., Plemmons, R.J.: Algorithms and applications for approximate nonnegative matrix factorization. Comput. Stat. Data Anal. **52**(1), 155–173 (2007)
4. Bhargava, P., Phan, T., Zhou, J., Lee, J.: Who, what, when, and where: multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In: Proceedings of the 24th International Conference on World Wide Web, WWW'15. Republic and Canton of Geneva, Switzerland, pp. 130–140. International World Wide Web Conferences Steering Committee (2015)
5. Brand, M.: Incremental singular value decomposition of uncertain data with missing values. In: Proceedings of the 7th European Conference on Computer Vision (ECCV), pp. 707–720. Copenhagen, Denmark (2002)
6. Cai, Y., Zhang, M., Luo, D., Ding, C., Chakravarthy, S.: Low-order tensor decompositions for social tagging recommendation. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM'11, New York, NY, USA, pp. 695–704. ACM (2011)
7. Campbell, S.L., Poole, G.D.: Computing nonnegative rank factorizations. Linear Algebra Appl. **35**, 175–182 (1981)
8. Cao, B., Shen, D., Sun, J.-T., Wang, X., Yang, Q., Chen, Z.: Detect and track latent factors with online nonnegative matrix factorization. In: IJCAI, pp. 2689–2694 (2007)
9. Cao, L.: Singular value decomposition applied to digital image processing. Division of Computing Studies Arizona State University Polytechnic Campus, Mesa, Arizona, 85212 (2006)
10. Carroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. Psychometrika **35**, 283–319 (1970)
11. Chen, S., Wang, F., Zhang, C.: Simultaneous heterogeneous data clustering based on higher order relationships. In: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops, ICDMW'07, pp. 387–392, Washington, DC, USA. IEEE Computer Society (2007)
12. Chin, T.-J., Schindler, K., Suter, D.: Incremental kernel svd for face recognition with image sets. In: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition (FGR 2006), pp. 461–466. IEEE (2006)
13. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)
14. Dhillon, I.S., Sra, S.: Generalized nonnegative matrix approximations with bregman divergences. In: NIPS, pp. 283–290 (2005)
15. Drineas, P., Kannan, R., Mahoney, M.W.: Fast monte carlo algorithms for matrices III: computing a compressed approximate matrix decomposition. SIAM J. Comput. **36**(1), 184–206 (2006)
16. Fodor, I.: A survey of dimension reduction techniques. Technical Report (2002)
17. Forsati, R., Mahdavi, M., Shamsfard, M., Sarwat, M.: Matrix factorization with explicit trust and distrust side information for improved social recommendation. ACM Trans. Inf. Syst. **32**(4), 17:1–17:38 (2014)
18. Fulekar, M.H.E.: Bioinformatics: Applications In Life And Environmental Sciences. Daya Publishing House (2009)
19. Furnas, G., Deerwester, S., Dumais, S., et al.: Information retrieval using a singular value decomposition model of latent semantic structure. In: Proceedings of ACM SIGIR Conference, pp. 465–480 (1988)
20. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore, MD, USA (1996)
21. Harshman, R.A.: Foundations of the parafac procedure: models and conditions for an 'exploratory' multimodal factor analysis. In: UCLA Working Papers in Phonetics, pp. 1–84 (1970)
22. Kalofolias, V., Gallopoulos, E.: Computing symmetric nonnegative rank factorizations. Linear Algebra Appl. **436**(2), 421–435 (2012). Special Issue devoted to the Applied Linear Algebra Conference (Novi Sad 2010)

23. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proceedings of ACM CIKM Conference, pp. 247–254 (2001)
24. Kolda, T.G., Sun, J.: Scalable tensor decompositions for multi-aspect data mining. In: ICDM'08: Proceedings of the 8th IEEE International Conference on Data Mining, pp. 363–372. IEEE Computer Society, Dec 2008
25. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'08, New York, NY, USA, pp. 426–434. ACM (2008)
26. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
27. de Lathauwer, L., de Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(4), 1253–1278 (2000)
28. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(4), 1253–1278 (2000)
29. Lee, D.D., Seung, H.S.: Learning the parts of objects by nonnegative matrix factorization. Nature **401**, 788–791 (1999)
30. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS, pp. 556–562 (2000)
31. Lin, C.-J.: On the convergence of multiplicative update algorithms for nonnegative matrix factorization. IEEE Trans. Neural Netw. **18**(6), 1589–1596 (2007)
32. Mahoney, M.W., Drineas, P.: Cur matrix decompositions for improved data analysis. Proc. Natl. Acad. Sci. **106**(3), 697–702 (2009)
33. Mahoney, M.W., Maggioni, M., Drineas, P.: Tensor-cur decompositions for tensor-based data. SIAM J. Matrix Anal. Appl. **30**(3), 957–987 (2008)
34. Moler, C.B., Stewart, G.W.: An algorithm for generalized matrix eigenvalue problems. SIAM J. Numer. Anal. **10**(2), 241–256 (1973)
35. Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics. Texts in Applied Mathematics. Springer (2000)
36. Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press, New York, NY, USA (2011)
37. Rendle, S.: Pairwise interaction tensor factorization for personalized tag recommendation
38. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
39. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedinmgs of WWW Conference, pp. 285–295 (2001)
40. Sarwar, B., Konstan, J., Riedl, J.: Incremental singular value decomposition algorithms for highly scalable recommender systems. In: International Conference on Computer and Information Science (2002)
41. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Proceedings of the 5th International Conference in Computers and Information Technology (2002)
42. Shashua, A., Hazan, T.: Non-negative tensor factorization with applications to statistics and computer vision. In: ICML'05: Proceedings of the 22nd International Conference on Machine Learning, pp. 792–799. ACM (2005)
43. Sun, J., Shen, D., Zeng, H., Yang, Q., Lu, Y., Chen, Z.: Cubesvd: a novel approach to personalized web search. In: World Wide Web Conference, pp. 382–390 (2005)
44. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Tag recommendations based on tensor dimensionality reduction. In: RecSys'08: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 43–50. ACM (2008)
45. Tucker, L.: Some mathematical notes on three-mode factor analysis. Psychometrika 279–311 (1966)
46. Wang, H., Ahuja, N.: A tensor approximation approach to dimensionality reduction. Int. J. Comput. Vis. 217–229 (2007)

47. X, W., Gao, X.: Non-negative matrix factorization by maximizing correntropy for cancer clus-
    tering. BMC Bioinform. **14**, 107 (2013)
48. Weisstein, E.W.: Eigen decomposition from mathworld.wolfram.com. http://mathworld.
    wolfram.com/EigenDecomposition.html
49. Yuan, Q., Chen, L., Zhao, S.: Factorization vs. regularization: fusing heterogeneous social
    relationships in top-n recommendation (2011)