Probabilistic Approaches to Recommendations

Synthesis Lectures on Data Mining and Knowledge Discovery

Synthesis Lectures on Data Mining and Knowledge Discovery is edited by Jiawei Han, Lise Getoor, Wei Wang, Johannes Gehrke, and Robert Grossman. The series publishes 50- to 150-page publications on topics pertaining to data mining, web mining, text mining, and knowledge discovery, including tutorials and case studies. The scope will largely follow the purview of premier computer science conferences, such as KDD. Potential topics include, but not limited to, data mining algorithms, innovative data mining applications, data mining systems, mining text, web and semi-structured data, high performance and parallel/distributed data mining, data mining standards, data mining and knowledge discovery framework and process, data mining foundations, mining data streams and sensor data, mining multi-media data, mining social networks and graph data, mining spatial and temporal data, pre-processing and post-processing in data mining, robust and scalable statistical methods, security, privacy, and adversarial data mining, visual data mining, visual analytics, and data visualization.

Graph Mining: Laws, Tools, and Case Studies
D. Chakrabarti and C. Faloutsos
2012

Mining Heterogeneous Information Networks: Principles and Methodologies
Yizhou Sun and Jiawei Han
2012

Privacy in Social Networks
Elena Zheleva, Evimaria Terzi, and Lise Getoor
2012

Community Detection and Mining in Social Media
Lei Tang and Huan Liu
2010

Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions
Giovanni Seni and John F. Elder
2010

Modeling and Data Mining in Blogosphere
Nitin Agarwal and Huan Liu
2009

Probabilistic Approaches to Recommendations

Nicola Barbieri, Giuseppe Manco, and Ettore Ritacco

www.morganclaypool.com

ISBN: 9781627052573 paperback
ISBN: 9781627052580 ebook

DOI 10.2200/S00574ED1V01Y201403DMK009

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON DATA MINING AND KNOWLEDGE DISCOVERY

Lecture #9
Series Editors: Jiawei Han, *University of Illinois at Urbana-Champaign*
Lise Getoor, *University of Maryland*
Wei Wang, *University of North Carolina, Chapel Hill*
Johannes Gehrke, *Cornell University*
Robert Grossman, *University of Chicago*
Series ISSN
Print 2151-0067 Electronic 2151-0075

Probabilistic Approaches to Recommendations

Nicola Barbieri
Yahoo Labs, Barcelona, Spain

Giuseppe Manco
ICAR-CNR, Rende, Italy

Ettore Ritacco
ICAR-CNR, Rende, Italy

SYNTHESIS LECTURES ON DATA MINING AND KNOWLEDGE DISCOVERY #9

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

The importance of accurate recommender systems has been widely recognized by academia and industry, and recommendation is rapidly becoming one of the most successful applications of data mining and machine learning. Understanding and predicting the choices and preferences of users is a challenging task: real-world scenarios involve users behaving in complex situations, where prior beliefs, specific tendencies, and reciprocal influences jointly contribute to determining the preferences of users toward huge amounts of information, services, and products. Probabilistic modeling represents a robust formal mathematical framework to model these assumptions and study their effects in the recommendation process.

This book starts with a brief summary of the recommendation problem and its challenges and a review of some widely used techniques Next, we introduce and discuss probabilistic approaches for modeling preference data. We focus our attention on methods based on latent factors, such as mixture models, probabilistic matrix factorization, and topic models, for explicit and implicit preference data. These methods represent a significant advance in the research and technology of recommendation. The resulting models allow us to identify complex patterns in preference data, which can be exploited to predict future purchases effectively.

The extreme sparsity of preference data poses serious challenges to the modeling of user preferences, especially in the cases where few observations are available. Bayesian inference techniques elegantly address the need for regularization, and their integration with latent factor modeling helps to boost the performances of the basic techniques.

We summarize the strengths and weakness of several approaches by considering two different but related evaluation perspectives, namely, rating prediction and recommendation accuracy. Furthermore, we describe how probabilistic methods based on latent factors enable the exploitation of preference patterns in novel applications beyond rating prediction or recommendation accuracy.

We finally discuss the application of probabilistic techniques in two additional scenarios, characterized by the availability of side information besides preference data.

In summary, the book categorizes the myriad probabilistic approaches to recommendations and provides guidelines for their adoption in real-world situations.

KEYWORDS

recommender systems, probability, inference, prediction, learning, latent factor models, maximum likelihood, mixture models, topic modeling, matrix factorization, Bayesian modeling, cold start, social networks, influence, social contagion

This book is dedicated to our families.

To Irene, Nicola, Maddalena and Bella. Thanks for your patience and continued support.

To Caterina and Nino. Thanks for your love and encouragement throughout my life.

To Tiziana, Teodora and Francesco for their love.

Contents

Preface

Recommendation is a special form of information filtering that extends the traditional concept of search by modeling and understanding the personal preferences of users. The importance of accurate recommender systems has been widely recognized by both academic and industrial efforts in the last two decades and recommender systems are rapidly becoming one of the most successful applications of data-mining and machine-learning techniques.

Within this context, a rich body of research has been devoted to the study of probabilistic methods for the analysis of the preferences and behavior of users. Real-world scenarios involve users in complex situations, where prior beliefs, specific tendencies, and reciprocal influences jointly contribute to determining the preferences of users toward huge amounts of information, services, and products. Can we build simple yet sophisticated models to explain the way these preferences occur? How can we identify and model the latent causes that guide and influence the adoptions and preferences of users in complex systems? Probabilistic modeling represents a robust, formal mathematical framework upon which to model assumptions and study their effects in real-world scenarios.

In this book, we study and discuss the application of probabilistic modeling to preference data provided by users. We draw from recent research in the field and attempt to describe and abstract the process adopted by the research community in the last five years. The increasing popularity of recommender systems (due also to the advent of the Netflix prize) led to an explosion of research on generative approaches for modeling preference data. Within this scenario, an awareness has grown that probabilistic models offer more than traditional methods in terms of both accuracy and exploitation. Hence the need for a systematic study aimed at categorizing the myriad approaches and providing guidelines for their adoption in real-world situations.

We start by formally introducing the recommendation problem and summarizing well-established techniques for the generation of personalized recommendations. We present an overview of evaluation methods and open challenges in the recommendation process, and then focus on collaborative filtering approaches to recommendation.

The core of the book is the description of probabilistic approaches to recommendations. We concentrate on probabilistic models based on latent factors, which have proved to be extremely powerful and effective in modeling and identifying patterns within high-dimensional (and extremely sparse) preference data. The probabilistic framework provides a powerful tool for the analysis and characterization of complex relationships among users and items. Probabilistic models can enhance and strengthen traditional techniques as they offer some significant advantages. Notably, they can: be tuned to optimize any of several loss functions; optimize the likelihood of enabling the modeling of a distribution over rating values, which can be used to determine the

confidence of the model in providing a recommendation; finally, they offer the possibility of including prior knowledge in the generative process, thus allowing a more effective modeling of the underlying data distribution.

Rooted in these backgrounds, this book focuses on the problem of effectively adopting probabilistic models for modeling preference data. Our contributions can be summarized in two ways. First, we study the effectiveness of probabilistic techniques in generating accurate and personalized recommendations. Notably, the interplay of several different factors, such as the estimate of the likelihood that the user will select an item, or the predicted preference value, provides a fruitful degree of flexibility. Secondly, we show how probabilistic methods based on latent factor modeling can effectively capture interesting local and global patterns embedded in the high-dimensional preference data, thereby enabling their exploitation in novel applications beyond rating prediction.

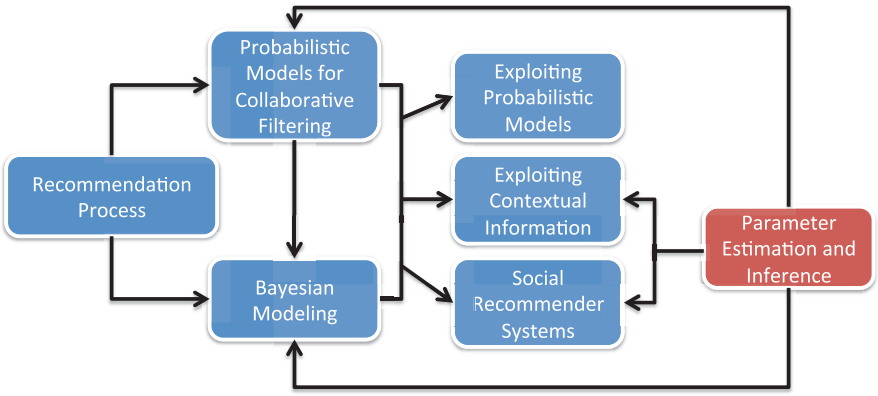
The effectiveness of probabilistic methods for recommendation has also been shown in other scenarios. The first concerns contextual information. Typically, preference data assumes a "bag-of-words" representation, i.e. the order of the items a user chooses can be neglected. This assumption allows us to concentrate on recurrent co-occurrence patterns and to disregard sequential information that may characterize the choice of the user. However, there are several real-world applications where contextual and side information play a crucial role. For example, content features can help characterize the preferential likelihood of an item in extremely dynamic content production systems. Also, there are some scenarios, such as web navigation logs or customer purchase history, where data can be "naturally" interpreted as sequences. Ignoring the intrinsic sequentiality of the data may result in poor modeling. By contrast, the focus should be on the context within which a current user acts and expresses preferences, i.e., the environment, characterized by side information, where the observations hold. The context enables a more refined modeling and, hence, more accurate recommendations.

Finally, the advent of social networking introduces new perspectives in recommendation. Understanding how the adoptions of new practices, ideas, beliefs, technologies and products can spread trough a population, driven by social influence, is a central issue for the social science and recommendation communities. Taking into account the modular structure of the underlying social network provides further important insights into the phenomena known as social contagion or information cascades. In particular, individuals tend to adopt the behavior of their social peers so that information propagation triggers "viral" phenomena, which propagate within connected clusters of people. Therefore, the study of social contagion is intrinsically connected to the recommendation problem.

In this book, the interested reader will find the mathematical tools that enable the understanding of the key aspects that characterize the main approaches for recommendation. This will allow both the understanding of the opportunities of the current literature and the development of ad-hoc methods for the solution of specific problems within the recommendation scenario.

The book is not self-comprehensive and a basic understanding probability and statistics, as well as machine learning and data mining methods, is needed.

The dependencies between chapters are shown in the following figure, where an arrow from one chapter to another indicates that the latter requires some knowledge of the first.



Within this structure, the appendix contains a primer on the basic inference and estimation tools used throughout the chapters and it should be considered as a reference tool.

Nicola Barbieri, Giuseppe Manco, and Ettore Ritacco
April 2014

CHAPTER 1

The Recommendation Process

1.1 INTRODUCTION

With the increasing volume of information, products, services, and resources (or, more generally, items) available on the Web, the role of *recommender systems (RS)* [162] and the importance of highly accurate recommendation techniques have become major concerns both in e-commerce and academic research. The goal of a RS is to provide users with recommendations that are non-trivial, useful to directly experience potentially interesting items, and that the user may not find on her own. Their exploitation in e-commerce enables a better interaction between the users and the system: RSs are widely adopted in different contexts, from music (Last.fm[1]) to books (Amazon[2]), movies (Netflix[3]), and news (Google News[4][49]), and they are quickly changing and reinventing the world of e-commerce [176].

A recommender can be considered as a "push" system that provides users with a personalized exploration of a wide catalog of possible choices. Search-based systems require a user to explicitly express a query specifying what she is looking for. By contrast, in recommendation, the query is implicit and corresponds to all past interactions of the user with the system. By collecting and analyzing past users' preferences in the form of explicit ratings or like/dislike products, the RSs provide the user with smart and personalized suggestions, typically in the form of "customers who bought this item also bought" or "customers who bought related items also bought." The goal of the service provider is not only to transform a regular user into a buyer, but also to make her browsing experience more comfortable, thus building a strong loyalty bond. This idea is better explained by Jeff Bezos, CEO of Amazon.com:

> *"If I have 3 million customers on the Web, I should have 3 million stores on the Web."*

The strategic importance of accurate recommendation techniques has motivated both academic and industrial research for over 15 years; this is witnessed by huge investments in the development of personalized and highly accurate recommendation approaches. In October 2006, Netflix, the leader in the American movie-rental market, promoted a competition, the *Netflix Prize* [5][27], with the goal of producing a 10% improvement on the prediction accuracy achieved by their internal recommender system, *Cinematch*. The competition lasted three years and involved several

[1] last.fm
[2] amazon.com
[3] netflix.com
[4] news.google.com
[5] netflixprize.com

research groups from all over the world, improving and inspiring a fruitful research. During this period, a huge number of recommendation approaches were proposed, creating a substantial advance in the literature, while simultaneously encouraging the emergence of new business models based on RSs.

This chapter is aimed at providing an overview of the recommendation scenario from a technical point of view. First, we provide a formal framework that specifies some notations used throughout the manuscript. The focus is the formal definition of a recommender system and the evaluation of its capability to provide adequate recommendations. Then we will introduce and discuss some well-known recommendation approaches, with an emphasis on collaborative filtering.

1.2 FORMAL FRAMEWORK

Recommendation is a form of information filtering that analyzes users' past preferences on a catalog of items to generate a personalized list of suggested items. In the following, we introduce some basic notations to model users, items, and their associated preferences.

Let $\mathcal{U} = \{u_1, \ldots, u_M\}$ be a set of M users and $\mathcal{I} = \{i_1, \ldots, i_N\}$ a set of N items. For notational convenience, in the following, we reserve letters u, v to denote users from $\mathcal{U}$ and letters i, j to indicate items from $\mathcal{I}$. Users and items can be characterized by means of specific properties, specified as atomic predicates in first-order logic. Predicates can be related with either discrete properties (such as Sex(u,male), Location(u,NYC), or Genre(i,comedy), Rating(i,PG) regarding the user u and the item i), or numeric (such as Age(u,23) or Length(i, 175)). Users' preferences can be represented as a $M \times N$ matrix $\mathbf{R}$, whose generic entry r_i^u denotes the preference value (i.e., the degree of appreciation) assigned by user u to item i. User preference data can be classified as *implicit* or *explicit*. Implicit data correspond to mere observations of co-occurrences of users and items, which can be recorded by analyzing clicks, users' web sessions, likes, or check-in. Hence, the generic entry r_i^u of the user-item rating matrix $\mathbf{R}$ is a binary value: $r_i^u = 0$ means that u has not yet experienced i, whereas by $r_i^u = 1$ we denote that user u has been observed to experience item i. By contrast, explicit data record the actual ratings explicitly expressed by individual users on the experienced items. Ratings are typically collected by questionnaires in which a user is asked to proactively evaluate the purchased/selected product on a given rating scale. Such feedback provides a user with a way to explicitly express their preferences. Implicit feedback is abundant but often unreliable, as the true users' evaluations are still hidden. Explicit feedback is generally more accurate and can be either positive or negative, while implicit feedback is always positive.

Generally, explicit ratings can be encoded as scores in a (totally-ordered) numeric domain $\mathcal{V}$, represented as a fixed rating scale that often includes a small number of interestingness levels. In such cases, for each pair (u, i), rating values r_i^u fall within a limited range $\mathcal{V} = \{0, \ldots, V\}$, where 0 represents an unknown rating and V is the maximum degree of preference. Notation $\bar{r}_{\mathbf{R}}$ denotes the average rating among all those ratings $r_i^u > 0$. The number of users M, as well as the number of items N, are very large (typically with $M >> N$) and, in real-world applications, the

rating matrix $\mathbf{R}$ is characterized by an exceptional sparseness, as individual users tend to rate a limited number of items. In the following, we denote by $\langle u, i \rangle$ the enumeration of all those dyads in $\mathbf{R}$ such that $r_i^u > 0$; analogously $\langle u, i, r \rangle$ represents an enumeration of all the explicit ratings. Again, specific properties in the form of first-order predicates can be associated with either $\langle u, i \rangle$ or $\langle u, i, r \rangle$. Example properties can specify, e.g., the time period Period($\langle u, i \rangle$, evening), or even the item experienced before Preceding($\langle u, i \rangle$, j). We denote the whole set of predicates relative to $\mathcal{U}, \mathcal{I}$ or $\mathcal{U} \times \mathcal{I} \times \mathcal{V}$ as $\mathcal{F}$.

The set of items rated by user u is denoted by $\mathcal{I}_{\mathbf{R}}(u) = \{i \in \mathcal{I} | \langle u, i \rangle \in \mathbf{R}\}$. Dually, $\mathcal{U}_{\mathbf{R}}(i) = \{u \in \mathcal{U} | \langle u, i \rangle \in \mathbf{R}\}$ is the set of all those users, who rated item i. With an abuse of notation, we use $\mathcal{U}(i)$ and $\mathcal{I}(u)$ when the rating matrix $\mathbf{R}$ is known from the context. Any user u with a rating history, i.e., such that $\mathcal{I}_{\mathbf{R}}(u) \neq \emptyset$, is an *active user*. Both $\mathcal{I}_{\mathbf{R}}(u)$ and $\mathcal{U}_{\mathbf{R}}(i)$ can be empty. This setting is known as *cold start* and it generally occurs whenever a new user or item is added to the underlying information system. Cold start is generally problematic in recommender systems, since these cannot provide suggestions for users or items in the absence of sufficient information.

The illustration in Fig 1.1 sketches a recommendation scenario with 10 users, 10 items and explicit preferences. The set of users who rated item i_2 is $\mathcal{U}(i_2) = \{u_1, u_4, u_8, u_{10}\}$. Also, the set of items rated by user u_2 is $\mathcal{I}(u_2) = \{i_3, i_5, i_7, i_9\}$. The rating value of user u_2 over item i_4, as well as the ratings from u_4 over i_1 and i_3, are unknown.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
u_1	2	5			3	1		2		
u_2			2		4		5		1	
u_3	1		4			5		1		
u_4		4			4		2		2	
u_5	1			2	3	3		5		
u_6			3	5	1	3			4	5
u_7	4			3	5			1	1	
u_8	3	4					1		3	
u_9	1				3	2			4	4
u_{10}		5				5			5	4

Figure 1.1: Example of users' preference matrix.

Given an active user u, the goal of a RS is to provide u with a recommendation list $\mathcal{L}_u \subseteq \mathcal{I}$, including unexperienced items (i.e., $\mathcal{L}_u \cap \mathcal{I}_{\mathbf{R}}(u) = \emptyset$) that are expected to be of her interest. This clearly involves predicting the interest of u toward unrated items: exploiting (implicit and/or explicit) information about users' past actions, the RS provides a scoring function $p_i^u : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$, which accurately estimates future preferences, and hence can be used to predict which are the most likely products to be purchased in the future. A general framework for the generation of $\mathcal{L}_u$, is encoded by algorithm 1. Here, a subset C of candidate items is chosen according to domain-specific criteria. For example, C may correspond to some items to promote, or even to

the whole domain $\mathcal{I}$. Then, each item i in C is scored by p_i^u, and the top-L items are selected for the recommendation list.

Algorithm 1 Generation of recommendation list.

Require: A number of candidate items, D (positive integer such that $D \leq N$)
Require: The size of the recommendation list, L (positive integer such that $L \leq D$)
1: Choose $\mathcal{C} \subseteq \mathcal{I}$, according to a business-specific criterion, such that $|\mathcal{C}| = D$ and $\mathcal{C} \cap \mathcal{I}_{\mathbf{R}}(u) = \emptyset$.
2: Associate each item $i \in \mathcal{C}$ with a score p_i^u, which represents the appreciation of u for i
3: Let $\mathcal{L}_u$ be the selection of the top L items in $\mathcal{C}$ with the highest values p_i^u
4: Return $\mathcal{L}_u$

1.2.1 EVALUATION

Different measures have been proposed in order to evaluate the accuracy of a RS (for a detailed survey see [79, 103]). As mentioned before, recommendations usually come in the form of a list of the L items that the user might like the most. Intuitively, an accuracy metric should measure how close the predicted list is to the actual preference list of a user or how close a predicted rating is to its actual value.

The adoption of a recommender system requires a prior evaluation of its ability to provide a positive impact on both a user's experience and a provider's revenues. Notably, the evaluation is accomplished by applying a protocol that delivers some objective measures stating the quality of the RS. These measures allow for a comparison among the several algorithmic and modeling alternatives. Two different strategies can be pursued, namely *offline* and *online* evaluation. Within online evaluation, a small percentage of users are asked to test the recommendation engine and to provide feedback about the main features. Online testing requires a careful design of the testing protocol, in order for the evaluation to be fair and the results to be statistically sound. This includes the selection of a representative subset of users and the features to be evaluated. Additionally, a direct involvement of actual users is not advisable, as it could negatively affect the experience of the test users: negative judgements can bias a user toward an under-evaluation of the capabilities of the recommendation engine.

By contrast, offline evaluation simulates the online process by employing simple statistical indices (either on synthesized or historical data) to measure the performance of the system. Indices make comparisons among alternative algorithms and schemes easier, and measure the effectiveness of a system at design time. Offline evaluation protocols and metrics usually rely on an evaluation framework where the rating matrix $\mathbf{R}$ is split into two matrices $\mathbf{T}$ and $\mathbf{S}$, such that the former is used to train a recommendation algorithm, while the latter is used for validation purposes, to measure the predictive abilities of the system. The latter can be measured according to different perspectives, discussed in the following.

Prediction accuracy. Historically, most CF techniques focused on the development of accurate techniques for rating prediction. The recommendation problem has been interpreted as a missing value prediction problem [174], in which, given an active user, the system is asked to predict her preference for a set of items. In this respect, the score p_i^u can be devised as a function of the prediction provided by the RS. Predictive accuracy metrics measure how close the predicted ratings are to the actual ratings. Let $\hat{r}_i^u$ denote the predicted rating on the dyad $\langle u, i \rangle$. Prediction accuracy is measured by means of statistical metrics that compare such predicted values with actual ratings. Widely adopted measures are:

- The **Mean Absolute Error (MAE)**, which measures the average absolute deviation between a predicted and a real rating. It is an intuitive metric, easy to compute and widely used in experimental studies:

$$MAE = \frac{1}{|\mathbf{S}|} \sum_{\langle u,i \rangle \in \mathbf{S}} |r_i^u - \hat{r}_i^u| \,. \tag{1.1}$$

- The **Mean Squared Error (MSE)** and **Root Mean Squared Error (RMSE)**. These measure the deviation of observed ratings from predicted values and, compared to MAE, emphasize large errors. They are defined as

$$MSE = \frac{1}{|\mathbf{S}|} \sum_{\langle u,i \rangle \in \mathbf{S}} \left(r_i^u - \hat{r}_i^u\right)^2 , \tag{1.2}$$

  and

$$RMSE = \sqrt{MSE}. \tag{1.3}$$

- The **Mean Prediction Error (MPE)**, expresses the percentage of predictions which differ from the actual rating values,

$$MPE = \frac{1}{|\mathbf{S}|} \sum_{\langle u,i \rangle \in \mathbf{S}} \mathbb{1}[\![r_i^u \neq \hat{r}_i^u]\!], \tag{1.4}$$

  where $\mathbb{1}[\![bexpr]\!]$ is the indicator function which returns 1 if the boolean expression *bexpr* holds and 0 otherwise. Notably, MPE resembles the classical accuracy measured on classification algorithms in machine learning and data mining. As such, it can be further refined into a confusion matrix denoting the misclassifications relative to each specific rating. Other more refined metrics relying on *ROC analysis* [56] can be devised, based on such a confusion matrix.

Recommendation Accuracy. The focus here is on the recommendation list $\mathcal{L}_u$, and we are interested in measuring the accuracy in building a list that matches the user's preferences. If we consider the problem from an information retrieval (IR) perspective [14], we can assume an explicit knowledge of u's actual preferences through a specific list $\mathcal{T}_u$. Classical IR measures can then be adapted to measure the correspondence between the proposed and the actual list.

Items in $\mathcal{T}_u$ represent the portion of the catalog that are likely to meet u's preferences. With implicit preferences, $\mathcal{T}_u$ can be any subset in $\mathcal{I}_\mathbf{S}$: the assumption here is that the user only selects relevant items. With explicit feedback, the relevance of an item can be further refined by focusing on those items in $\mathcal{I}_\mathbf{S}$ for which the evaluation is representative of high interest. For example, we can fix a threshold t (e.g., the average evaluation $\overline{r}_\mathbf{T}$), and then select $\mathcal{T}_u$ as a random subset of $\{i \in \mathcal{I}_\mathbf{S}(u) | r_i^u > t\}$.

Given $\mathcal{T}_u$ for each user u, the evaluation of a recommendation list built according to Algorithm 1 can be measured through the standard precision and recall measures. In particular, the quality of a system which generates recommendation lists of size L can be measured as

$$\begin{aligned} Recall(L) &= \frac{1}{M} \sum_{u \in \mathcal{U}} \frac{|\mathcal{L}_u \cap \mathcal{T}_u|}{|\mathcal{T}_u|}. \\ Precision(L) &= \frac{1}{M} \sum_{u \in \mathcal{U}} \frac{|\mathcal{L}_u \cap \mathcal{T}_u|}{|\mathcal{L}_u|}. \end{aligned} \tag{1.5}$$

Precision and recall are conflicting measures. For instance, by increasing the size of the recommendation list, recall is expected to increase but precision decreases. Typically, these measures are combined into the *F-score*, a single measure representing their harmonic mean and balancing both contributions:

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{1.6}$$

Recall and precision can also be exploited in a more refined evaluation protocol that relies on the notion of *hit rate*. We can devise a random process where the recommendation list is built by scoring a set made of both random elements and elements within $\mathcal{T}_u$. Then, hits represent the elements in $\mathcal{L}_u$ that also belong to $\mathcal{T}_u$. This testing protocol, called *user satisfaction* on the recommendation list, was proposed by Cremonesi et al. in [47] and is illustrated in Algorithm 2. Recall values relative to a user u can be adapted accordingly:

$$US\text{-}Recall(u, L) = \frac{\#hits}{|\mathcal{T}_u|}, \tag{1.7}$$

and global values can be obtained by averaging overall users. As for precision, there is a difficulty in considering *false positives*, which negatively influence the accuracy of the system. False positives represent the amount of spurious items, which are not relevant for the user and by contrast are scored high in the list. Clearly, the size L can affect the amount of such items, and hence, a suitable approximation for precision can be obtained by simply reweighting the recall to represent the percentage of relevant items relative to the recommendation list,

$$US\text{-}Precision(u, L) = \frac{\#hits}{L \cdot |\mathcal{T}_u|}. \tag{1.8}$$

More refined measures can be introduced that consider both the amount of spurious items and the position of the relevant item within the list.

Algorithm 2 User satisfaction of user u for item i.

Require: An item $i \in \mathcal{T}_u$
Require: A number of candidate items, D (positive integer such that $D \leq |\mathcal{I}|$)
Require: The size of the recommendation list, L (positive integer such that $L \leq D$)
1: Let $\mathcal{C}$ be a random subset of $\mathcal{I}$, with size D, whose elements have not been rated by u
2: Add i in $\mathcal{C}$
3: Assign to each item $i \in \mathcal{C}$ a score p_i^u, which represents the appreciation of u for i
4: Let $\mathcal{L}_u$ be the selection of the L items in $\mathcal{C}$ with the highest values p_i^u
5: **if** $i \in \mathcal{L}_u$ **then**
6: return a *hit*
7: **else**
8: return a *miss*
9: **end if**

Rank Accuracy. The generation of a recommendation list is based on the generation of a scoring function. In our basic framework, we assumed that users provide explicit feedback in terms of a scoring value in a given range. However, an explicit feedback can also consist of an explicit ranking of items for each user. That is, users can provide information on which items are better than others, according to their preferences.

True ranking is unrealistic with huge item catalogs. Nevertheless, it allows us to define ranking accuracy metrics that measure the adequacy of the RS in generating a personalized ordering of items that matches with the true user's ordering.

Typically, the predicted and observed orderings are compared by means of *Kendall*'s ($\mathcal{K}$) or *Spearman*'s (ρ) coefficients. Let τ_u be the full descendingly ordered ranking list for the user u and let $\hat{\tau}_u$ denote the predicted ranking list, both defined over the domain $\mathcal{I}_{\mathbf{S}}(u)$. Then:

$$\mathcal{K}(\tau_u, \hat{\tau}_u) = \frac{2\left(\sum_{i,j \in \mathcal{I}} \mathbb{1}[\![\tau_u(i) \prec \tau_u(j) \ \wedge \ \hat{\tau}_u(j) \prec \hat{\tau}_u(i)]\!]\right)}{N(N-1)} \tag{1.9}$$

$$\rho(\tau_u, \hat{\tau}_u) = \frac{\sum_{i \in \mathcal{I}} (\tau_u(i) - \bar{\tau}_u)(\hat{\tau}_u(i) - \bar{\hat{\tau}}_u)}{\sqrt{\sum_{i \in \mathcal{I}} (\tau_u(i) - \bar{\tau}_u)^2 \sum_{i \in \mathcal{I}} (\hat{\tau}_u(i) - \bar{\hat{\tau}}_u)^2}} \tag{1.10}$$

where $\tau(i)$ is the rank associated with the item i in the list τ, $\bar{\tau}$ is the average rank in the list τ, and $i \prec j$ is the comparison operator to denote if i is ranked ahead of j. The index $\mathcal{K}$, within the range $[-1, 1]$, measures the agreement between two rankings, which in this case are the real ranked list and the predicted one. If the predicted ranked list matches perfectly with the observed one, then $\mathcal{K} = 1$. By contrast, ρ ranges within $[-1, +1]$ and measures the correlation between two ranked variables. Once again, the perfect matching yields $\rho = 1$.

Other Evaluation Measures. The aforesaid accuracy metrics are important for offline evaluation and provide viable mathematical frameworks to compare different recommendation algorithms and techniques. However, they are not focused on evaluating the output of a recommender, as users really perceive it from a psychological standpoint. Several key dimensions, beyond accuracy, are gaining increasing importance in the evaluation of the quality of recommendations:

Novelty. This is closely related to the utility of a RS, which should recommend items the user is not already aware of. Recommending a new "Star Trek" movie to a Mr. Spock's fan is not a novel suggestion, since it is likely that the user is already aware of the movie.

Serendipity. This measures the extent to which the RS is able to suggest items that are both of interest to the user and *unexpected, surprising*. Intuitively, assuming that a list of "obvious" items to be recommended is available, serendipity can be measured as the ability of the RS to avoid the recommendations of such items.

Diversity. This measures the capability of suggesting "different" items. Most often, a real RS provides focused suggestions, which are similar to each other, e.g., books by the same author. Diversity should allow recommenders to include items that span, as much as possible, over all of a user's interests.

Coverage. This focuses on the domain of items which the RS is effectively able to recommend to users, and measures whether recommender systems are biased toward some specific subsets.

Limitations of Off-Line Evaluations. Evaluating recommendations is still among the biggest unsolved problems in RSs. Offline evaluation does not necessarily align with actual results: offline metrics like prediction/recall, RMSE, and $\mathcal{K}$ are based on snapshots of the past activities of users, and they cannot measure the real impact of recommendations on users' purchasing activities. The fact that a recommender matches past preferences does not necessarily mean that it is capable of influencing a user's behavior in the future. A strong issue is how to consider the unrated items. Do users dislike them? Haven't users experienced them yet? Do users know about their existence?

The limitations of off-line evaluation motivates the study of more sophisticated evaluation protocols and accurately controlled experiments. In general, a more effective evaluation should focus on how the user perceives recommendations and on measuring their helpfulness. Current research is investigating several aspects correlated with evaluation, such as the analysis of the *return of investment (ROI)* in improving prediction and recommendation metrics and the proposal of more convincing offline evaluation metrics.

1.2.2 MAIN CHALLENGES

There are a number of issues that the design of a recommendation algorithm is typically required to deal with. These are generally relative to the quality and the size of the data, and the security and privacy issues that a recommendation engine may breach. A brief review of these challenges is summarized next.

Sparsity and Cold Start. Recommendation systems in e-commerce have to deal with a huge number of items and users. Usually, even the most active users purchase only a limited fraction of the available items. The sparsity problem poses a real challenge to recommendation techniques. For example, the preference matrix might not provide enough information for particular users/items. This problem is known as *reduced coverage* [175], and can seriously affect the accuracy of recommendations. A wide range of methods have been proposed to deal with the sparsity problem. Many of them use dimensionality reduction techniques to produce a low-dimensional representation of the original customer-product space [175].

A related shortcoming is the inclusion of new items or new users. This problem is known as cold-start [177] and has been formalized before. The recommendation engine cannot provide suggestions to unknown users, or regarding unknown items. The cold-start new-user scenario can be faced by soliciting information about her tastes via *preference elicitation*. For example, a new user can be asked to rate a list of items in the registration phase. Selecting the most informative items to propose is a challenge in this context [158], as this selection is crucial in order to obtain an initial estimate of a user's preferences. Other methods focus on the definition of transitive properties between users via trust inferences [146], or by exploiting graph-theory [5].

More generally, cold start can be faced by means of hybrid approaches that consider both the rating matrix and additional information about items and/or users. Demographic information about the user [151] (such as age, sex, marital status) and item features (such as genre, rating, year of release) [154], can be used to infer relationships between the new user (or new item) and already registered users (or items in the catalog) for which the recommendation system already has enough information.

Scalability. Large user and item bases require robust computational resources. This is especially true in recommender systems associated with e-commerce or web services, like Amazon.com or Google News, which also require real-time response rates. The need to generate high-quality suggestions in a few milliseconds calls for the adoption of scalable methods and sophisticated data structures for data analysis, access, and indexing. Since the number of users is, typically, much larger than the number of items (even different orders of magnitude), scalable approaches focus on the analysis of the item-to-item relationships [174] or tend to model single users in communities [86]. Dimensionality reduction techniques [175], even in incremental versions [173], are used as well, as they allow us to discriminate between a learning phase, to be processed offline, and a recommendation phase, which can be performed online.

Item Churn. In dynamic systems, such as news recommenders, items are inserted and deleted quickly. The obsolescence of the model in such situations is a big issue, and it calls for the need to develop incremental model-maintenance techniques.

Privacy. Recommender systems can pose serious threats to individual privacy. Personal data, collected using the interactions of customers with the systems, is commonly sold when companies suffer bankruptcy [38] and, although recommendations are provided in an anonymous form,

aggregations of user preferences can be exploited to identify information about a particular user [157]. Sensitive information, like political and sexual orientations, were breached from the dataset made public for the Netflix prize [141].

Privacy-preserving techniques for recommendation include randomized perturbation methods [153] and user-community aggregations of preferences that do not expose individual data [38, 39]. Experimental studies have shown that accurate recommendations are still possible while preserving user privacy.

Security. Collaborative recommender systems can be exposed to malicious users willing to influence suggestions about items. An attacker can modify the original behavior of the systems by using a set of *attacker profiles* [198], corresponding to fictitious user identities. Three different attack intents have been identified [109]: the goal of an attacker is "pushing" a particular item by raising its predicted ratings; or "nuking" competitor products by lowering predicted ratings; and, in the worst case, damaging the whole recommender system. The works [109, 145, 171] study the robustness of collaborative recommendation, as well as countermeasures for making attacks ineffective. Recent studies [136, 198] have formalized the push/nuke attacks and, consequently, they propose classification techniques for detecting the attacking profile. The general form of an attack is composed of a *target item* (i.e., the item to push or nuke), a set of ratings over *selected items* (chosen for their similarity respect to the target item), and a set of ratings on *filler items*. Each attack can be characterized by considering how the attackers identified the selected items, what proportion of the remaining items they choose as filler items, and how they assign specific ratings to each set of items.

1.3 RECOMMENDATION AS INFORMATION FILTERING

The idea behind information filtering is that human interests and preferences are essentially correlated, thus, it is likely that a target user tends to prefer what other like-minded users have appreciated in the past. Hence, the basic approach to information filtering consists of collecting data about user interests and preferences to build reliable user profiles. These allow us to predict the interests of a target user based on the known preferences of similar users and, eventually, the provision of suitable recommendations of potentially relevant items. Recommendations generated by means of *filtering* techniques can be divided into three main categories.

- **Demographic filtering** approaches assume the possibility of partitioning the overall set of users into a number of classes with specific demographic features. Customization is achieved by exploiting manually-built, static rules that offer recommendations to the target user on the basis of the demographic features of the class she belongs to.
- **Content-based filtering** techniques learn a model of users' interests and preferences by assuming that item and user features are collected and stored within databases' information systems. The aim is to provide the target user with recommendations of unexperienced content items that are thematically similar to those she liked in the past.

- **Collaborative filtering** works by matching the target user with other users with similar preferences and exploits the latter to generate recommendations.

We briefly describe the first two information filtering techniques. Collaborative filtering approaches are the focus of the entire monograph and will be detailed in the next section.

1.3.1 DEMOGRAPHIC FILTERING

Demographic filtering partitions users in $\mathcal{U}$ into a certain number of disjoint classes $\mathcal{U}^{(1)}, \ldots, \mathcal{U}^{(C)}$ and then establishes relationships between items in $\mathcal{I}$ and the corresponding class(es) of interested users. Personal demographic data is basically exploited to group users into classes according to some partitioning criterion, such as the homogeneity of their purchasing history or lifestyle characteristics [108]. A set $\mathcal{P} = \{p^{(1)}, \ldots, p^{(C)}\}$ of handcrafted rules [121] is then exploited to specify the set of items recommendable to the users within the individual classes. A generic rule $p^{(c)}$ associates $\mathcal{U}^{(c)}$ with the corresponding subset $\mathcal{I}^{(c)}$, so that, for a given user u in $\mathcal{U}^{(c)}$, the (binary) prediction criterion

$$\hat{r}_i^u = \begin{cases} 1 & \text{if } i \in p^{(c)}(u) \\ 0 & \text{otherwise} \end{cases} \tag{1.11}$$

can be devised.

Several disadvantages limit the effectiveness of demographic filtering. The assumption behind recommender systems based on demographic filtering is that users within the same demographic class are interested in the same items. However, in practice, demographic classes are too broad and do not catch the actual differences in the interests, preferences, and expectations of the involved users. This results in too vague recommendations. The collection of demographic data raises privacy concerns that make users reluctant to provide personal information. Finally, considerable effort is required to manually specify and periodically update the set $\mathcal{P}$ of recommendation rules.

1.3.2 CONTENT-BASED FILTERING

In content-based filtering, each item is described by a set of keywords or attributes and each user is associated with a user profile that summarizes the features of the products she liked/purchased in the past. Content-based recommendations are performed by ranking items according to a similarity function that takes into account the profile of the current user [110, 150]. We assume a scenario where items can be annotated with textual features. Hence, in its most basic form, content-based filtering is essentially an application of information retrieval techniques to the domain of recommendation.

In particular, we assume that $\mathcal{F}$ embodies a set $\{f_1, \ldots, f_q\}$ of common descriptive features for the items in $\mathcal{I}$, which summarize and index such textual content. These features can be, for example, keywords from the textual content associated with an item [14, 170], e.g., keywords

extracted from the plot of the movie. Features can then be exploited to model items into a *vector representation*„ that allows us to project them into a multidimensional Euclidean space. A feature vector $\mathbf{w}_i \in \mathbb{R}^q$ is associated with an item $i \in \mathcal{I}$, and $w_{i,f}$ represents the relevance of feature f for the considered item. The weight $w_{i,f}$ can be either binary or numeric, depending on the adopted representation format[14]. For example, TF/IDF [14, 170] is an established weighting scheme that aims to penalize those features that frequently occur throughout $\mathcal{I}$, since these do not allow us to distinguish any particular item from the others.

Feature values can be exploited to predict the interest of the user for an unexperienced item. In particular, machine-learning approaches based on classification can be exploited by assuming that the previous choices of a user u in the preference matrix $\mathbf{R}$ can be partitioned into disjoint sets of positive and negative examples of her interests, i.e., $\mathcal{I}(u) = \mathcal{I}^+(u) \cup \mathcal{I}^-(u)$. For example, $\mathcal{I}^+(u)$ may consist of items with ratings above the average rating $\bar{r}_{\mathbf{R}}$ and, similarly, characterize $\mathcal{I}^-(u)$ as the set of items with ratings below the average. Partitioning the items allows us to build a base set upon which to train a classifier.

Besides classification, content features can be also exploited to strengthen collaborative filtering approaches, described in the next section. For example, items can be deemed similar if a suitable metric for measuring the proximity of their contents can be devised. Content similarity has been studied in several scenarios, and the most common used metrics are the following.

- **Minkowski distance** [93] generalizes the notion of distance between two points in an Euclidean space. Given the feature vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ respectively associated to the items i and j, it is defined as

$$dist_p^M(i,j) = \left(\sum_{l=1}^{q} |w_{i,l} - w_{j,l}|^p \right)^{\frac{1}{p}}.$$

  Both the Euclidean and Manhattan distances originate from the above definition, for $p = 2$ and $p = 1$, respectively.

- **Cosine similarity** provides an alternative comparison for comparing items in $\mathcal{I}$. It measures the similarity of any two items in terms of the angle between their corresponding feature vectors $\mathbf{w}_i$ and $\mathbf{w}_j$:

$$sim^c(i,j) = \frac{\mathbf{w}_i^T \cdot \mathbf{w}_j}{\|\mathbf{w}_i\|_2 \cdot \|\mathbf{w}_j\|_2}.$$

  Due to its ease of interpretation and effectiveness in dealing with sparse feature vectors [52], cosine similarity is a commonly used measure.

- **Jaccard similarity** [189], originally conceived for boolean representations, tends to measure how common features tend to be predominant in the whole set of features:

$$sim^J(i,j) = \frac{\mathbf{w}_i^T \cdot \mathbf{w}_j}{\mathbf{w}_i^T \cdot \mathbf{w}_i + \mathbf{w}_j^T \cdot \mathbf{w}_j - \mathbf{w}_i^T \cdot \mathbf{w}_j}.$$

Jaccard similarity exhibits aspects of both the Euclidean and cosine measures. In particular, it tends to behave like the former (resp. the latter) if $sim^J(i,j) \to 1$ (resp. if $sim^J(i,j) \to 0$).

Choosing among the above measures is essentially a matter of how sparsely items map in the feature space. [189] provides an extensive comparison of these measures and proposes some guidelines for their adoption.

1.4 COLLABORATIVE FILTERING

State-of-the-art recommendation methods have been largely approached from a *Collaborative Filtering (CF)* perspective, which essentially consists of the posterior analysis of past interactions between users and items, aimed at identifying suitable preference patterns in users' preference data. CF techniques aim at predicting the interest of users on given items, based exclusively on previously observed preference. The assumption is that *users who adopted the same behavior in the past will tend to agree also in the future*. The main advantage of using CF techniques is their simplicity: only past ratings are used in the learning process, and no further information, like demographic data or item descriptions, is needed. This solves some of the main drawbacks of content-based and demographic approaches [114, 152].

- No personal information about users is needed; this guarantees a higher level of privacy.
- CF approaches are more general and re-usable in different contexts, while content-based techniques require the specification of a complete profile (a set of features) for each user/item.
- Content-based techniques provide the user with a list of products with features "similar" to the ones that she experienced in the past. This approach may imply the recommendations of redundant items and the lack of novelty.
- The effectiveness of recommendations increases as the user provides more feedback.

It is important to stress here that the term "collaborative" refers to the capability of summarizing the experiences on multiple users and items. There are other naive methods that still rely on the preference matrix, but they only focus on individual entries and ignore their collaborative features. Simple baselines are the item and user averages,

$$itemAvg(i) \equiv \bar{r}_i = \frac{1}{|\mathcal{U}(i)|} \sum_{u \in \mathcal{U}(i)} r_i^u \tag{1.12}$$

$$userAvg(u) \equiv \bar{r}^u = \frac{1}{|\mathcal{I}(u)|} \sum_{i \in \mathcal{I}(u)} r_i^u, \tag{1.13}$$

or more complex combinations, such as:

$$doubleCentering(u,i) = \alpha\ \bar{r}_i + (1-\alpha)\ \bar{r}_u, \tag{1.14}$$

where $0 \leq \alpha \leq 1$ is a parameter that can be tuned to minimize the error in prediction. More sophisticated baselines can be achieved by combining other components. In the following, we shall refer to b_i^u as any of these baseline models.

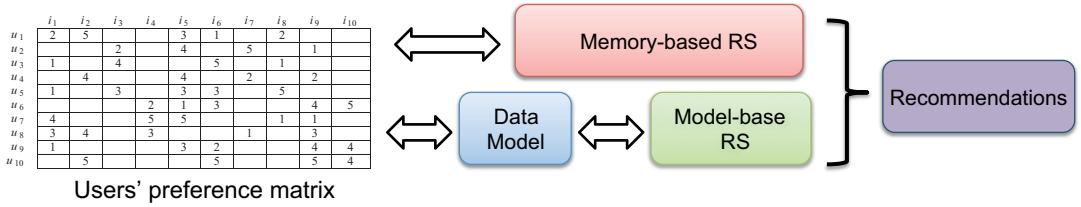


Figure 1.2: Memory-based vs. model-based approaches.

Collaborative filtering approaches can be classified in two categories [36], namely *memory-based* and *model-based* methods. Figure 1.2 provides a sketch of the differences between them. Both categories rely on the preference matrix. However, memory-based methods infer the preference of the active user for an item by using a database of previously recorded preferences. Among memory-based approaches, a prominent role is played by neighborhood-based methods, which are based on the definition of similarity between pairs of users/items. By contrast, model-based approaches operate in two phases: in the off-line phase, the rating matrix is used to learn a compact personalization model for each user; then, the model is used in an on-line phase to predict the degree of interest of the user on candidate items.

Memory-based approaches are intuitive, as they directly transform stored preference data into predictions. The drawback is that they need access to the whole dataset to make recommendations, and, thus, they require specific indexing techniques, especially when the size of the data increases. On the other hand, model-based approaches require access to only a compact representation of the data, but the recommendation provided to the user may not be easily interpretable. A fundamental distinction also relies on the kind of relationships among users and items that they are able to exploit. Neighborhood models are effective at detecting *strong but local relationships*, as they explicitly model local similarities. Model-based approaches typically employ dimensionality reduction techniques, and hence focus on the estimation of *weak but global relationships*. Probabilistic methods, which are the focus of this manuscript, represent a refinement of the model-based approach, which relies on probabilistic modeling both in the learning phase and in the prediction phase.

In the next sections, we present a brief review of the most-used CF approaches for explicit preference data. The remainder of the manuscript analyses probabilistic methods in detail.

1.4.1 NEIGHBORHOOD-BASED APPROACHES

The idea behind *neighborhood-based approaches* [78, 174] is that similar users share common preferences, and, hence, the predicted rating on a pair $\langle u, i \rangle$ can be generated by selecting the most-similar users to u. Similar users are called *neighbors*, and their preferences on i can be aggregated and combined to produce the prediction. In a real-life scenario, this would correspond to asking friends for their opinions before purchasing a new product.

We consider here the *K-nearest-neighbors (K-NN)* approach. Within this framework, the rating prediction $\hat{r}_i^u$ is computed following simple steps: (i) a similarity function allows us to specify the degree of similarity of each a of users, thus enabling the identification of the K users most-similar to u; (ii) the rating prediction is computed as the average of the ratings given by the neighbors on the same item, weighted by the similarity coefficients. Formally, by denoting with $s_{u,v}$ the similarity between u and v and by $\mathcal{N}^K(u)$ the K most-similar neighbors of u, we have

$$\hat{r}_i^u = \frac{\sum_{v \in \mathcal{N}^K(u)} s_{u,v} \cdot r_i^v}{\sum_{v \in \mathcal{N}^K(u)} s_{u,v}}. \tag{1.15}$$

Dually, one can consider an *item-based* approach [174]: The predicted rating for the pair $\langle u, i \rangle$ can be computed by aggregating the ratings given by u on the K most-similar items to i. The underlying assumption is that the user might prefer items more similar to the ones he liked before, because they share similar features. Formally,

$$\hat{r}_i^u = \frac{\sum_{j \in \mathcal{N}^K(i;u)} s_{i,j} \cdot r_j^u}{\sum_{j \in \mathcal{N}^K(i;u)} s_{i,j}}, \tag{1.16}$$

where $s_{i,j}$ is the similarity coefficient between i and j, and $\mathcal{N}^K(i;u)$ is the set of the K items evaluated by u, which are most similar to i.

Similarity coefficients play a central role, which is two-fold: they are used to identify the neighbors, and they also act as weights in the prediction phase. In addition to the measures presented in Section 1.3.2, we can alternatively express the similarity by looking at the rating matrix. In particular, two items (resp. users) can be deemed similar if the ratings they obtained (resp. provided) are similar.

Let $\mathcal{U}_{\mathbf{R}}(i, j)$ denote the set of users who provided a rating for both i and j, i.e., $\mathcal{U}_{\mathbf{R}}(i, j) = \mathcal{U}_{\mathbf{R}}(i) \cap \mathcal{U}_{\mathbf{R}}(j)$. Two standard definitions for the similarity coefficients are the *Pearson Correlation* or the *Adjusted Cosine*. The latter is an adaptation of the cosine similarity shown in Section 1.3.2 [174]:

$$Pearson(i, j) \triangleq \frac{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i,j)} \left(r_i^u - \overline{r}_i\right) \cdot \left(r_j^u - \overline{r}_j\right)}{\sqrt{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i,j)} \left(r_i^u - \overline{r}_i\right)^2} \sqrt{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i,j)} \left(r_j^u - \overline{r}_j\right)^2}};$$

$$AdjCosine(i,j) \triangleq \frac{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i,j)} \left(r_i^u - \overline{r}_u\right) \cdot \left(r_j^u - \overline{r}_u\right)}{\sqrt{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i,j)} \left(r_i^u - \overline{r}_u\right)^2} \sqrt{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i,j)} \left(r_j^u - \overline{r}_u\right)^2}}.$$

The basic prediction Equations 1.15 and 1.16 can be extended to include unbiased adjustments. For example, the adoption of a composite baseline b_i^u allows us to tune the predictions for specific users:

$$\hat{r}_i^u = b_i^u + \frac{\sum_{j \in \mathcal{N}^K(i;u)} s_{i,j} \cdot (r_j^u - b_j^u)}{\sum_{j \in \mathcal{N}^K(i;u)} s_{i,j}}.$$

Also, it is possible to generalize the weighting scheme in the equations. In particular, within Equation 1.16, the term $s_{i,j} / \sum_{j \in \mathcal{N}^K(i;u)} s_{i,j}$ represents a weight associated with rating r_j^u, and it is fixed. We can devise a more general formulation of the prediction function, as

$$\hat{r}_i^u = \sum_{j \in \mathcal{N}^K(i;u)} w_{i,j} \cdot r_j^u. \tag{1.17}$$

Here, the term $w_{i,j}$ represents a weight associated with the pair (i,j), to be estimated. The *neighborhood relationship model* [23, 25] provides an approach to compute them as the solution of the optimization problem:

$$\min \sum_{v \in \mathcal{U}_{\mathbf{R}}(i)} \left(r_i^v - \sum_{j \in \mathcal{N}^K(i;u,v)} w_{i,j} \cdot r_j^v \right)^2$$
$$s.t. \qquad w_{i,j} \geq 0 \; \sum_j w_{i,j} = 1 \; \forall i,j \in \mathcal{I}.$$

Here, the set $\mathcal{N}^K(i;u,v)$ is defined as the K items most similar to i, which are evaluated both by u and v.

The K-NN approach is characterized by a relatively lightweight learning phase: we only need to collect, for each user/item, a sufficient number of events that allow the computation of pairwise similarities. Such similarities can be directly computed online during the prediction phase. A big issue, however, is the computation of the similarity coefficients for each user/item pair. When the number of users and/or items is huge, computing such coefficients can be extremely impractical. To alleviate this problems, indexing methods are needed, which makes the search for neighbors more efficient. [205] contains a detailed survey on the subject.

1.4.2 LATENT FACTOR MODELS

The assumption behind *latent factor models* is that the overall preference of the user can be decomposed on a set of contributes that represent and weight the interaction between her tastes

and the target item on a set of features. This approach has been widely adopted in information retrieval. For example, the *latent semantic indexing (LSI)* [50] is a dimensionality reduction technique that assumes a latent structure in word usage across documents. LSI employes the *singular value decomposition* to represent terms and documents in the features space: some of these feature components are very small and may be ignored, obtaining an approximate model. Given a $M \times N$ matrix $\mathbf{A}$ with rank r, the singular value decomposition of $\mathbf{A}$, denoted by $SVD(\mathbf{A})$ (see Figure 1.3) , is defined as:

$$SVD(\mathbf{A}) = \mathbf{U} \times \Sigma \times \mathbf{V}^T, \tag{1.18}$$

where:

- $\mathbf{U}$ is an $M \times M$ unitary matrix; the first r columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^T$ (*left singular vectors* of $\mathbf{A}$);
- $\mathbf{V}$ is an $N \times N$ unitary matrix; the first r columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{A}^T\mathbf{A}$ (*right singular vectors* of $\mathbf{A}$);
- $\boldsymbol{\Sigma}$ is a $M \times N$ diagonal matrix with only r nonzero values, such that: $\boldsymbol{\Sigma} = diag\{\sigma_1, \cdots, \sigma_n\}$, $\sigma_i \geq 0 \; \forall \; 1 \leq i < n$, $\sigma_i \geq \sigma_{i+1}$, $\sigma_j = 0 \; \forall \; j \geq r+1$;
- $\{\sigma_1, \cdots, \sigma_r\}$ are the nonnegative square root of the eigenvalues of $\mathbf{A}^T\mathbf{A}$ and are called *singular values* of $\mathbf{A}$.

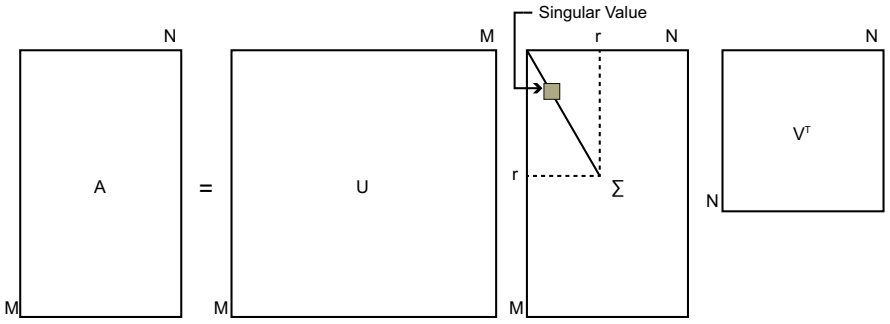


Figure 1.3: Singular value decomposition.

SVD has an important property: it provides the best low-rank linear approximation of the original matrix $\mathbf{A}$. Given a number $k \leq r$, called *dimension* of the decomposition, the matrix $\mathbf{A}_k = \sum_{i=1}^{k} u_i \sigma_i v_i^T$ minimizes the *Frobenius norm* $\|\mathbf{A} - \mathbf{A}_k\|_F$ over all rank-k matrices. Therefore, focusing only on the first k singular values of $\boldsymbol{\Sigma}$ and reducing the matrices $\mathbf{U}$ and $\mathbf{V}$, the original matrix can be approximated using $\mathbf{A}_k$:

$$\mathbf{A} \approx \mathbf{A}_k = \mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}_k^T, \tag{1.19}$$

where $\mathbf{U}_k$ is obtained by removing $(M-k)$ columns from the matrix $\mathbf{U}$ and $\mathbf{V}_k^T$ is produced by removing $(N-k)$ rows from the matrix V. An example of this approximation is given in Figure 1.4. Considering the text analysis case, *LSI* factorizes the original term-document matrix

$$A \approx U_k \cdot \Sigma_k \cdot V_k^T$$

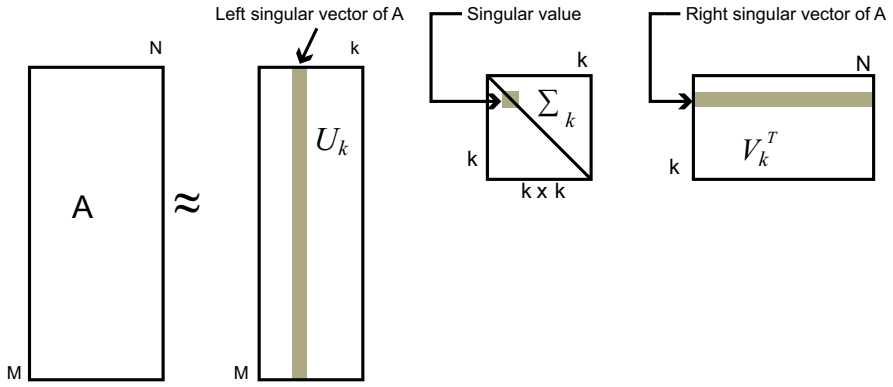


Figure 1.4: k-rank approximation of $\mathbf{A}$.

into the product of three matrices, which reflect the relationships between each single term and document in the k features-space, where k is the number of considered features. The derived matrix $\mathbf{A}_k$ is not the exact factorization of $\mathbf{A}$: the procedure of selecting only the k largest single values captures the underlying structure of $\mathbf{A}$, removing the noise at the same time [29]. Menon and Elkan in [132] provide a comparative study of the several methods for approximating the decomposition in the case of large matrices.

Several works have studied the application of SVD in recommender systems [30, 175]. A low-rank approximation provides a *low-dimensional* representation of the original *high-dimensional* rating matrix $\mathbf{R}$, thereby disclosing the hidden relationships between users and products that could be used to infer a user's preference on the considered item. If we consider a scenario involving ratings given by users on a set of movies, then we can provide a high-level interpretation of the rows of both $\mathbf{U}_k$ and $\mathbf{V}_k$. Intuitively, the row vector $\mathbf{U}_u \in \mathbb{R}^k$ maps a given user into a k-dimensional space, representing the underlying factors that influence each user's choice. By analogy, the row vector $\mathbf{V}_i \in \mathbb{R}^k$ maps a movie i into the same k-dimensional space. An example of what hidden factors might represent in this scenario is given by the movie genres. Assuming the existence of a limited number of different such genres (action, romance, comedy, etc.), the rating can be influenced by the user's preference on some genres and by the adherence of a movie to those genres. Figure 1.5 depicts this example scenario by exploiting three hidden factors.

With an abuse of notation, in this section we will use a simplified but equivalent formalization for the SVD, in which the original matrix is approximated as the product of two component

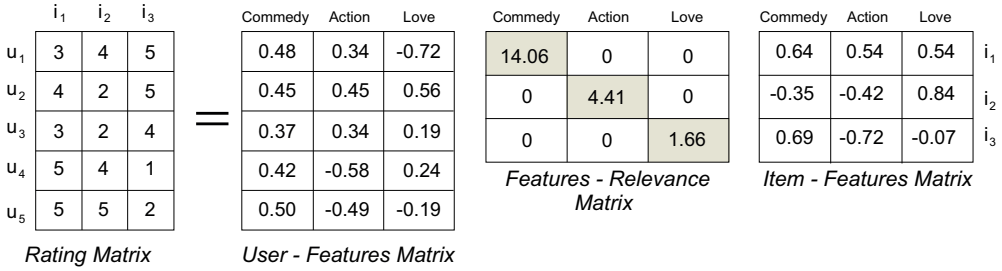


Figure 1.5: Example of the application of SVD decomposition.

matrices with K features:

$$\mathbf{R} \approx \left(\mathbf{U}_K \sqrt{\Sigma}_K^T\right)\left(\sqrt{\Sigma}_K \mathbf{V}_K^T\right) = \mathbf{U} \cdot \mathbf{V}, \tag{1.20}$$

where $\mathbf{U}$ is a $M \times K$ matrix and $\mathbf{V}$ is a $K \times N$. Intuitively, each user's preference on an item matrix is decomposed as the product of the dimensional projection of the users and items into the K-dimensional feature space:

$$\hat{r}_i^u = \sum_{k=1}^{K} \mathbf{U}_{u,k} \cdot \mathbf{V}_{k,i}. \tag{1.21}$$

The direct application of standard SVD to the context of factorizing user preferences poses some issues due to the extreme sparsity of the data. In fact, SVD requires a completely specified matrix, where all of entries are fit. In a CF scenario, missing data, i.e., unobserved user-item pairs which represent that the user did not purchase the item, can be interpreted in several ways. For instance, the user could already own the product and this purchase was not recorded on the system or he could simply not be aware of it. In other words, "the absence of evidence is not evidence of absence" and missing data requires special treatment that standard SVD does not perform. In fact, by treating as zero the preference value corresponding to unobserved user-item pairs, and applying SVD on the sparse preference matrix, the resulting model is biased toward producing low scores for items that a user has not adopted before, which may not be an accurate assumption.

To address this issue, it is convenient to minimize the prediction/reconstruction error by focusing exclusively on observed entries. Given the number of features K, and assuming that $\mathbf{U}$ and $\mathbf{V}$ represent a low-rank approximation of the original rating matrix $\mathbf{R}$, we can estimate the feature matrices by solving this optimization problem:

$$(\mathbf{U}, \mathbf{V}) = \underset{\mathbf{U},\mathbf{V}}{\operatorname{argmin}} \left[\sum_{(u,i) \in \mathcal{T}} \left(r_i^u - \sum_{k=1}^{K} \mathbf{U}_{u,k} \mathbf{V}_{k,i} \right)^2 \right]. \tag{1.22}$$

This optimization problem has been extensively studied, both theoretically [185] and practically [60]. For example, an incremental procedure to minimize the error of the model on observed

ratings, based on gradient descent, has been proposed in [60]. This was one of the major contributions achieved during the Netflix Prize. The feature matrices are randomly initialized and updated as follows:

$$\mathbf{U}'_{u,k} = \mathbf{U}_{u,k} + \eta \left(2e_{u,i} \cdot \mathbf{V}_{k,i}\right) \qquad \mathbf{V}'_{k,i} = \mathbf{V}_{k,i} + \eta \left(2e_{u,i} \cdot \mathbf{U}_{u,k}\right), \tag{1.23}$$

where $e_{u,i} = \hat{r}_i^u - r_i^u$ is the prediction error on the pair $\langle u, i \rangle$ and η is the learning rate.

The optimization problem can be further refined by constraining $\mathbf{U}$ and $\mathbf{V}$: for example, by forcing non-negativity and sparseness [88, 112]. Particular attention has been devoted to the problem of capacity control and overfitting prevention. In a collaborative prediction setting, only some of the entries of $\mathbf{R}$ are observed. As a consequence, the generalization capabilities of the model can be compromised by the learning procedure, which can be trapped into local minima and, in particular, can be influenced by extreme values. For example, some users or items could be associated with too few observations. *Regularization* in this case aims at balancing the values of the matrices by shrinking them to more controlled values. [160, 186] suggested a formulation of this regularization termed *Maximum Margin Matrix Factorization* (MMMF). Roughly, MMMF constrains the norms of $\mathbf{U}$ and $\mathbf{V}$ to a bounded value. This corresponds to constraining the overall "strength" of the factors, rather than their number. That is, a potentially large number of factors is allowed, but only a few of them are allowed to be very important. For example, "when modeling movie ratings, there might be a very strong factor corresponding to the amount of violence in the movie, slightly weaker factors corresponding to its comic and dramatic value, and additional factors of decaying importance corresponding to more subtle features such as the magnificence of the scenery and appeal of the musical score" [160].

Mathematically, the regularization constraints can be specified within the optimization function,

$$(\mathbf{U}, \mathbf{V}) = \underset{\mathbf{U},\mathbf{V}}{\operatorname{argmin}} \left[\sum_{(u,i) \in \mathcal{T}} \left(r_i^u - \sum_{k=1}^{K} \mathbf{U}_{u,k} \mathbf{V}_{k,i} \right)^2 + \lambda_U tr(\mathbf{U}^T \mathbf{U}) + \lambda_V tr(\mathbf{V}^T \mathbf{V}) \right], \tag{1.24}$$

where λ_U and λ_V are regularization coefficients and $tr(A)$ denotes the trace of the square matrix A. The above reformulation of the problem yields the new update rules that take into account the regularization:

$$\begin{aligned} \mathbf{U}'_{u,k} &= \mathbf{U}_{u,k} + \eta \left(2e_{u,i} \cdot \mathbf{V}_{k,i} - \lambda_U \cdot \mathbf{U}_{u,k}\right), \\ \mathbf{V}'_{k,i} &= \mathbf{V}_{k,i} + \eta \left(2e_{u,i} \cdot \mathbf{U}'_{u,k} - \lambda_V \cdot \mathbf{V}_{k,i}\right). \end{aligned} \tag{1.25}$$

Regularization is extremely important both theoretically and practically, and has several related aspects. For example, the regularization terms can be adapted based on the popularity of users and/or items. For instance, estimation for a user who has seen very few movies will likely suffer from overfitting unless heavily regularized. Likewise, the number of ratings per movie varies widely and the regularization should take this into account. [197] further refines the optimization

of Equation 1.24 by taking these aspects into account. In practice, the regularization coefficients λ_U and λ_V should be adapted to users and items, for example, by taking into account the component number of preference observation for user/item as a weighting component in the objective function. We will discuss the probabilistic interpretation of this regularization in detail in Chapter 3.

Several variations of the basic prediction rule 1.21 have been proposed. For example, the basic model can be refined by combining the baseline model with the SVD prediction [148]:

$$\hat{r}_i^u = b_i^u + \sum_{k=1}^{K} \mathbf{U}_{u,k} \cdot \mathbf{V}_{k,i}.$$

An alternative formulation reduces the number of parameters to be learned by relating the user matrix $\mathbf{U}_u$ to all the items preferred by u:

$$\mathbf{U}_{u,k} \propto \sum_{i \in \mathcal{I}(u)} \mathbf{V}_{k,i}.$$

Koren [106] further extends this model by considering both a free-factor contribution and a constrained contribution. The resulting *SVD++* model thus proposes the prediction rule:

$$\hat{r}_i^u = b_i^u + \sum_{k=1}^{K} \left(\mathbf{U}_{u,k} + \alpha \sum_{i \in \mathcal{I}(u)} \mathbf{V}_{k,i} \right) \cdot \mathbf{V}_{k,i}.$$

The gradient-descent algorithm can be tuned accordingly by plugging the above equation into Equation 1.22.

In addition to these approaches, the general task of matrix factorization for recommender systems has been widely explored and is surveyed in [107, 133]. The performance of latent factor models based on the SVD decomposition strongly depends on the number of features and the structure of the model, characterized by the presence of bias and baseline contributions. The optimization procedure used in the learning phase also plays an important role: learning can be incremental (one feature at the time) or in batch (all features are updated during the same iteration). Incremental learning usually achieves better performances at the cost of higher learning time.

1.4.3 BASELINE MODELS AND COLLABORATIVE FILTERING

We conclude this section by mentioning the relationships between the baseline methods and collaborative filtering. We discussed how the baseline methods still rely on the preference matrix, but they only focus on individual entries and ignore the collaborative nature of the data.

Besides their predictive capabilities, baselines can be exploited to refine collaborative methods, and we have seen some examples of this throughout this section. Alternatively, baselines can

be used for preprocessing the data prior to the application of other techniques. The *Global Effects Model* has been introduced by *Bell and Koren* in the context of the Netflix Prize [24], and it is considered an extremely effective technique. The key idea is to adjust standard collaborative filtering algorithms with simple models that identify systematic tendencies in rating data, called *global effects*. For example, some users might tend to assign higher or lower ratings to items with respect to their average rating (*user effect*), while some items tend to receive higher rating values than others (*item effect*). Other effects can involve the time dimension: for example, temporal patterns can cause user's ratings to increase or decrease in specific periods of time, and, similarly, ratings associated with some movies can decay soon after their release. Stepwise estimation and removal of the global effects provides a substantial advantage in terms of reduction of noise. As shown in [24], adopting this pre-processing strategy enables the identification of more precise models, which are robust to the biases introduced by the global effects.