

UNIVERSITÀ DELLA CALABRIA



Dipartimento di ELETTRONICA,
INFORMATICA E SISTEMISTICA

UNIVERSITÀ DELLA CALABRIA

Dipartimento di Elettronica,
Informatica e Sistemistica

Dottorato di Ricerca in
Ingegneria dei Sistemi e Informatica
XXVciclo

Tesi di Dottorato

Probabilistic Approaches to Recommendations

Nicola Barbieri

D.E.I.S., Novembre 2012
Settore Scientifico Disciplinare: ING-INF/05

UNIVERSITÀ DELLA CALABRIA

Dipartimento di Elettronica,
Informatica e Sistemistica

Dottorato di Ricerca in
Ingegneria dei Sistemi e Informatica
XXV ciclo

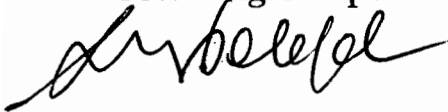
Tesi di Dottorato

Probabilistic Approaches to Recommendations

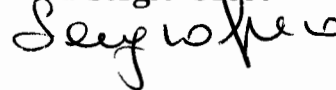
Nicola Barbieri



Coordinatore
Prof. Luigi Palopoli



Supervisor
Prof. Sergio Greco



Dr. Giuseppe Manco



DEIS

DEIS- DIPARTIMENTO DI ELETTRONICA, INFORMATICA E SISTEMISTICA
Novembre

Settore Scientifico Disciplinare: ING-INF/05

Preface

The design of accurate recommender systems is rapidly becoming one of the most successful application of data mining and machine learning techniques. Recommendation is a special form of information filtering, which extends the traditional concept of search, by modeling and understanding personal users' preference. The importance of an accurate Recommender System is widely witnessed by both academic and industrial efforts in the last two decades. In this thesis, we are going to present and discuss the application of novel probabilistic approaches for the modeling of users' preference data. We start our treatment by formally introducing the recommendation problem and summarizing the state of the art techniques for the generation of personalized recommendations. This analysis proves the effectiveness of probabilistic models based on latent factors in modeling and identifying patterns within the high dimensional (and exceptionally sparse) preference data. The probabilistic framework provides a powerful tool for the analysis and characterization of users and items, as it allows the identification of similar users and items which can be gathered in user communities and item categories.

Rooted on these backgrounds, our work focuses then on the problem of effectively adopting probabilistic models to preference data. Our contributions can be summarized in three directions. First, we study the effectiveness of the probabilistic techniques in generating accurate and personalized recommendation lists. Probabilistic techniques provide several kind of item ranking function which can be used to produce the recommendation list. This rankings focus on the estimate of the item selection, or on the predicted preference of each pair user,item or on a combination of these two, that we named *item selection and relevance*. The latter, which is based on the idea of predicting the chances that a user will "play and enjoy" a given content, is the best performer in terms of recommendation accuracy, and we design two probabilistic methods for the direct estimation of this function from data. Finally, starting from the observation that the high dimensional preference data may exhibit both global and local patterns, we propose and a probabilistic hierarchical model, in which user communities and item categories depend on each other.

VI Preface

This means that different user communities can exhibit different preference patterns on different item categories, where the identification of the latter depend on the considered community. This approach better models relationships between users and items and improve existing techniques both in terms of prediction and recommendation accuracy.

During the development of this thesis, it has been my privilege to work with brilliant and wonderful people. Among all, my sincere thanks go to Giuseppe Manco, for his support and guidance over the past three years. I am grateful for his collaboration whose benefits go over and above the learning of data mining techniques. His wide perspective, motivation, passion and, above all, his work ethical standards, had a great impact in shaping my enthusiasm towards research.

Rende,
November 2012

Nicola Barbieri

Contents

1	The Recommendation Process	1
1.1	Introduction	2
1.2	Formal Framework	3
1.2.1	Evaluation Metrics	4
1.3	Collaborative Filtering	6
1.3.1	Baseline Models	8
1.3.2	Neighborhood-Based Approaches	9
1.3.3	Latent Factor Models	11
2	Probabilistic Approaches to Preference Data	17
2.1	Introduction	18
2.2	Mixture Models	19
2.3	Probabilistic Topic Models	21
2.4	Probabilistic Matrix Factorization	24
3	Pattern Discovery and Users' Profiling in CF Data	27
3.1	Introduction	28
3.2	Characterizing Relationships Through Co-Clustering	30
3.2.1	A Block Mixture Model for Preference Data	30
3.2.2	Pattern Discovery using BMM	35
3.3	Users' Profiling with Soft Constraints	43
3.3.1	Community Detection and Neighborhood Regularization	48
3.3.2	Experimental Evaluation	50
4	Enhancing Recommendation Accuracy	55
4.1	Introduction	56
4.2	Probabilistic Methods for Top-N Recommendation	57
4.2.1	Evaluating Recommendation Accuracy	57
4.2.2	Probabilistic Modeling of Item Ranking	59
4.2.3	Evaluation	60
4.2.4	Predicted Rating	60

VIII Contents

4.2.5	Item Selection and Relevance	62
4.2.6	Discussion	63
4.3	Modeling Item Selection and Relevance.....	65
4.3.1	The Bayesian User Community Model	65
4.3.2	Parameter Estimation	66
4.3.3	Experimental Evaluation	69
5	Hierarchical Co-clustering of Users' Preference Data	75
5.1	Introduction	76
5.2	A 2-phase Probabilistic Hierarchical Co-clustering Approach ..	77
5.2.1	Modeling User Communities.	80
5.2.2	Local Community Patterns via Topic Analysis.....	83
5.2.3	Computational aspects.	83
5.2.4	Evaluation	85
5.3	A Bayesian Hierarchical Model for Preference Data	90
5.3.1	The Bayesian Hierarchical Model	92
5.3.2	Evaluation	99
6	Conclusions	105
	References	111

The Recommendation Process

1.1 Introduction

With the increasing volume of information, products, services (or, more generally, items) available on the Web, the role of *Recommender Systems (RS)* [1] and the importance of highly-accurate recommendation techniques have become a major concern both in e-commerce and academic research. In particular, the goal of a RS is to provide users with not trivial recommendations, that are useful to directly experience potentially interesting items. Moreover, their exploitation in e-commerce can also provide more interactions between the users and the system, that can be profitably exploited for delivering more accurate recommendations. RSs are widely employed in different contexts, from music (Last.fm ¹) to books (Amazon ²), movies (Netflix ³) and news (Google News ⁴[2]), and they are quickly changing and reinventing the world of e-commerce[3].

Recommendation can be considered as a “push” system which provides users with a personalized exploration of a wide catalog of possible choices. While in a search based system the user is explicitly required to type a query (what she is looking for), here the query is implicit and corresponds to all the past interactions of the user with the system (items/web pages previously purchased/viewed). Recommendations, as introduced before, help users in exploring and finding interesting items which the user may not found on her own. 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 Items in Your Recent History Also Bought”. The goal of the provider of the service is not just to transform a regular user in a buyer, but also make her browsing experience more comfortable, building a strong loyalty bond. This idea is better explained by Jeff Bezos (chief of Amazon.com) words: “If I have 3 million customers on the Web, I should have 3 million stores on the Web”.

The strategical importance of the development of always more 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 high accuracy recommendation approaches. On October 2006, Netflix, leader in the movie-rental American market, released a dataset containing more of 100 million ratings and promoted a competition, the *Netflix Prize* ⁵[4], whose goal was to produce a 10% improvement on the prediction quality achieved by its own recommender system, *Cinematch*. The competition lasted three years and was attended by several research groups from all over the world, improving and inspiring a fruitful research. During

¹ last.fm

² amazon.com

³ netflix.com

⁴ news.google.com

⁵ netflixprize.com/

this period a huge number of recommendation approaches has been proposed and the improvements reached in personalization and recommendation encouraged the emergence of new business models based on RSs.

In this chapter we will provide an overview of the recommendation scenario; in particular, we will firstly provide a formal framework which specifies some notations used throughout this work and discuss how to evaluate recommendations. Then we will introduce and discuss the most known collaborative filtering approaches, and we will present an empirical comparison of the main techniques.

1.2 Formal Framework

Recommendation is a particular form of information filtering which analyze past user's preference on a catalog of items to generate a personalized list of suggested items. Thus, in the following, we will introduce some basilar notations to model users, items, and their associated preferences.

Let $\mathcal{U} = \{u_1, \dots, u_M\}$ be a set of M users and $\mathcal{I} = \{i_1, \dots, i_N\}$ 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} . User's 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 preference) assigned by user u to item i . User preference data can be classified as *implicit* or *explicit*. Implicit data corresponds to mere observations of co-occurrences of users and items. Hence, the generic entry r_i^u of the user-item rating matrix \mathbf{R} is a binary value. Precisely, $r_i^u = 0$ means that u has not yet experienced i , whereas $r_i^u = 1$ simply signifies that user u has been observed to experience item i . Explicit data is more informative, since it records the actual ratings from the individual users on the experienced items. Ratings are essentially judgments on the interestingness of the corresponding items. Generally, in the context of real-world recommender systems, ratings are scores in a (totally-ordered) numeric domain \mathcal{V} , which is a fixed rating scale that often includes a small number of interestingness levels, such as a five-star rating scale. In such cases, for each pair $\langle u, i \rangle$, rating values r_i^u fall within a limited range $\mathcal{V} = \{0, \dots, V\}$, 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 $M \gg N$) and, in practical applications, the rating matrix \mathbf{R} is characterized by an exceptional sparseness (e.g., more than 95%), since the individual users tend to rate a limited number of items. The set of items rated by user u is denoted by $\mathcal{I}_{\mathbf{R}}(u) = \{i \in \mathcal{I} | r_i^u > 0\}$. Dually, $\mathcal{U}_{\mathbf{R}}(i) = \{u \in \mathcal{U} | r_i^u > 0\}$ is the set of all those users, who rated item i . Any user u with a rating history, i.e., such that $\mathcal{I}_{\mathbf{R}}(u) \neq \emptyset$ is said to be an active user. Both $\mathcal{I}_{\mathbf{R}}(u)$ and $\mathcal{U}_{\mathbf{R}}(i)$ can be empty. This is known as *cold start*, and it generally happens 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.

Fig 1.1 exemplifies the aforementioned notions. The illustration sketches a recommendation scenario with 10 users, 10 items and explicit observed preferences. The set of users who rated item i_2 is $\mathcal{U}_{\mathbf{R}}(i_2) = \{u_1, u_4, u_8, u_{10}\}$. Also, the set of items rated by user u_2 is $\mathcal{I}_{\mathbf{R}}(u_2) = \{i_3, i_5, i_7, i_9\}$. 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		3		3	3		5		
u_6				2	1	3			4	5
u_7	4			5	5			1	1	
u_8	3	4		3			1		3	
u_9	1				3	2			4	4
u_{10}		5				5			5	4

Fig. 1.1. Example of Rating Matrix.

Given an active user u , the goal of a RS is to provide u with a recommendation list $\mathcal{L} \subseteq \mathcal{I}$ including unexperienced items (i.e., $\mathcal{L} \cap \mathcal{I}_{\mathbf{R}}(u) = \emptyset$), that are expected to be of interest to u . This clearly involves predicting the interest of u into unrated items. Let \hat{r}_i^u denote the predicted rating of the user u on the item i , while we denote by p_i^u the overall predicted preference for the same pair, which is actually used to produce a ranked list of items. In a first instance, we can assume that candidate items for recommendation are ranked according to their predicted preference. This assumption and other ranking functions will be later analyzed in Chap. 4.

1.2.1 Evaluation Metrics

Different evaluation metrics have been proposed in order to evaluate the accuracy of a RS[5, 6, 7]. Recommendations usually come in the form of a list of the k items that the user might like the most. Intuitively, an accuracy metric should measure how close the predicted list is to the actual ranked list of preferences of the user, or how close is the predicted rating from the its actual preference value. According to [5], evaluation metrics for recommendations can be classified in three classes: *prediction accuracy*, *classification accuracy* and *rank accuracy* metrics. Predictive accuracy metrics measure how close the predicted ratings are to the actual ratings. Classification accuracy is appropriate metric in case of binary preferences, because it measures the frequency with which the system makes the right choice in the item suggestion phase. Rank accuracy metrics measure how close the top- k list of recommended items is close to the actual order of the same items.

A common framework in the evaluation of the predictive capabilities of a RS algorithm is to split the rating matrix \mathbf{R} 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.

Prediction accuracy is measured by using statistical metrics that compare predicted values with actual ratings:

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

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

- **Mean Squared Error (MSE)**: computes the mean squared error between the observed ratings and the predicted values:

$$MSE = \frac{1}{|\mathbf{S}|} \sum_{(u,i) \in \mathbf{S}} (r_i^u - \hat{r}_i^u)^2. \quad (1.2)$$

- **Root Mean Squared Error (RMSE)**: it has been used as Netflix Prize metric for the prediction error. It is very similar to MSE but emphasizes large errors:

$$RMSE = \sqrt{\frac{1}{|\mathbf{S}|} \sum_{(u,i) \in \mathbf{S}} (r_i^u - \hat{r}_i^u)^2}. \quad (1.3)$$

- **Mean Prediction Error (MPE)**: measures how many predictions differ from the actual rating values:

$$MPE = \frac{1}{|\mathbf{S}|} \sum_{(u,i) \in \mathbf{S}} [r_i^u \neq \hat{r}_i^u]. \quad (1.4)$$

MPE represents the percentage of data instances for which the recommendation system is not able to compute the exact observed rating.

Classification metrics measure the ability of the RS in detecting relevant items for each user. The recommendation list \mathcal{L}_u for the user u is generated by ranking a set of candidate items according to a personalized ranking function. The accuracy of \mathcal{L}_u is ultimately assessed through a comparison with the items appearing in $\mathcal{I}_{\mathbf{S}}(u)$. Therein, the standard classification-based metrics, i.e., precision and recall, can be adopted to evaluate the recommendation accuracy of the recommendation list. Such metrics require the capability to distinguish between relevant and irrelevant recommendations. Given a user u and a subset $\mathcal{T}_u^r \subseteq \mathcal{I}_{\mathbf{S}}(u)$ of relevant items, the degree of precision and recall of the k items within \mathcal{L} is defined as shown next:

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

Item relevance can be measured in several different ways. When explicit preference values are available, we consider as relevant all those items that received a rating greater than the average ratings in the training set:

$$\mathcal{T}_u^r = \{i \in \mathcal{I}_{\mathbf{S}}(u) | r_i^u > \bar{r}_{\mathbf{T}}\}.$$

Precision and Recall are conflicting measures of the accuracy: for instance, by increasing the size of the recommendation list, Recall is expected to increase but Precision decreases. *F-score* is the harmonic mean of Precision and Recall, used to evaluate them in conjunction:

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

Finally, ranking accuracy metrics measure the ability of the RS to generate a personalized ordering of the items, which match up with the the true users' ordering. Typically the predicted and observed orderings are compared by using Kendall's tau and Spearman's rho. This perspective is without any doubts approach is the lack of data, since generally the full user-based ranking of all items is not available.

Evaluating recommendation is one of the biggest unsolved problem of current research in RSs. In particular:

- Offline evaluation rarely match up to online results;
- Offline metrics like Prediction/Recall, RMSE and MAP are poor proxies for what happens in the online world and they have been reported to be far from measuring the real impact of recommendations on users' purchasing activities;
- Common evaluations do not consider the real users's satisfaction.

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.

1.3 Collaborative Filtering Approaches to Recommendation

State-of-the art recommendation methods have been largely approached from a *Collaborative Filtering (CF)* perspective, which essentially consists in the

posterior analysis of past interactions between users and items, aimed to identify suitable preference patterns in users' preference data. Tapestry [8] is considered the ancestor of CF systems. It was designed to support both content-based and collaborative filtering. Recording the reactions of each user on a particular document and explicit feedback, Tapestry allows mail filtering in an area of interest basing on other people annotations.

Modern CF techniques aim to predict the preferences values of users on given items, based on previously observed behavior. The assumption is that *users who adopted the same behavior in the past, will tend to agree also in the future*. The main advantage in using CF techniques relies on their simplicity: only users' past ratings are used in the learning process, no further informations, like demographic data or item descriptions, are needed. CF solves some of the main drawbacks of *Content Based (CB)* approaches [9, 10]:

- CF approaches are more general and re-usable in different context, while CB techniques require the specification of a complete profile (set of features) for each user/item;
- CB techniques provide the user with a list of products whose features are "similar" to ones that she experienced in the past. This approach may imply the recommendations of redundant items and the lack of novelty;
- The effectiveness of recommendation increases as the user provides more feedback.

According to [11], collaborative filtering approaches can be classified in two classes, *Memory-based* and *Model-based*. The first class infers the preference of the active user on an item by using the database of preferences. The most common memory-based approaches are the *Nearest Neighbors* methods, which use the whole preference history of users and items and statistical techniques to infer similarities and then use this similarity measures to make a prediction. Model-based approaches operate in two phases: initially the preference database is used to learn a compact model and, in the second phase, this model is used in order to infer users' preferences. Actually, memory-based approaches can rely on a model as well (i.e. similarity matrix in neighbors models), which is usually built in a off-line mode, but they still need to access to a database of preference values. Memory-based approaches are intuitive because, in the simplest case, they directly transforms stored preferences data in predictions, but they need to keep the whole dataset in memory. On the other hand, model-based approaches require less memory because they need to access only to the small-size model previously computed from the dataset, but the reason behind the provided predictions may not be easily interpretable. Moreover, memory-based approaches, such as *Neighborhood models*, are most effective at detecting *strong but local relationships*, while model-based approaches, such as *Latent Factor models*, can estimate *weak but global relationships*. Figure 1.2 summarizes the different approaches used by the two methods.

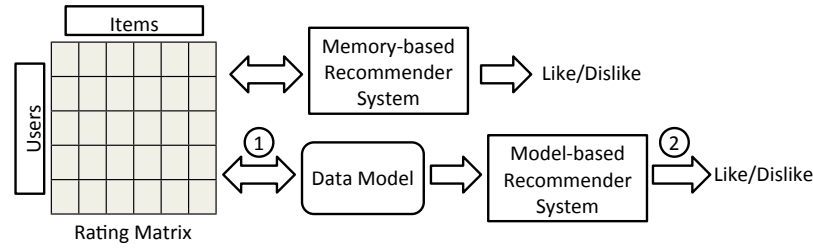


Fig. 1.2. Memory-based vs Model-based Approaches

In the following we will briefly review three family of CF approaches, namely *Baseline*, *Nearest Neighbors* and *Latent Factor* models, before focusing in more detail on Probabilistic Approaches in the next chapter.

1.3.1 Baseline Models

Baseline algorithms are the simplest approaches for predicting users' numerical preferences. This section will focus on the analysis of the following algorithms: *OverallMean*, *MovieAvg*, *UserAvg*, *DoubleCentering* and the *Global Effects Model*.

OverallMean computes the mean of all ratings in the training set, this value is returned as prediction for each pair $\langle u, i \rangle$. *MovieAvg* predicts the rating of a pair $\langle u, i \rangle$ as the mean of all ratings received by i in the training set. Similarly, *UserAvg* predicts the rating of a pair $\langle u, i \rangle$ as the mean of all ratings given by u . Given a pair $\langle u, i \rangle$, *DoubleCentering* compute separately the mean of the ratings of the movie \bar{r}_i , and the mean of all the ratings given by the user \bar{r}_u . The prediction is computed as a linear combination:

$$\hat{r}_i^u = \alpha \bar{r}_i + (1 - \alpha) \bar{r}_u, \quad (1.7)$$

where $0 \leq \alpha \leq 1$.

The *Global Effects Model* is considered one of the most useful and effective technique in the context of the Netflix Prize. The key idea, proposed by *Bell and Koren* in [12] and widely appreciated by the Netflix Prize's contestants, is to precede 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 respect to their rating average (user effect), while some items tend to receive higher rating values than others (item effect). Some other interesting considerations involve the time dimension: for example could be identified temporal patterns in user's ratings which might rise or fall over period of time. A similar analysis could be performed on items: for example some movies might receive higher ratings when they first are proposed, while ratings might fall after their release period. The proposed strategy is to estimate and remove one global

effect at time, in sequence: at each step we do not focus on the raw ratings but on the *residuals* of the previous step. In [12] authors identify 11 kind of effects, and empirically evaluate the effectiveness of this method by recording an improvement in RMSE after each cleaning step.

The main advantage of this approach relies on its capacity of removing noise from data. As instance, once global effects has been completely removed from data, the *K-Nearest Neighbors* approach, which works better on residuals because they make possible to correctly identify neighbors and the actual strength of their relationships, could be used to compute the predicted residuals and then, the final predicted values could be obtained by adding the global effects back.

1.3.2 Neighborhood-Based Approaches

Neighborhood based approaches [13, 14] compute each rating prediction by relying on a chosen portion of the data. According to a user-based perspective, the predicted rating on an item is generated by selecting the most similar users to the active one, said *neighbors*, and then using an aggregate function of their ratings. This intuitive definition corresponds to the classic behavior of asking friends for their opinions before purchasing a new item and it raises some fundamental questions:

1. How is it possible to model similarity about users/items?
2. How can we identify a neighborhood for the current user/item?
3. How can ratings of neighbors be used to generate a rating prediction?

The most common formulation of the neighborhood approach is the *K-Nearest-Neighbors (K-NN)*. The rating prediction \hat{r}_i^u is computed following simple steps: (i) a similarity function associates a numerical coefficient to each pair of users, then *K-NN* finds the *neighborhood* of u by selecting her the K most similar users; (ii) the rating prediction is computed as the average of the ratings given by neighbors on the same item, weighted by the similarity coefficients. The user-based *K-NN* algorithm is intuitive but doesn't scale because it requires the computation of similarity coefficients for each pair of users. Considering that the number of items is generally lower than the number of users, a more scalable formulation can be obtained considering an *item-based* approach [13]. Thus, in the rest of the section, we will focus on item-based approaches. According to this perspective, 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 movies to i . The underlying assumption is that the user might prefer movies more similar to the ones he liked before, because they share similar features. The prediction is computed as:

$$\hat{r}_i^u = \frac{\sum_{j \in \mathcal{N}(i;u)} s_{i,j} \cdot r_j^u}{\sum_{j \in \mathcal{N}^K(i;u)} s_{i,j}}, \quad (1.8)$$

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 in the Neighborhood based approaches: in the first phase they are used to identify the neighbors, while in the prediction phase they act as weights. Different techniques may be used to compute similarity coefficients for items or users and in most cases the similarity is computed by considering only the co-rated pairs. As instance, considering the case of the item-to-item similarity computation, only ratings from users who rated both the items are taken into analysis, Dually, the user-to-user similarity computation is commonly based on the ratings given on co-rated items. Let $\mathcal{U}_{\mathbf{R}}(i, j)$ denote the set of users who provided a rating for both i and j , i.e $\mathcal{U}_{\mathbf{R}}(i, j) = \mathcal{U}_{\mathbf{R}}(i) \cap \mathcal{U}_{\mathbf{R}}(j)$. Similarity coefficients are often computed according to the *Pearson Correlation* or the *Adjusted Cosine*, which improves the cosine similarity by taking into account the differences in rating scale between different users[13]:

$$s_{ij} = \text{Pearson}(i, j) = \frac{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i, j)} (r_i^u - \bar{r}_i) \cdot (r_j^u - \bar{r}_j)}{\sqrt{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i, j)} (r_i^u - \bar{r}_i)^2} \sqrt{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i, j)} (r_j^u - \bar{r}_j)^2}};$$

$$s_{ij} = \text{AdjCosine}(i, j) = \frac{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i, j)} (r_i^u - \bar{r}_u) \cdot (r_j^u - \bar{r}_u)}{\sqrt{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i, j)} (r_i^u - \bar{r}_u)^2} \sqrt{\sum_{u \in \mathcal{U}_{\mathbf{R}}(i, j)} (r_j^u - \bar{r}_u)^2}}.$$

As discussed in [15], similarity coefficients based on a larger support are more reliable than the ones computed using few rating values, so it is a common practice to weight the similarity coefficients using the support size. This technique is called *shrinkage*:

$$s'_{i,j} = \frac{s_{i,j} \cdot |\mathcal{U}_{\mathbf{R}}(i, j)|}{|\mathcal{U}_{\mathbf{R}}(i, j)| + \alpha}, \quad (1.9)$$

where α is an empirical value.

An improved version of K -NN can be obtained by refining the neighborhood model with baseline values. Formally:

$$\hat{r}_i^u = b_i^u + \frac{\sum_{j \in \mathcal{N}^K(i; u)} s_{i,j} \cdot (r_j^u - b_j^u)}{\sum_{j \in \mathcal{N}^K(i; u)} s_{i,j}}, \quad (1.10)$$

where b_i^u denote a baseline value.

An alternative way to estimate item-to-item interpolation weights is by solving a least squares problem minimizing the error of the prediction rule. This strategy, proposed in [16, 17], defines the *Neighborhood Relationship Model*, one of the most effective approaches applied in the Netflix prize competition. In this case the prediction is computed as:

$$\hat{r}_i^u = \frac{\sum_{j \in \mathcal{N}^K(i; u)} w_{i,j} \cdot r_j^u}{\sum_{j \in \mathcal{N}^K(i; u)} w_{i,j}}, \quad (1.11)$$

where the non-negative interpolation weights $w_{i,j}$ are computed as the solution of the following optimization problem:

$$\min_W \sum_{v \in \mathcal{I}_{\mathbf{R}}(i)} \left(r_i^v - \frac{\sum_{j \in \mathcal{N}(i;u,v)} w_{i,j} \cdot r_j^v}{\sum_{j \in \mathcal{N}(i;u,v)} w_{i,j}} \right)^2 \quad s.t. \ w_{i,j} \geq 0 \ \forall i, j \in \mathcal{I}$$

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

Fig. 1.3 shows the performance in prediction accuracy of K -NN models. We empirically compare two different choices for the baseline function:

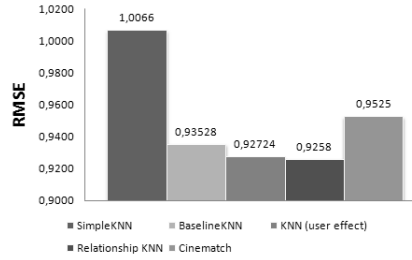


Fig. 1.3. Prediction Accuracy: K -NN models

BaselineK-NN relies on a double-centering baseline, whereas the *K-NN (user effect)* version computes the baseline values according to the *User Effect Model* [12]. Except the *SimpleK-NN*, all the considered K -NN approaches improve *Cinematch*'s precision, and the best performance is achieved by the *Neighborhood Relationship Model*.

1.3.3 Latent Factor Modes

The assumption behind *Latent Factor models* is that the preference of the user can be expressed considering a set of contributes which 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)* [18] is a dimensionality reduction technique which assumes a latent structure in word usage across documents. *LSI* uses *Singular Value Decomposition* to represents terms and documents in the features space: some of these feature components are be 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 $SVD(\mathbf{A})$ Figure 1.4, is defined as:

$$SVD(\mathbf{A}) = \mathbf{U} \times \Sigma \times \mathbf{V}^T, \quad (1.12)$$

where:

- \mathbf{U} is an $m \times m$ orthogonal matrix⁶; 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 $n \times n$ orthogonal matrix; the first r columns of \mathbf{V} are the eigenvectors of $\mathbf{A}^T\mathbf{A}$ (*right singular vectors* of \mathbf{A});
- Σ is a $n \times n$ diagonal matrix with only r nonzero values, such that: $\Sigma = \{\sigma_1, \dots, \sigma_n\}$, $\sigma_i \geq 0 \forall 1 \leq i < n$, $\sigma_i \geq \sigma_{i+1}$, $\sigma_j = 0 \forall j \geq r + 1$;
- $\{\sigma_1, \dots, \sigma_n\}$ are the nonnegative square root of the eigenvalues of $\mathbf{A}^T\mathbf{A}$ and are called *singular values* of \mathbf{A} .

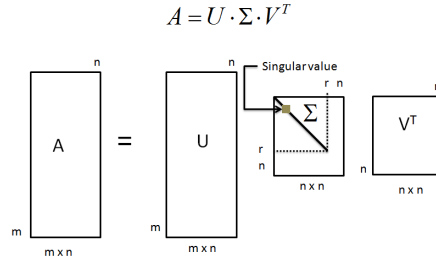


Fig. 1.4. Singular Value Decomposition

SVD has an important property: it provides the best low-rank linear approximation of the original matrix \mathbf{A} . Fixed 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 Σ and reducing the matrices \mathbf{U} and \mathbf{V} , the original matrix can be approximated using \mathbf{A}_k :

$$\mathbf{A} \approx \mathbf{A}_k = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T \quad (1.13)$$

where \mathbf{U}_k is obtained by removing $(r - k)$ columns from the matrix \mathbf{U} and \mathbf{V}_k^T is produced by removing $(r - k)$ rows from the matrix \mathbf{V} . An example of this approximation is given in Figure 1.5. Considering the text analysis case, *LSI* factorizes the original term-document matrix into the product of three matrices which reflects 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 at the same time noise [19].

Several works have studied the application of SVD in recommender systems [20, 21]: the dimensional reduction approach is very useful for CF because is able to produce a *low-dimensional* representation of the original *high-dimensional* rating matrix \mathbf{R} and to capture the hidden relationships between

⁶ $\mathbf{U}^T\mathbf{U} = \mathbf{I}$

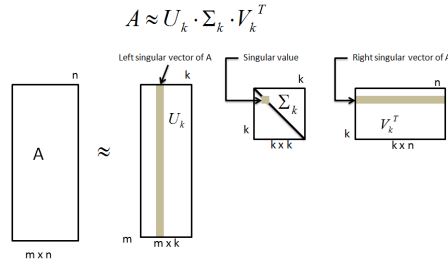


Fig. 1.5. k -rank approximation of \mathbf{A}

users and products that could be used to infer the user’s preference on considered items. Focusing on a scenario that involves ratings given by a set of users on a set of movies, it is plausible to assume that the user’s rating on a movie is influenced by a set of feature of the movie itself and by the user’s preference on those features. An intuitive idea of what of hidden features might represent in this scenario, is the *genre interpretation*: assuming the existence of a limited number of different genres of movies (action, romance, comedy, etc.), the rating is influenced by the user’s preference on those genres and by a movie’s factors that represent how much the considered movie belongs to each genre. Figure 1.6 shows an example of this scenario, with 3 hidden factors.

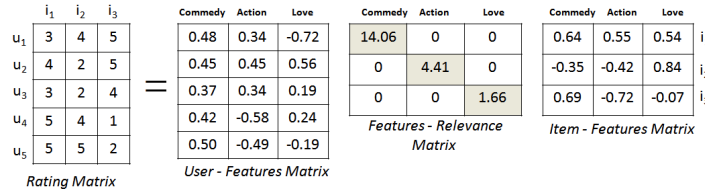


Fig. 1.6. Example of the application of SVD decomposition

With an abuse of notation, in this section will use a simplified but equivalent formalization for the SVD, in which the original matrix is approximated by the product of 2 component matrices with K features:

$$\mathbf{R} \approx \left(\mathbf{U}_k \sqrt{\Sigma_k^T} \right) \left(\sqrt{\Sigma_k} \mathbf{V}_k^T \right) = \mathbf{U} \cdot \mathbf{V}, \tag{1.14}$$

where \mathbf{U} is a $M \times K$ matrix and \mathbf{V} is a $K \times N$. Intuitively, each users’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:

$$\hat{r}_i^u = \sum_{k=1}^K \mathbf{U}_{u,k} \cdot \mathbf{V}_{k,i}. \tag{1.15}$$

Fixed the number of features K , we can estimate the feature matrices by solving this optimization problem:

$$(\mathbf{U}, \mathbf{V}) = \arg \min_{\mathbf{U}, \mathbf{V}} \left[\sum_{(u,i) \in \mathcal{T}} \left(r_i^u - \sum_{k=1}^K \mathbf{U}_{u,k} \mathbf{V}_{k,i} \right) \right]. \quad (1.16)$$

Funk in [22] proposed an incremental procedure, based on gradient descent, to minimize the error of the model on observed ratings. The feature matrices are randomly initialized and updated as follows:

$$\begin{aligned} \mathbf{U}'_{u,k} &= \mathbf{U}_{u,k} + \eta (2e_{u,i} \cdot \mathbf{V}_{k,i}), \\ \mathbf{V}'_{k,i} &= \mathbf{V}_{k,i} + \eta (2e_{u,i} \cdot \mathbf{U}_{u,k}), \end{aligned} \quad (1.17)$$

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 procedure could be further improved considering regularization coefficients λ . Updating rules become:

$$\begin{aligned} \mathbf{U}'_{u,k} &= \mathbf{U}_{u,k} + \eta (2e_{u,i} \cdot \mathbf{V}_{k,i} - \lambda \cdot \mathbf{U}_{u,k}), \\ \mathbf{V}'_{k,i} &= \mathbf{V}_{k,i} + \eta (2e_{u,i} \cdot \mathbf{U}_{u,k} - \lambda \cdot \mathbf{V}_{k,i}). \end{aligned} \quad (1.18)$$

A more refined model can be obtained by integrating a combining a baseline model with the SVD prediction:

$$\hat{r}_i^u = b_i^u + \sum_{k=1}^K \mathbf{U}_{u,k} \cdot \mathbf{V}_{k,i}, \quad (1.19)$$

or considering user and item bias components [23]:

$$\hat{r}_i^u = c_u + d_i + \sum_{k=1}^K \mathbf{U}_{u,k} \cdot \mathbf{V}_{k,i}, \quad (1.20)$$

where c_u is the user-bias vector and d_i is the item-bias vector, which are trained simultaneously with the features matrices with regularization rate λ_2 :

$$\begin{aligned} c'_u &= c_u + \eta (e_{u,i} - \lambda_2 (c_u + d_i - \bar{r}_{\mathbf{T}})) \\ d'_i &= d_i + \eta (e_{u,i} - \lambda_2 (c_u + d_i - \bar{r}_{\mathbf{T}})). \end{aligned} \quad (1.21)$$

An alternative and more effective formulation, known as *Asymmetric SVD* [23], models each user by exploiting all the items that she rated:

$$\mathbf{U}_{u,k} = \frac{1}{\sqrt{|\mathcal{I}(u)| + 1}} \sum_{i \in \mathcal{I}(u)} w_{k,i}. \quad (1.22)$$

A slightly different version, named *SVD++* [15], models each user by considering both a user-features vector and the corresponding bag-of-items representation.

Latent factor models based on the SVD decomposition change according to the number of considered features and the structure of model, characterized by presence of bias and baseline contributes. The optimization procedure used in the learning phase plays an important role: learning could be incremental (one feature at the time) or in batch (all features are updated during the same iteration of data). Incremental learning usually achieves better performances at the cost of learning time.

To assess the predictive capabilities of latent factor models, we tested several version of SVD model, considering the batch learning with learning rate 0.001. The regularization coefficient, where needed, has been set to 0.02. To avoid overfitting, the training set has been partitioned into two different parts: the first one is used as actual training set, while the second one, called validation set, is used to evaluate the model. The learning procedure is stopped as soon the error on the validation set increases. Fig.1.7 shows the accuracy of the main SVD approaches. An interesting property of the analyzed models is that they reach convergence after almost the same number of iteration, no matter how many features are considered. Better performances are achieved if the model includes bias or baseline components; the regularization factors decrease the overall learning rate but are characterized by a high accuracy. In the worst case, the learning time for the regularized versions is about 60 min. The *SVD++* model with 20 features obtains the best performance with a relative improvement on the Cinematch score of about 5%.

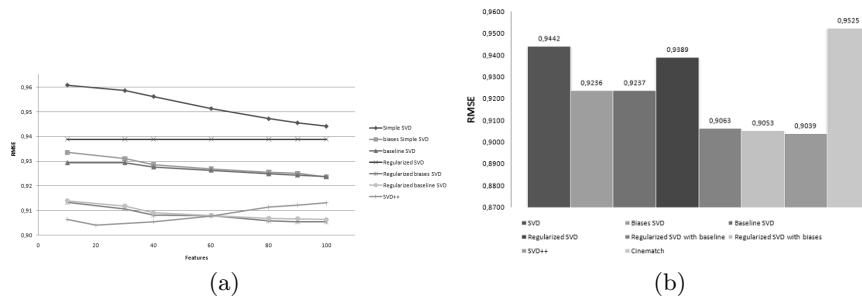


Fig. 1.7. Prediction Accuracy: Comparison of SVD based models

2.1 Introduction

Probabilistic approaches assume that each preference observation is randomly drawn from the joint distribution of the random variables which model users, items and preference values (if available). Typically, the random generation process follows a bag of words assumption and preference observations are assumed to be generated independently. A key difference between probabilistic and deterministic models relies in the inference phase: while the latter approaches try to minimize directly the error made by the model, probabilistic approaches do not focus on a particular error metric; parameters are determined by maximizing the likelihood of the data, typically employing an Expectation Maximization procedure. In addition, background knowledge can be explicitly modeled by means prior probabilities, thus allowing a direct control on overfitting within the inference procedure [24]. By modeling prior knowledge, they implicitly solve the need for regularization which affects traditional gradient-descent based latent factors approaches.

Further advantages of probabilistic models can be found in their easy interpretability: they can often be represented by using a graphical model, which summarizes the intuition behind the model by underlying causal dependencies between users, items and hidden factors. Also, they provide an unified framework for combining collaborative and content features [25, 26, 27], to produce more accurate recommendations even in the case of new users/items. Moreover, assuming that an explicit preference value is available, probabilistic models can be used to model a distribution over rating values which can be used to infer confidence intervals and to determine the confidence of the model in providing a recommendation.

In the following we will briefly introduce some paradigmatic probabilistic approaches to recommendation, which can be categorized into 3 main classes: *Mixture Models*, *Probabilistic Topic Models* and *Probabilistic Matrix Factorization Techniques*. Fig. 2.1 offers an overview of the generative models of the main techniques. The underlying idea of probabilistic models based on latent factors is that each preference observation $\langle u, i \rangle$ is generated by one of k possible states, which informally model the underlying reason why u has chosen/rated i . Based on the mathematical model, two different inferences can be then supported to be exploited in the modelling of preference data [28]:

- *Forced Prediction*: the model provides estimate of $P(r|u, i)$, which represents the conditional probability that user u assign a rating value r given the item i ;
- *Free prediction*: the item selection process is included in the model, which is typically based on the estimate of $P(r, i|u)$. In this case we are interested in predicting both the item selection and the preference of the user for each selected item. $P(r, i|u)$ can be factorized as $P(r|i, u)P(i|u)$; the resulting model still includes a component of forced prediction which however is weighted by the item selection component and thus allows a more precise estimate of user's preferences.

2.2 Mixture Models

A mixture model [29] is a general probabilistic approach which introduces a set of latent factor variables to represent a generative process in which data points may be generated according to a finite set of probability distribution. The simplest latent factor approach to model users' preference data is the **Multinomial Mixture Model** (MMM, [30]). According to this model, we assume that a user u is associated with a latent factor z , which models her “attitude”, and ratings for an item i are generated according to this factor. The corresponding generative semantic, represented graphically in Figure 2.1(a), can be summarized as:

1. For each user $u = \{1, \dots, M\}$
 - a) Sample a user attitude z according to a multinomial distribution over the latent variable θ
 - b) For each $i \in \mathcal{I}(u)$
 - i. Draw a rating value r according to multinomial distribution over rating values $\beta_{i,z}$

The θ parameter here is the prior probability distribution over the possible states of the latent variables, $P(Z)$, whereas $\beta_{i,z}$ is a multinomial over \mathcal{V} which drives for the rating generation $P(R = r|i, z)$. The learning of the parameter of the model can be accomplished by employing the EM algorithm, which also provides, with the “responsibilities”, an estimate of the mixing proportion for each user $P(z|u)$. Given those parameters, the probability distribution over ratings for the pair $\langle u, i \rangle$ can be computed as:

$$P(r|i, u) = \sum_{z=1}^K P(z|u) \cdot \beta_{z,i,r} \quad (2.1)$$

where

$$P(z|u) \propto P(\mathbf{u}_{obs}|z)\theta_z$$

and \mathbf{u}_{obs} represents the observed values (u, i, r) in \mathbf{R} .

The **User Communities Model** (UCM, [31]) (see Sec. 5.2.1) refines the inference formula Eq. 2.1 by introducing some key features. First, the exploitation of a unique prior distribution θ over the user communities helps in preventing overfitting. Second, adds flexibility in the prediction by modeling an item as an observed (and hence randomly generated) component, thus following a free-prediction perspective.

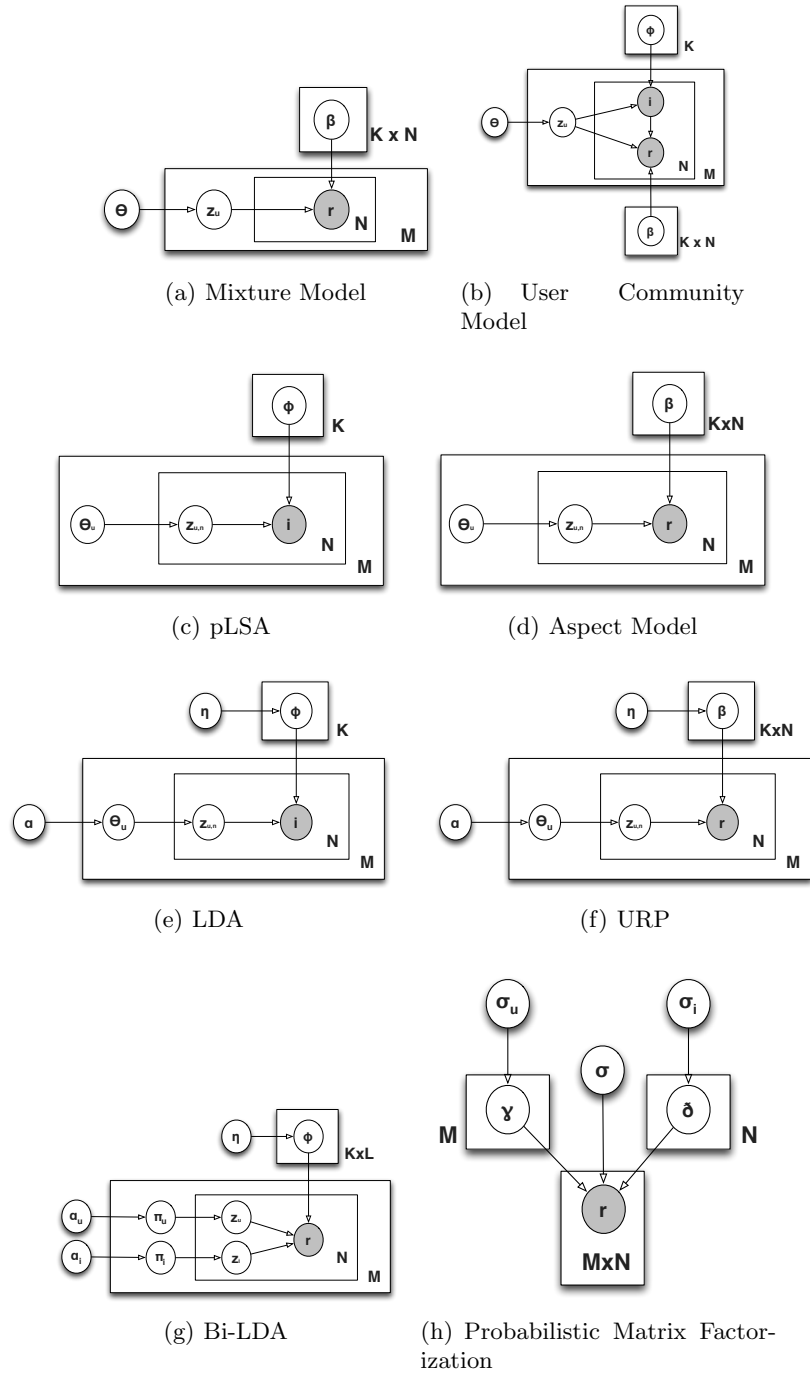


Fig. 2.1. Generative models to preference data.

2.3 Probabilistic Topic Models

Probabilistic Topic models [32, 33, 34] include a suite of techniques which widely used in text analysis: they provide a low-dimensional *semantic* representation that allows the discovering of *global relationships* within data. Given a *corpus of documents*, the assumption behind this family of techniques is that each document may exhibit multiple topics and each word in the document is generated by a particular topic. A generative process for CF, based on latent topics, is shown in Figure 2.2. In the CF context, topics could be interpreted as

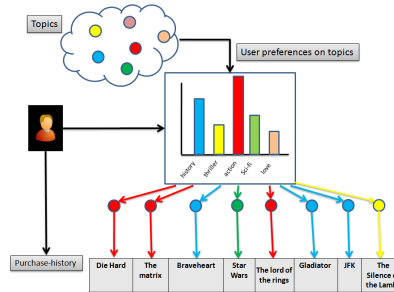


Fig. 2.2. Latent Class Model for CF - Generative Process

genres, item categories or user’s attitudes, although no prior meaning is generally associated with them. A proper definition of topics might be obtained by considering them as “abstract preference pattern”: users, or items, participate in each preference pattern with a certain degree, and these membership weights project each user/item into the latent factor space. We assume that there are a fixed number of topics, and each user is characterized by her own preference on genres. For example, in Figure 2.2, the considered user shows a particular interest in action and historic movies, while her interest in romance is low. Each genres specifies the probability of observing each single item. Movies like “Independence Day” and “Die hard” will have an higher probability of being observed given the “action” topic, than in the context of “romance”. Given an user and her preferences on topics (which defines preferences on movies), the corresponding purchasing history can be generated by choosing a topic, and then drawing an item from the corresponding distribution over items. In the example, the first topic to be chosen is “action”, which generates the movie “Die Hard”; the process of topic and item selection is iteratively repeated to generate the complete purchase history of the current user.

The **probabilistic Latent Semantic Analysis** approach (PLSA, [35, 36]) specifies a co-occurrence data model in which the user u and item i are conditionally independent given the state of the latent factor Z . Differently from the previous mixture model, where a single latent factor is associated

with every user u , the PLSA model associates a latent variable with every observation triplet $\langle u, i, r \rangle$. Given an active user u , her purchase history can be generated according to the following generative semantic:

1. For $n = 1$ to $|\mathcal{I}(u)|$:
 - a) Select a user profile z according to user-specific multinomial distribution over topics $\theta_u = P(Z = z|u)$;
 - b) Pick an item i by sampling from the multinomial distribution $\phi_z = P(i|z)$;

Hence, different ratings of the same user can be explained by different latent causes in PLSA (modeled as priors $\{\theta_u\}_{1,\dots,m}$ in Figure 2.1(c)), whereas a mixture model assumes that all ratings involving the same user are linked to the same underlying community. PLSA directly supports item selection:

$$P(i|u) = \sum_{z=1}^K \phi_{z,i} \theta_{u,z} \quad (2.2)$$

where ϕ_z represents a multinomial distribution over items. The main drawback of the PLSA approach is that it cannot directly model new users, because the parameters $\theta_{u,z} = P(z|u)$ are specified only for those users in the training set.

We consider three further variants for the PLSA, where explicit preferences are modeled by an underlying distribution $\beta_{z,i}$. In the **Aspect Model** (AM, [36]) $\beta_{z,i}$ is a multinomial over \mathcal{V} . In this case, the rating probability can be modeled as

$$P(r|u, i) = \sum_{z=1}^K \beta_{r,i,z} \theta_{u,z} \quad (2.3)$$

Conversely, the **Gaussian PLSA** (G-PLSA, [24]) models $\beta_{z,i} = (\mu_{iz}, \sigma_{iz})$ as a gaussian distribution, and provides a normalization of ratings through the user's mean and variance, thus allowing to model users with different rating patterns. The corresponding rating probability is

$$P(r|u, i) = \sum_{z=1}^K \mathcal{N}(r; \mu_{iz}, \sigma_{iz}) \theta_{u,z} \quad (2.4)$$

The **Flexible Mixture Model (FMM)** [37] extends the Aspect Model, by allowing each user/item to belong to multiple clusters, which are determined simultaneously, according to a coclustering approach. Assuming the existence of K user clusters indexed by c and L item clusters, indexed by d , and let $P(c_k)$ be the probability of observing the user-cluster k with $P(u|c_k)$ being the probability of observing the user profile u given the cluster k and

using the same notations for the item-cluster, the joint probability $P(u, i, r)$ is defined as:

$$P(u, i, r) = \sum_{c=1}^C \sum_{d=1}^D P(c)P(d)P(u|c)P(i|d)P(r|c, d)$$

The **Latent Dirichlet Allocation** [32] is designed to overcome the main drawback in the PLSA-based models, by introducing Dirichlet priors, which provide a full generative semantic at user level and avoid overfitting. The generative process that characterized LDA can be formalized as follows:

1. For each user $u = \{1, \dots, M\}$
 - a) Choose N_u according to a Poisson's distribution, or to any other distribution that could model the number of items in the user profile
 - b) Choose $\theta_u \sim Dir(\alpha)$
 - c) For each of the N_u items to be generated:
 - i. Choose a topic $z \sim Multinomial(\theta_u)$
 - ii. Choose an item i according to the multinomial probability conditioned on the selected topic: $P(i|z, \beta)$

Again, two different formulations, are available, based on whether we are interested in modeling implicit (LDA) or explicit (**User Rating Profile**, URP[38]) preference values. In the first case, we have:

$$P(i|u) = \int \sum_{z=1}^K \phi_{z,i} \theta_z P(\theta|\mathbf{u}_{obs}) d\theta \quad (2.5)$$

(where $P(\theta|\mathbf{u}_{obs})$ is estimated in the inference phase). Analogously, for the URP we have

$$P(r|u, i) = \int \sum_{z=1}^K \beta_{z,i,r} \theta_z P(\theta|\mathbf{u}_{obs}) d\theta \quad (2.6)$$

The idea behind the models presented so far, is the starting point of more advanced approaches that include side information to achieve better results in prediction accuracy and provide tools for cold-start recommendation [26, 25, 39]. Interesting results have been obtained by adopting a co-clustering structure [40, 41, 42, 37, 43]. As instance, the *Bi-LDA* model proposed in [43] (represented in Figure 2.1(g)), extends the URP model employing two interacting LDA models which enforce the simultaneous clustering of users and items in homogeneous groups. The rating probability according to this bayesian coclustering model can be computed as:

$$P(r|u, i) = \int \int \sum_{z_u, z_i} P(r|z_u, z_i)P(z_u|\pi_u)P(z_i|\pi_i)P(\pi_u|\mathbf{u}_{obs})P(\pi_i|\mathbf{i}_{obs})d\pi_u d\pi_i \quad (2.7)$$

where both $P(\pi_u|\mathbf{u}_{obs})$ and $P(\pi_i|\mathbf{i}_{obs})$ are estimated in the inference phase.

The generative process of the Bi-LDA is as follows:

1. For each user $u = \{1, \dots, M\}$
 - a) Choose N_u according to a Poisson's distribution
 - b) For each of the N_u items to be generated:
 - i. Choose a user attitude $z_{u,n}^u \sim Discrete(\boldsymbol{\pi}^u)$
 - ii. Choose an item category $z_{u,n}^i \sim Discrete(\boldsymbol{\pi}^i)$
 - iii. Generate a rating value for the chosen item according to the distribution $P(r|\boldsymbol{\varphi}_{z_{u,n}^u, z_{u,n}^i})$

2.4 Probabilistic Matrix Factorization

The **Probabilistic Matrix Factorization** approach (PMF, [44]) reformulates the rating assignment as a matrix factorization. Given the latent user and item k -feature matrices γ_u and δ_i , (where K denotes the number of the features employed in the factorization), the preference value is generated by assuming a Gaussian distribution over rating values conditioned on the interactions between the user and the considered item in the latent space, as shown in Figure 2.1(h). In practice, $P(r|u, i)$ is modeled as a gaussian distribution, with mean $\gamma_u^T \delta_i$ and fixed variance σ :

$$P(r|u, i) = \mathcal{N}(r; \gamma_u^T \delta_i, \sigma^2) \quad (2.8)$$

The conditional distribution of the observed data, given the latent user and item feature matrices, Γ and Δ , is computed as

$$P(\mathbf{R}|\Gamma, \Delta, \sigma) = \prod_{u=1}^M \prod_{i=1}^N \mathcal{N}(r_i^u; \gamma_u^T \delta_i, \sigma^2)^{\delta(u, i)}$$

where $\delta(u, i)$ is an indicator function that is equal to 1 if the user u has rated the item i , zero otherwise. Moreover, to avoid overfitting, user and item features are generated by two zero-mean spherical Gaussian priors:

$$P(\Gamma|\sigma_u) = \prod_{u=1}^M \mathcal{N}(\gamma_u; 0, \sigma_u^2 \mathbf{I}) \quad P(\Delta|\sigma_i) = \prod_{i=1}^N \mathcal{N}(\delta_i; 0, \sigma_i^2 \mathbf{I})$$

Different extensions of the basic framework have been proposed: a constrained version [44] of the PMF model is based on the assumption that users who have

rated similar sets of items are likely to exhibit similar preferences, while its bayesian generalizations [45] and extensions with side information [46, 47, 48] are characterized by an higher prediction accuracy.

**Pattern Discovery and Users' Profiling in CF
Data**

3.1 Introduction

Recent studies [49, 50] have shown that the focus on prediction does not necessarily help in devising good recommender systems. As the number of available customers increases, it is always more difficult to understand, profile and segment their behaviors and a similar consideration holds for the catalog of products. Under this perspective, CF models should be considered in a broader sense, for their capability to understand deeper and hidden relationships among users and products they like. For example, the high dimensional preference matrix can be partitioned to detect user communities and item categories; the analysis of the relationships between these two levels can provide a faithful yet compact description of the data which can be exploited for better decision making.

The latent variable modeling, exploited within a probabilistic framework, offers some important, and easily interpretable, insights into the users's purchase and preference patterns. In this chapter we are going to discuss some of the applications of probabilistic models to the task of pattern discovery in collaborative filtering data.

A first study which describes the application of probabilistic approaches, to understand users's preference and interests, is presented in [51]. In this work, authors exploited the pLSA model to infer the underlying task of a web browsing session and to discover hidden semantic relationships between users and web pages. More specifically, the web session of the user u can be represented using co-occurrence pairs notations (u, p) , where p is a collection of web objects, called *pageview*, resulting from a single action of the user. During a web session the user can visit different pageviews, and their corresponding importance within the session can be represented using a weight function $w(u, p)$, which could be binary (existence or non existence of the pageview in a user session) or numeric (number of times that the pageview p is visited within the same user session, duration of the pageview in that session). The probability distributions that define the pLSA model and other probabilities, like the prior $P(z)$ or the user specific distribution $P(z|u)$, which can be obtained by marginalization or Baye's rule, allow the following analysis:

- **Characterize Topics by Items:** Each topic, represented by a latent factor, can be characterized by a set of web pages which are strongly associated with it. Those pages, or in general products belonging to an user purchase session, are called *characteristic objects* for the topic z . Intuitively, a characteristic page given a topic z is a page which exhibits an high probability of being observed given the considered topic ($P(i|z)$ is high) and a low probability for being generated by other different topics. The *characteristic objects* for a topic z_k can be defined as the set of products/pages, indexed by i , such that: $P(i|z_k) \cdot P(z_k|i) \geq \mu$, where μ is a fixed threshold. Characteristic objects identification allows a better understanding of hidden factors. The pLSA model does not associate any *a priori* meaning with the states of the hidden variables which are used to

detect relationships between users, grouping like-minded users in the same community, and items, grouping similar items in the same category. Characteristic objects might provide *a posteriori* interpretation for the state of the hidden variable, which can be an useful starting point for the following analysis of users patterns and relationships between different topics.

- **Characterize Topics by User Histories:** A similar approach can be used to associate each topic with a set of *characteristic users*. A characteristic user u_k for a given topic z_k is an user which prefers z_k over the other different topics, which implies that a wide fraction of the products in the purchase history of u_k share the topic z_k . The characteristic users for the topic z_k can be formally defined as the set of users u which satisfy $P(u|z_k) \cdot P(z_k|u) \geq \mu$. In this case $p(u|z_k)$ can be computed from $P(Z = z|u)$ by applying the Bayes' rule.
- **User Segments Identification:** The relationships among the state of the hidden variable, users and items can be used to understand common interests and preferences of users. An *user segment* is a set of users that shared a similar topic in their past sessions. The user segment corresponding to the topic z_k can be computed by selecting those users with $P(u|z_k) \geq \mu$ where the parameter μ is used as threshold. A projection of the user segment into the item space can be obtained considering the most frequent items for the user segment.
- **Topic Identification:** the pLSA model provides an easy way to identify the topic in a given user session. Given an active user u and a list of items/pages recently purchased/viewed called *session*, the probability $P(z|session)$ can be estimated via a modified version EM algorithm, known as *folding-in* approach [52], in which the the probabilities $P(i|z)$ are fixed.

In the next, we are going to discuss two novel applications of probabilistic approaches to pattern discovery in CF data. Mutual relationship between users and items can be detected by means of co-clustering approaches. The key idea is that similar users are detected by taking into account their ratings on similar items, which in turn are identified considering the ratings assigned by similar users. In Sec. 3.2, we will provide a co-clustering approach to model preference data, and then we will discuss some of its applications.

As the volume of the users grows, users profiling and segmentation techniques are becoming essential to understand their behavior and needs. Identifying groups of similar minded users is a powerful tool to develop targeted marketing campaigns. Motivated by these observations, in Sec. 3.3 we will propose a probabilistic method to detect communities of customers who exhibit the same preference patterns.

3.2 Characterizing Relationships Through Co-Clustering

In this section we present a co-clustering approach to preference prediction and rating discovery. Unlike traditional CF approaches, which try to discover similarities between users or items using clustering techniques or matrix decomposition methods, the aim of the BMM is to partition data into homogeneous blocks enforcing a simultaneous clustering which consider both the dimension of the preference data. This approach highlights the mutual relationship between users and items: similar users are detected by taking into account their ratings on similar items, which in turn are identified considering the ratings assigned by similar users.

To detect this hidden block structure within preference matrix, in the following we will provide an extension the *Block Mixture Model (BMM)* proposed in [53, 54] for binary incidence matrices. We extend the original BMM formulation to model each preference observation as the output of a gaussian mixture employing a maximum likelihood (ML) approach to estimate the parameter of the model. The strict interdependency between user and item cluster makes difficult the application of traditional optimization approaches like EM; thus, we perform approximated inference based on a variational approach and a two-step application of the EM algorithm. The block structure retrieved by the model in the inference phase allows to infer patterns and trends within each block. Overall, the proposed model guarantees a competitive prediction accuracy with regards to state-of-the art co-clustering approaches, it allows to infer topics for each item category, and finally allow as learn characteristic items for each user community, or to model community interests and transitions among topics of interests.

3.2.1 A Block Mixture Model for Preference Data

In this section, we are interested in devising how the available data fits into ad-hoc communities and groups, where groups can involve both users and items. Fig. 5.1 shows a toy example of preference data co-clustered into blocks. As we can see, a coclustering induces a natural ordering among rows and columns, and it defines blocks in the preference matrix with similar ratings. The discovery of such a structure is likely to induce information about the population, and to improve the personalized recommendations.

Formally, a *block mixture model (BMM)* can be defined by two partitions (\mathbf{z}, \mathbf{w}) which, in the case of preference data and considering known their respective dimensions, have the following characterizations:

- $\mathbf{z} = \{z_1, \dots, z_K\}$ is a partition of the user set \mathcal{U} into K clusters and $z_{uk} = 1$ if u belongs to the cluster k , zero otherwise;
- $\mathbf{w} = \{w_1, \dots, w_L\}$ is a partition of the item set \mathcal{I} into L clusters and $w_{il} = 1$ if the item i belongs to the cluster l , zero otherwise.

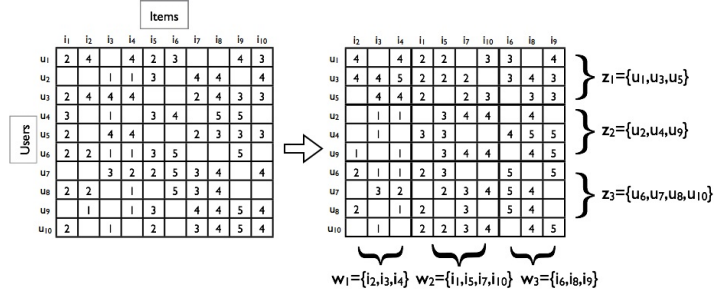


Fig. 3.1. Example Co-Clustering for Preference Data

Given a rating matrix \mathbf{R} , the goal is to determine such partitions and the respective partition functions which specify, for all pairs $\langle u, i \rangle$ the probabilistic degrees of membership wrt. to each user and item cluster, in such a way to maximize the likelihood of the model given the observed data. According to the approach described [53, 54], and assuming that the rating value r observed for the pair $\langle u, i \rangle$ is independent from the user and item identities, fixed z and w , the generative model can be described as follows:

1. For each u generate $z_u \sim \text{Discrete}(\pi_1; \dots; \pi_K)$
2. for each i generate $w_i \sim \text{Discrete}(\psi_1; \dots; \psi_L)$
3. for each pair $\langle u, i \rangle$:
 - detect k and l such that $z_{uk} = 1$ and $w_{il} = 1$
 - generate $r \sim N(R; \mu_k^l, \sigma_k^l)$

The corresponding data likelihood in the Block Mixture can be modeled as

$$p(\mathbf{R}, \mathbf{z}, \mathbf{w}) = \prod_{u \in \mathcal{U}} p(z_u) \prod_{i \in \mathcal{I}} p(w_i) \prod_{\langle u, i, r \rangle \in \mathbf{R}} P(r | z_u, w_i)$$

and consequently, the log-likelihood becomes:

$$\begin{aligned} L_c(\Theta; \mathbf{R}, \mathbf{z}, \mathbf{w}) &= \sum_{k=1}^K \sum_{u \in \mathcal{U}} z_{uk} \log \pi_k + \sum_{l=1}^L \sum_{i \in \mathcal{I}} w_{il} \log \psi_l \\ &+ \sum_{\langle u, i, r \rangle \in \mathbf{R}} \sum_k \sum_l [z_{uk} w_{il} \log \varphi(r; \mu_k^l, \sigma_k^l)] \end{aligned} \quad (3.1)$$

where Θ represents the whole set of parameters

$$\{\pi_1, \dots, \pi_K, \psi_1, \dots, \psi_L, \mu_1^1, \dots, \mu_K^L, \sigma_1^1, \dots, \sigma_K^L\},$$

and $\varphi(r; \mu, \sigma)$ is the gaussian density function on the rating value r with parameters μ and σ :

$$\varphi(r; \mu; \sigma) = (2\pi)^{-1/2} \sigma^{-1} \exp\left(\frac{-1}{2\sigma^2}(r - \mu)^2\right)$$

Inference and Parameter Estimation.

Denoting $P(z_{uk} = 1|u, \Theta^{(t)}) = c_{uk}$, $P(w_{il} = 1|i, \Theta^{(t)}) = d_{il}$ and $P(z_{uk}w_{il} = 1|u, i, \Theta^{(t)}) = e_{ukil}$, The conditional expectation of the complete data log-likelihood becomes:

$$Q(\Theta; \Theta^{(t)}) = \sum_{k=1}^K \sum_u c_{uk} \log \pi_k + \sum_{l=1}^L \sum_i d_{il} \log \psi_l + \sum_{\langle u, i, r \rangle \in \mathbf{R}} \sum_k \sum_l [e_{ukil} \log \varphi(r; \mu_k^l, \sigma_k^l)]$$

As pointed out in [53], the above function is not tractable analytically, due to the difficulties in determining e_{ukil} ; nor the adoption of its variational approximation ($e_{ukil} = c_{uk} \cdot d_{il}$) allows us to derive an Expectation-Maximization procedure for $Q'(\Theta, \Theta^{(t)})$ where the M-step can be computed in closed form. In [53] the authors propose an optimization of the complete-data log-likelihood based on the *CEM* algorithm. We adapt the whole approach here. First of all, we consider that the joint probability of a normal population x_i with $i = 1$ to n can be factored as:

$$\prod_{i=1}^n \varphi(x_i; \mu, \sigma) = h(x_1, \dots, x_n) * \varphi(u_0, u_1, u_2; \mu, \sigma),$$

where

$$h(x_1, \dots, x_n) = (2\pi)^{-n/2},$$

$$\varphi(u_0, u_1, u_2; \mu, \sigma) = \sigma^{-u_0} \exp\left(\frac{2u_1\mu - u_2 - u_0\mu^2}{2\sigma^2}\right),$$

and u_0 , u_1 and u_2 are the sufficient statistics.

Based on the above observation, we can define a two-way EM approximation based on the following decompositions of Q' :

$$Q'(\Theta, \Theta^{(t)}) = Q'(\Theta, \Theta^{(t)}|\mathbf{d}) + \sum_{i \in \mathcal{I}} \sum_{l=1}^L d_{il} \log \psi_l - \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}(u)} d_{il} / 2 \log(2\pi)$$

where

$$Q'(\Theta, \Theta^{(t)}|\mathbf{d}) = \sum_{u=1}^M \sum_{k=1}^K c_{uk} (\log(\pi_k) + \tau_{uk})$$

$$\tau_{uk} = \sum_{l=1}^L \log\left(\varphi(u_0^{(u,l)}, u_1^{(u,l)}, u_2^{(u,l)}; \mu_k^l, \sigma_k^l)\right)$$

$$u_0^{(u,l)} = \sum_{i \in \mathcal{I}(u)} d_{il}; \quad u_1^{(u,l)} = \sum_{i \in \mathcal{I}(u)} d_{il} r_i^u; \quad u_2^{(u,l)} = \sum_{i \in \mathcal{I}(u)} d_{il} (r_i^u)^2$$

Analogously,

$$\mathcal{Q}'(\theta, \theta^{(t)}) = \mathcal{Q}'(\theta, \theta^{(t)} | \mathbf{c}) + \sum_{u \in \mathcal{U}} \sum_{k=1}^K c_{uk} \log \pi_k - \sum_{i \in \mathcal{I}} \sum_{u \in \mathcal{U}(i)} c_{uk} / 2 \log(2\pi)$$

where

$$\begin{aligned} \mathcal{Q}'(\theta, \theta^{(t)} | \mathbf{c}) &= \sum_{i=1}^N \sum_{l=1}^L d_{il} (\log(\psi_l) + \tau_{il}) \\ \tau_{il} &= \sum_{k=1}^K \log \left(\varphi(u_0^{(i,k)}, u_1^{(i,k)}, u_2^{(i,k)}; \mu_k^l, \sigma_k^l) \right) \\ u_0^{(i,k)} &= \sum_{u \in \mathcal{I}(u)} c_{uk}; \quad u_1^{(i,k)} = \sum_{u \in \mathcal{I}(u)} c_{uk} r_i^u \quad u_2^{(i,k)} = \sum_{u \in \mathcal{I}(u)} c_{uk} (r_i^u)^2 \end{aligned}$$

The advantage in the above formalization is that we can approach the single components separately and, moreover, for each component it is easier to estimate the parameters. In particular, we can obtain the following:

1. **E-Step (user clusters):**

$$c_{uk} = \frac{P(u|z_k) \cdot \pi_k}{\sum_{k'=1}^K P(u|z_{k'}) \cdot \pi_{k'}} \quad P(u|z_k) = \prod_{l=1}^L \varphi(u_0^{(u,l)}, u_1^{(u,l)}, u_2^{(u,l)}; \mu_k^l, \sigma_k^l)$$

2. **M-Step (user clusters):**

$$\begin{aligned} \pi_k &= \frac{\sum_{u \in \mathcal{U}} c_{uk}}{M} \\ \mu_k^l &= \frac{\sum_{u=1}^M \sum_{i \in \mathcal{I}(u)} c_{uk} d_{il} r_i^u}{\sum_{u=1}^M \sum_{i \in \mathcal{I}(u)} c_{uk} d_{il}} \quad (\sigma_k^l)^2 = \frac{\sum_{u=1}^M \sum_{i \in \mathcal{I}(u)} c_{uk} d_{il} (r_i^u - \mu_k^l)^2}{\sum_{u=1}^M \sum_{i \in \mathcal{I}(u)} c_{uk} d_{il}} \end{aligned}$$

3. **E-Step (item clusters):**

$$d_{il} = \frac{P(i|w_l) \cdot \psi_l}{\sum_{l'=1}^L P(i|w_{l'}) \cdot \psi_{l'}} \quad P(i|w_l) = \prod_{k=1}^K \varphi(u_0^{(i,k)}, u_1^{(i,k)}, u_2^{(i,k)}; \mu_k^l, \sigma_k^l)$$

4. **M-Step (item clusters):**

$$\begin{aligned} \psi_l &= \frac{\sum_{i \in \mathcal{I}} d_{il}}{N} \\ \mu_k^l &= \frac{\sum_{i=1}^N \sum_{u \in \mathcal{U}(i)} d_{il} c_{uk} r_i^u}{\sum_{i=1}^N \sum_{u \in \mathcal{U}(i)} d_{il} c_{uk}} \quad (\sigma_k^l)^2 = \frac{\sum_{i=1}^N \sum_{u \in \mathcal{U}(i)} c_{uk} d_{il} (r_i^u - \mu_k^l)^2}{\sum_{i=1}^N \sum_{u \in \mathcal{U}(i)} d_{il} c_{uk}} \end{aligned}$$

Rating Prediction

The blocks resulting from a co-clustering can be directly used for prediction. Given a pair $\langle u, i \rangle$, the probability of observing a rating value r associated to the pair $\langle u, i \rangle$ can be computed according to one of the following schemes:

- *Hard-Clustering Prediction:*
 $P(r|i, u) = \varphi(r; \mu_k^l, \sigma_k^l)$, where $k = \operatorname{argmax}_{j=1, \dots, K} c_{uj}$ and $l = \operatorname{argmax}_{h=1, \dots, L} d_{ih}$ are the clusters that better represent the observed ratings for the considered user and item respectively.
- *Soft-Clustering Prediction:*
 $P(r|i, u) = \sum_{k=1}^K \sum_{l=1}^L c_{uk} d_{il} \varphi(r; \mu_k^l, \sigma_k^l)$, which consists of a weighted mixture over user and item clusters.

The final rating prediction can be computed by using the expected value of $P(r|u, i)$.

In order to test the predictive accuracy of the BMM we performed a suite of tests on a sample of Netflix data. The training set contains 5,714,427 ratings, given by 435,656 users on a set of 2,961 items (movies). Ratings on those items are within a range 1 to 5 (max preference value) and the sample is 99% sparse. The test set contains 3,773,781 ratings given by a subset of the users (389,305) in the training set over the same set of items. Over 60% of the users have less than 10 ratings and the average number of evaluations given by users is 13.

We evaluated the performance achieved by the BMM considering both the Hard and the Soft prediction rules and performed a suite of experiments varying the number of user and item clusters. Experiments on the three models have been performed by retaining the 10% of the training (user,item,rating) triplets as held-out data and 10 attempts have been executed to determine the best initial configurations. Performance results measured using the RMSE for two BMM with 30 and 50 user clusters are showed in Figure 3.2(a) and Figure 3.2(b), respectively. In both cases the soft clustering prediction rule overcomes the hard one, and they show almost the same trend. The best result (0.9462) is achieved by employing 30 user clusters and 200 item clusters. We can notice from Table 3.1 that the results follow the same trend as other probabilistic models (pLSA[28], FMM [37], ScalableCC[55]).

Method	Best RMSE	K	H
BMM	0.946	30	200
PLSA	0.947	30	-
FMM	0.954	10	70
Scalable CC	1.008	10	10

Table 3.1. Prediction Accuracy.

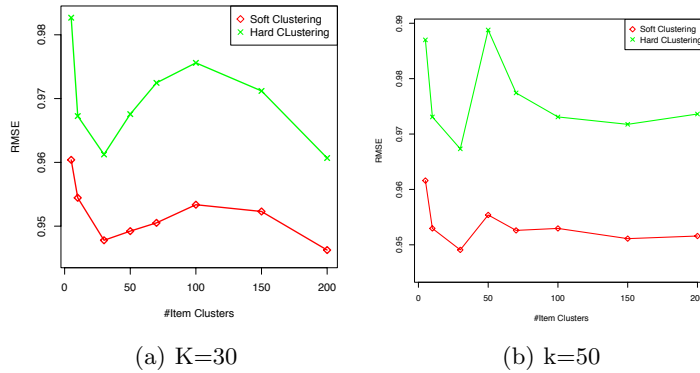


Fig. 3.2. Predictive Accuracy of BMM

3.2.2 Pattern Discovery using BMM.

The probabilistic formulation of the BMM provides a powerful framework for discovering hidden relationships between users and items. As exposed above, such relationships can have several uses in users segmentation, product catalog analysis, etc. Several works have focused on the application of clustering techniques to discover patterns in data by analyzing user communities or item categories. The co-clustering structure proposed so far increases the flexibility in modeling both user communities and item categories patterns. Given two different user clusters which group users who have showed a similar preference behavior, the BMM allows the identification of common rated items and categories for which the preference values are different. For example, two user community might agree on action movies while completely disagree on one other. The identification of the topics of interest and their sequential patterns for each user community lead to an improvement of the quality of the recommendation list and provide the user with a more personalized view of the system. In the following we will discuss examples of pattern discovery and user/item profiling tasks on MovieLens data.

Co-Clustering Analysis

The relationships between groups of users and items captured by the BMM can be easily recognized by analyzing the distribution of the preference values for each cocluster. Given a co-cluster $\langle k, l \rangle$, we can analyze the corresponding distribution of rating values to infer the preference/interest of the users belonging to the community k on item of the category l . Figure 3.3 shows graphically a block mixture model with 10 users clusters and 9 item clusters built on the MovieLens dataset. A hard clustering assignment has been performed both on users and clusters: each user u has been assigned to the cluster

c such that $c = \operatorname{argmax}_{k=1, \dots, K} c_{uk}$. Symmetrically, each item i has been assigned to the cluster d such that: $d = \operatorname{argmax}_{l=1, \dots, L} d_{il}$. The background color of each block

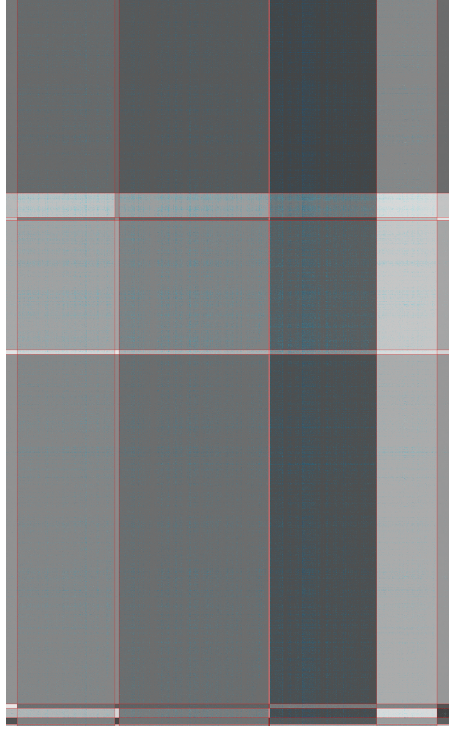


Fig. 3.3. Coclustering

$\langle k, l \rangle$ describes both the density of ratings and the average preference values given by the users (rows) belonging to the k -th group on items (columns) of the l -th category: the background intensity increases with the average rating values of the coclusters, which are given in Table 3.2. Each point within the coclusters represents a rating, and again an higher rating value corresponds to a more intense color. The analysis underlines interesting tendencies: for example, users belonging to the user community c_1 tend to assign higher rating values than the average, while items belonging to item category d_6 are the most appreciated. Two interesting blocks of the whole image are further analyzed in Figure 3.4(a) and in Figure 3.4(b). Here, two blocks are characterized by opposite preference behaviors: the first block contains few (low) ratings, whereas the second block exhibits a higher density of high value ratings.

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
c_1	3.4	3.59	3.59	4	2.91	4.43	3.59	2.93	3.65
c_2	2.23	2.2	2.92	2.79	2	3.45	2.07	1.80	2.51
c_3	2.11	3.24	3	3.66	2	4.17	1	1.03	5
c_4	2.45	2.69	2.54	3.2	2.43	3.74	2.51	2	2.56
c_5	1	1.79	1	2.32	1	2.98	1.66	1	1.75
c_6	2.93	3.07	3	3.57	2.20	4.09	2.9	2.3	3.16
c_7	1	3.56	3.9	3.7	3.64	3.39	4	3.49	2
c_8	2.25	2.26	1.62	3.27	1	4.17	4.54	1	2.45
c_9	4.08	3.24	4.40	3.54	5	4	3.71	4.5	5
c_{10}	1.91	2.82	1	2.7	4.3	2.2	1	4	2

Table 3.2. Gaussian Means for each block

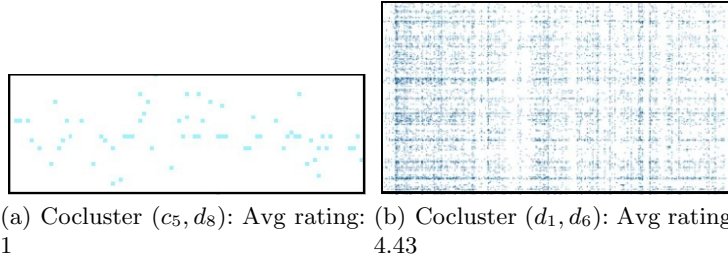


Fig. 3.4. Cocluster Analysis

Item-Topic Analysis

A structural property of of interest is the item-topic dependency. Given a set of F topics $\mathcal{G} = \{g_1, \dots, g_F\}$ and assuming that each item is tagged with at least one of those, we can estimate the relevance of each topic within item clusters through a variant of the *tf-idf* measure [56], namely *topic frequency - inverse category frequency (tf-icf)*.

The *topic frequency* (similar to the *term frequency*) of a topic g in a cluster d_l can be defined as:

$$tf_{g,d_l} = \frac{\sum_{i \in d_l} \frac{\delta(g \in \mathcal{Q}_i)}{|\mathcal{Q}_i|} \sum_{u \in \mathcal{U}} \delta(u, i)}{\sum_{g'=1}^F \sum_{i \in d_l} \frac{\delta(g' \in \mathcal{Q}_i)}{|\mathcal{Q}_i|} \sum_{u \in \mathcal{U}} \delta(u, i)}$$

In a scenario, where items are associated with several topics (genres), and where the number of topics is much lower than size of the itemset, it is high likely that all topics appear at least one in each item category. According to this consideration, the standard definition of *idf* would be useless for our purposes. We, hence, provide an alternative formulation based on entropy [57], namely *inverse category frequency (icf)* for a topic g is:

$$icf_g = 1 + P(g) \log_2[P(g)] + [1 - P(g)] \log_2[1 - P(g)]$$

Here, $P(g)$ represent the prior probability of observing a item-genre and is computed as $P(g) = \sum_{l=1}^L P(g|d_l) \cdot P(d_l)$, where $P(g|d_l) = tf_{g,d_l}$ and $P(d_l) =$

ψ_l . By combining the above definitions we can finally obtain the *tf-icf* measure for a topic g in a category d_l as:

$$tf\text{-}icf_{g,d_l} = tf_{g,d_l} \times icf_g$$

We can also exploit the fact that BMM provides a soft assignment to clusters, and provide an alternative version of *tf* as:

$$tf_{g,d_l} = \frac{\sum_{i \in d_l} \frac{\delta(g \in \mathcal{Q}_i)}{|\mathcal{Q}_i|} \cdot d_{il} \sum_{u \in \mathcal{U}} \delta(u, i)}{\sum_{g'=1}^F \sum_{i \in d_l} \frac{\delta(g' \in \mathcal{Q}_i)}{|\mathcal{Q}_i|} \cdot d_{il} \sum_{u \in \mathcal{U}} \delta(u, i)}$$

The above considerations can be also applied to the case of item frequency:

$$if_{i,d_l} = \frac{d_{il} \sum_{u \in \mathcal{U}} \delta(u, i)}{\sum_{i' \in d_l} d_{i'l} \sum_{u \in \mathcal{U}} \delta(u, i')}$$

$$icf_i = 1 + P(i) \log_2[P(i)] + [1 - P(i)] \log_2[(1 - P(i))]$$

where:

$$P(i) = \frac{|\mathcal{U}(i)|}{|\mathcal{U}|}$$

The topic and item relevance described so far can be directly employed to identify and measure the interest of each user community into topics and items. More specifically, we can measure the interest of a user community c_k for a topic g as:

$$CI_t(c_k, g) = \frac{\sum_{l=1}^L \mu_k^l \cdot tf\text{-}icf_{g,d_l}}{\sum_{g'=1}^F \sum_{l=1}^L \mu_k^l \cdot tf\text{-}icf_{g',d_l}}$$

The item-based counterpart follows straightforwardly:

$$CI_i(c_k, j) = \frac{\sum_{l=1}^L \mu_k^l \cdot if\text{-}icf_{j,d_l}}{\sum_{j'=1}^F \sum_{l=1}^L \mu_k^l \cdot if\text{-}icf_{j',d_l}}$$

where j is the item target.

Evaluation

The MovieLens dataset provides for each movie a list of genres. This information can be used to characterize each item category, by exploiting the within-cluster topic relevance discussed so far. The *tf-icf* measure of observing each genre within each item category is given in Table 3.3, where the dominant topic is in bold.

The pie charts in Figure 3.5(a), Figure 3.5(b) and Figure 3.5(c) show the distribution on topics for different item clusters. We can observe different patterns: d_2 is characterized by a strong attitude for horror movies, animation

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
Action	0.03640	0	0.07375	0.06054	0.05152	0	0.05624	0.06966	0
Adventure	0.01981	0	0.04237	0.04339	0.03813	0	0.03828	0	0
Animation	0.01591	0	0.00660	0.00926	0.01801	0.24622	0.00999	0	0
Children's	0.01581	0	0.03228	0.01643	0.02261	0	0.02855	0	0
Comedy	0.04137	0.03559	0.05403	0.05185	0.04730	0.06209	0.05685	0.10228	0
Crime	0.03319	0	0.01585	0.02217	0.01973	0	0.02515	0	0
Documentary	0.01423	0	0.00028	0.00053	0.00291	0	0.00341	0	0.94466
Drama	0.09777	0.00923	0.02308	0.05247	0.07720	0.04839	0.05099	0.06727	0
Fantasy	0.00553	0	0.01175	0.01579	0.01171	0	0.01559	0	0
Film-Noir	0.01485	0	0.00029	0.00123	0.00580	0	0.00113	0	0
Horror	0.01570	0.53057	0.08225	0.02691	0.01569	0	0.04014	0.03426	0
Musical	0.01739	0	0.00619	0.00914	0.02224	0	0.01088	0	0
Mystery	0.01697	0	0.00832	0.02757	0.00958	0	0.00952	0	0
Romance	0.03470	0	0.02395	0.05776	0.05092	0.09889	0.04625	0	0
Sci-Fi	0.02818	0	0.06247	0.04644	0.03843	0	0.04150	0	0
Thriller	0.04613	0	0.05851	0.05052	0.04771	0	0.05057	0	0
War	0.03902	0	0.01268	0.01041	0.01442	0.12291	0.00716	0.11860	0
Western	0.01653	0	0.00625	0.00704	0.00641	0	0.00875	0	0

Table 3.3. *tf-icf* measures for each genre in each movie category

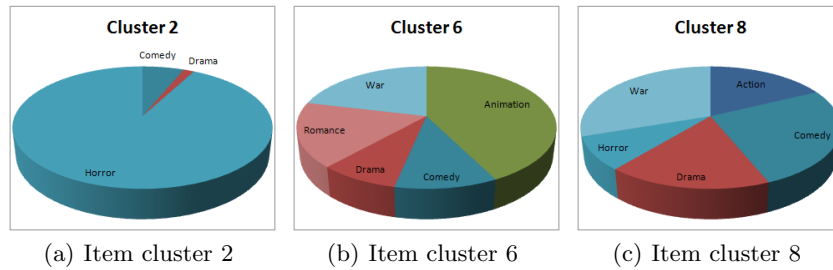


Fig. 3.5. Topic Analysis on Item Clusters

is the dominant topic in cluster 6, and d_8 is summarized by the war genre. Finally, the cluster d_9 shows a predominance of drama movies. A summary of the dominant genres in each item cluster, i.e., with higher *tf-icf*, is given below:

Item Cluster	Dominant Genre
d_1	Drama
d_2	Horror
d_3	Horror
d_4	Action
d_5	Drama
d_6	Animation
d_7	Comedy
d_8	War
d_9	Documentary

Figure 3.6 shows the $CI_t(g, c_k)$ values (in gray scale). We can further analyze such values to infer the interest of a user community for a given topic. In particular, a community exhibits a high interest for a topic if the corresponding CI_t value is sufficiently higher than the average CI_t value of all the

other topics. Table 3.4 summarizes the associations among user communities and item topics. For example, users in c_8 exhibit preferences for the *Action* and *War* genres.

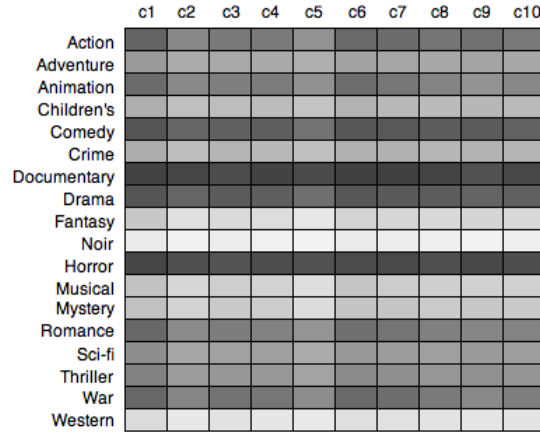


Fig. 3.6. Topic-Interests for User Communities

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
Action	y		y					y	y	y
Advent.										
Animat.			y							y
Children's										
Comedy	y	y	y	y	y	y	y	y	y	y
Crime										
Documen.	y	y	y	y	y	y	y	y	y	y
Drama	y	y	y	y	y	y	y	y	y	y
Fantasy										
Noir										
Horror	y	y	y	y	y	y	y	y	y	y
Musical										
Mystery										
Romance			y						y	y
Sci-Fi										
Thriller										
War			y					y		y
Western										

Table 3.4. Summary of Interests in Topics For User Communities

User Profile Segmentation

The topics of interest of a user may change within time and consecutive choices can influence each other. We can analyze such temporal dependencies by mapping each user's choice into their respective item cluster. Assume that movieLens data can be arranged as a set $\{\bar{u}_1, \dots, \bar{u}_M\}$, where

$\bar{u} = \{\langle r_i^u, i, t_i^u \rangle \forall i \in \mathcal{I}(u)\}$ and t_i^u is the timestamp corresponding to the rating given by the user u on the item i . By chronologically sorting \bar{u} and segmenting it according to item cluster membership, we can obtain a view of how user's tastes change over time. Three example of user profile segmentation are given in the figures below (the mapping between item categories and colors is given by the included table).

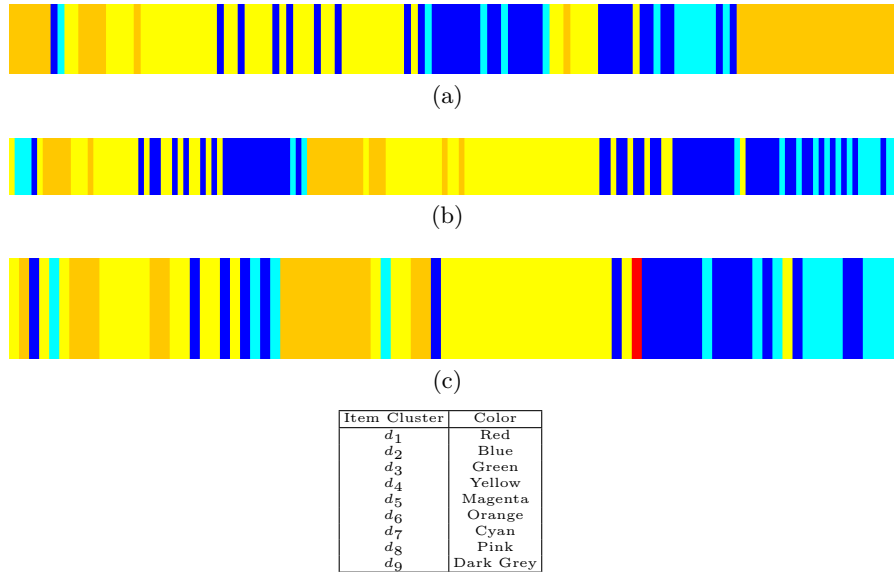


Fig. 3.7. User Profile Segmentation

In practice, we can assume that the three users show a common attitude towards comedy and drama, which are the dominant topics corresponding to the colors yellow and orange. Notice, however, that users (b) and (c) are prone to change their interest towards comedy, as clearly shown by the change in color.

Modeling Topic Transitions

Based on the above observations, we aim at estimating the sequential connections among topics: In practice, we would like to analyze which item categories are likely to next capture the interests of a user. Those sequential patterns can be modeled by exploiting *Markov Models*. The latter are probabilistic models for discrete processes characterized by the Markov properties. We adopt a Markov Chain property here, i.e., a basic assumption which states that any future state only depends from the present state. This property limits the ‘memory’ of the chain which can be represented as a digraph where nodes

represent the actual states and edges represent the possible transitions among them.

Assuming that the last observed item category for the considered user is d_i , the user could pick an item belonging to the another topic d_j with probability $p(d_j|d_i)$. Thus, we need to estimate all the *transition probabilities*, starting from a $|L + 1| \times |L + 1|$ *transition count matrix* \mathcal{T}_c , where $\mathcal{T}_c(i, j)$ stores the number of times that category j follows i in the rating profile of the users.¹

The estimation we provide is rather simple, corresponding to a simple frequency count:

$$p(d_j|d_i) = \frac{\mathcal{T}_c(i, j)}{\sum_{j=1}^{L+1} \mathcal{T}_c(i, j')}$$

Figure 3.8 represents the overall transition probability matrix, which highlights some strong connection among given categories. As instance, the item categories having drama as dominant genre, d_4, d_6 and d_9 are highly correlated as well as d_2, d_7 and d_8 which correspond to comedy movies.

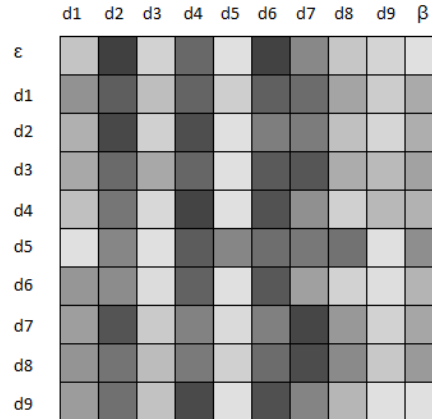


Fig. 3.8. Item Categories Transitions

It is interesting to compare how the transition probabilities change within different user communities. Figure 3.9 shows the transitions for three different communities. Notice that, besides common transition patterns, each community has some distinctive transitions that characterize their population. For all the considered user communities, the most likely initial item category is d_6 ; while the first and the last community reproduced in the example show a strong attitude corresponding to the transition $d_8 \rightarrow d_2$, this is instead a weak pattern within c_7 . The same consideration can be done for the transition

¹ We assume two further states ϵ , representing the initial choice, and β , representing the last choice.

$d_9 \rightarrow d_7$, which is strong for c_7 and c_{10} , while users belonging to c_3 are more prone to the transition towards d_6 .

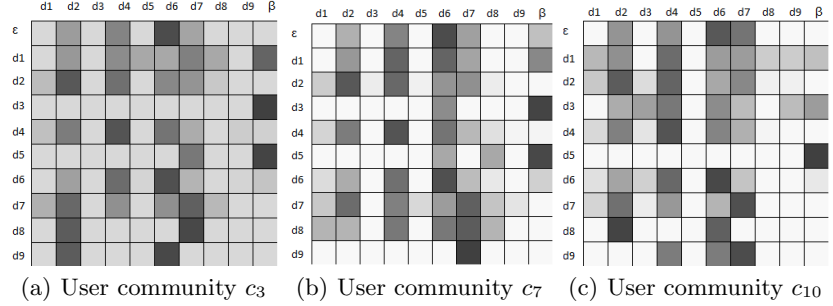


Fig. 3.9. Community-based Item Category Transitions

The analysis of the transition probabilities can be hence exploited for generating new recommendations enforcing topic diversity in the top- K lists of items by taking into account not exclusively the current topic of interest but the ones that more likely could be connected to it.

3.3 Users' Profiling with Soft Constraints

The *User Rating Profile (URP)* [38] is an extension of the LDA which provides a full generative semantic for explicit preference data. The model is defined by a mixture of the so called *user attitudes* which represent abstract rating patterns. Each user is represented by its own distribution over attitudes, namely ϑ_u , which specifies a probability of selecting each of the K possible topics z . As in LDA, this distribution is not a parameter of the model but it is a random variable sampled from a Dirichlet distribution with parameter α . Each topic is then characterized by a multinomial distribution over rating values Φ_z . We extend the original formulation of the model by employing Dirichlet priors β over the Φ distribution, that should help in improving the estimate of rating probabilities for items which have received few evaluations. The model is characterized by the following generative process:

1. For each user $u \in \mathcal{U}$ sample user community-mixture components $\vartheta_u \sim Dir(\alpha)$
2. For each user attitude $z = \{1, \dots, K\}$,
 - a) For each item (i) sample rating probabilities $\varphi_{z,i} \sim Dir(\beta)$
3. For $i \in \mathcal{I}(u)$

- a) Choose a user attitude $z \sim Discrete(\boldsymbol{\vartheta}_u)$
- b) Generate a rating value for the chosen item according to the distribution $P(R|\boldsymbol{\varphi}_{z,i})$

A graphical representation of the parameters of the URP model is given in Figure 3.10. The projection into the latent space of each user, $\boldsymbol{\vartheta}_u$, can be used

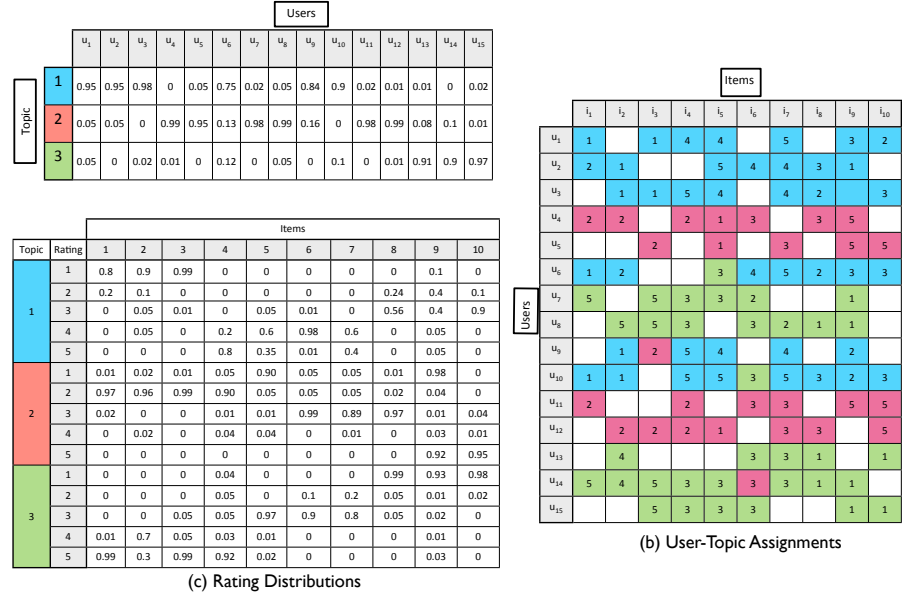


Fig. 3.10. URP Example

to detect groups of users which share the same interests or preference pattern. In fact, each topic is defined by a set of users who exhibit the same rating patterns and thus can be used to detect community-memberships, where each user community gathers users who tend to express the same preference value on the same item. However, the generative presented above relies exclusively on the observed users' preferences, while we may be interested in retrieving communities focusing on additional information. As instance, we extend this definition of user community to capture both users who tend to agree in their preferences over items, and users who tend to select the same set of items, no matter the subjective preference. To include such information, we can devise a regularization procedure which is embedded in the learning phase of the model. In the following, we will first derive a Gibbs Sampling parameter

estimation for the URP model and then design a regularization procedure to enforce a set of soft constraints. Finally, we will evaluate the prediction accuracy of the overall approach and the effectiveness of the regularization procedure.

Parameter Estimation.

Given the hyperparameters α , β , the joint distribution of the data \mathbf{R} , the user-topic mixture Θ , the rating probabilities Φ and the observation-topic assignments \underline{Z} , can be computed as:

$$P(\mathbf{R}, \underline{Z}, \Theta, \Phi | \alpha, \beta) = P(\underline{Z} | \Theta) P(\Theta | \alpha) P(\Phi | \beta) P(\mathbf{R} | \underline{Z}, \Phi)$$

The complete data likelihood can be obtained by integrating over Θ and Φ :

$$P(\mathbf{R}, \underline{Z} | \alpha, \beta) = \int \int P(\underline{Z} | \Theta) P(\Theta | \alpha) P(\Phi | \beta) P(\mathbf{R} | \underline{Z}, \Phi) d\Theta d\Phi$$

which, due to the conditional independence $\mathbf{R} \perp\!\!\!\perp \alpha | \underline{Z}$, can be factored as:

$$P(\mathbf{R}, \underline{Z} | \alpha, \beta) = \int P(\underline{Z} | \Theta) P(\Theta | \alpha) d\Theta \int P(\mathbf{R} | \underline{Z}, \Phi) P(\Phi | \beta) d\Phi$$

The first term of the complete data likelihood, $P(\underline{Z}, | \Theta)$ represents the probability of observing the topic-assignments matrix \underline{Z} given the multinomial parameters Θ and can be computed as:

$$P(\underline{Z} | \Theta) = \prod_{u \in \mathcal{U}} \prod_{i \in \mathcal{I}} p(z_{u,i} | \vartheta_u) = \prod_{u \in \mathcal{U}} \prod_{k=1}^K \vartheta_{u,k}^{n_u^k}$$

where n_u^k denote the number of times that the topic k has been assigned to observations corresponding to the user u and $\vartheta_{u,k} = P(z_k | u)$. The k -dimensional Dirichlet random variable Θ can take values in the $(k-1)$ simplex, and the corresponding k -dimensional distribution is:

$$P(\Theta | \alpha) = \prod_{u \in \mathcal{U}} \frac{1}{\Delta(\alpha)} \prod_{k=1}^K \vartheta_{u,k}^{\alpha_k - 1}$$

where $\Delta(\alpha) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)}$ is introduced for notational convenience.

The likelihood of the rating matrix \mathbf{R} given the topic assignments \underline{Z} and the distribution over rating values Φ can be computed as:

$$P(\mathbf{R} | \underline{Z}, \Phi) = \prod_{u \in \mathcal{U}} \prod_{i \in \mathcal{I}} \prod_{r=1}^V \varphi_{z_{u,i}, i, r} = \prod_{k=1}^K \prod_{i \in \mathcal{I}} \prod_{r=1}^V \varphi_{k, i, r}^{n_{i,r}^k}$$

where $n_{i,r}^k$ denotes the number of times that the rating r has been assigned to the item i when the topic is k and $\varphi_{k, i, r} = P(R = r | z_k, i)$. Again, Φ is multinomial distribution with Dirichlet priors β :

$$P(\Phi|\beta) = \prod_{i \in \mathcal{I}} \prod_{k=1}^K \frac{1}{\Delta(\beta)} \prod_{r=1}^V \varphi_{k,i,r}^{\beta_r-1}$$

Thus, rearranging the components and grouping Dirichlet-Multinomial distributions, the complete data likelihood can be expressed as:

$$P(\mathbf{R}, \underline{Z}|\alpha, \beta) = \prod_{u \in \mathcal{U}} \frac{1}{\Delta(\alpha)} \int \prod_{k=1}^K \vartheta_{u,k}^{n_u^k + \alpha_k - 1} d\vartheta_u \\ \prod_{i \in \mathcal{I}} \prod_{k=1}^K \frac{1}{\Delta(\beta)} \int \prod_{r=1}^V \varphi_{k,i,r}^{n_{i,r}^k + \beta_r - 1} d\varphi_{ik}$$

The formulation of the joint distribution is the starting point for the inference: in fact, the distribution $P(\underline{Z}|\mathbf{R})$ whose determination is the target of the inference process is directly proportional to the joint distribution:

$$P(\underline{Z}|\mathbf{R}, \alpha, \beta) = \frac{P(\underline{Z}, \mathbf{R}|\alpha, \beta)}{P(\mathbf{R}|\alpha, \beta)}$$

which however is intractable mainly because the computation of the denominator requires a summation over an exponential number of terms. The Gibbs Sampling procedure addresses this point by using the full conditional $P(Z_n|\underline{Z}_{-n}, \mathbf{R}, \alpha, \beta)$ in order to simulate $P(\underline{Z}|\mathbf{R}, \alpha, \beta)$, where n denotes a single rating observation $n = \{u, i, r_i^u\}$, Z_n is the cell of the matrix \underline{Z} which corresponds to this observation, while \underline{Z}_{-n} denotes the remaining topic assignments. The Gibbs Sampling algorithm estimates the probability of assigning the topic k to the observation n -th. given the assignment corresponding to all the other rating observations as:

$$P(Z_n = k|\underline{Z}_{-n}, \mathbf{R}) \propto \frac{P(\underline{Z}, \mathbf{R})}{P(\underline{Z}_{-n}, \mathbf{R}_{-n})} \\ = \frac{n_{u-n}^k + \alpha_k}{\sum_{k'=1}^K (n_{u-n}^{k'} + \alpha_{k'}) - 1} \frac{n_{i,r-n}^k + \beta_r}{\sum_{r'=1}^V (n_{i,r-n}^{k'} + \beta_{r'}) - 1} \quad (3.2)$$

The multinomial parameters Θ and Φ can be obtained according to their definition and considering the expectation of the corresponding prior distributions:

$$\vartheta_{u,k} = \frac{n_u^k + \alpha_k}{\sum_{k'=1}^K n_u^{k'} + \alpha_{k'}} \quad (3.3)$$

and

$$\varphi_{k,i,r} = \frac{n_{i,r}^k + \beta_r}{\sum_{r'=1}^V n_{i,r'}^k + \beta_{r'}} \quad (3.4)$$

A summary of symbols used in the derivation of the model is given in Table 3.5. Algorithm 6 shows the pseudocode for the inference phase: the Gibbs Sampling procedure starts with a random initialization; then topic assignments

are estimated till convergence (error on a held-out set starts to increase) or till the number of performed iteration reaches the maximum value. Hyperparameters can be updated employing iterative approximation (see [58] for further details).

Algorithm 1 URP-GibbsSampling

Require: The sets $\mathcal{U} = \{u_1, \dots, u_M\}$ and $\mathcal{I} = \{i_1, \dots, i_N\}$
 the rating matrix \mathbf{R} , the number of latent topics K , initial hyperparameters α and β .

- 1: *initializeTopicAssignments()*
- 2: *iteration* \leftarrow 0
- 3: *converged* \leftarrow *false*
- 4: **while** *iteration* $<$ *nMaxIterations* & *!converged* **do**
- 5: **for all** $\langle u, i, r \rangle \in \mathbf{R}$ **do**
- 6: $z_{u,i} \leftarrow$ *topicAssignments.get*(u, i)
- 7: $z'_{u,i} \leftarrow$ *sampleTopic*(u, i, r) {According to Eq. 5.10};
- 8: *topicAssignments.set*($u, i, z'_{u,i}$)
- 9: update counts using the new topic for the observation $\langle u, i, r \rangle$
- 10: **end for**
- 11: *updateHyperParams()*
- 12: **if** *iteration* $>$ *burnin* & *iteration* $\%sampleLag == 0$ **then**
- 13: *sampleMultinomials()* {According to Eq. 5.17 and Eq. 3.4};
- 14: *converged* \leftarrow *checkConvergence()* {Current RMSE on HeldOut data $>$ Previous RMSE};
- 15: **end if**
- 16: *iteration* \leftarrow *iteration* + 1
- 17: **end while**

Rating Prediction.

The main goal of the model is to predict preference values for the items that users have not rated yet. Given a pair $\langle u, i \rangle$, the distribution over ratings corresponding to that observation can be computed by using the parameter estimated during the learning phase:

$$P(R = r|u, i) = \sum_{k=1}^K P(z_k|u)P(r|z_k, i) = \sum_{k=1}^K \vartheta_{u,k} \cdot \varphi_{k,i,r}$$

The final preference value can be computed from this distribution according to different schema:

- *Most Probable Rating:* $r_i^u = \operatorname{argmax}_{r=1, \dots, V} P(R = r|u, i)$
- *Median Rating:* $r_i^u : P(R \leq r|u, i) \geq \frac{1}{2}$ and $P(R \geq r|u, i) \leq \frac{1}{2}$
- *Expected Rating:* $r_i^u = E[R|u, i]$

Symbol	Description
M	#users
N	#items
V	#stars
K	#latent factors / user attitudes
α	K -vector; parameters of the Dirichlet prior over topics
ϑ_u	K -vector; Multinomial distribution over topics for the user u : $\vartheta_{u,k} = P(z_k u)$
Φ	$K \times N \times V$ matrix which encodes the parameters of the multinomial distribution: $\varphi_{i,k,r} = P(R = r z_k, i)$
β	V -vector; parameters of the Dirichlet prior over rating values
\underline{Z}	$M \times N$ matrix which represents hidden topic assignments for each pair $\langle u, i \rangle$
Z_n	topic assignment for the pair $n = \langle u, i \rangle$
\underline{Z}_{-n}	topic assignments for all the pairs $\langle u, i \rangle$ excluding the observation n
n_u^k	# times that the topic k has been assigned to observation corresponding to the user u
$n_{i,r}^k$	# times that the topic k has been assigned to the item i when the associated rating is r

Table 3.5. Summary of Symbols

The latter method, which computes the prediction for the pair $\langle u, i \rangle$ as the expected rating, is the most used, mainly because it minimizes MSE, and thus RMSE. The distribution of probability over ratings represents an important advantage of probabilistic approaches, because it can be used to infer confidence intervals and variance which can in turn be used to estimate the confidence of the prediction itself and thus improve the recommendation accuracy.

3.3.1 Community Detection and Neighborhood Regularization.

Probabilistic approaches based on latent topics, such as PLSA and LDA, provide a powerful framework for community detection and analysis. According to the parameters of the URP-model, similar minded users can be detected by analyzing their distribution over topics, which represent abstract interests and rating patterns. More specifically, we can partition users into homogenous groups, named *user communities*, by considering the dominant topic in their respective profile:

$$cluster(u) = \operatorname{argmax}_{k=1, \dots, K} \vartheta_{u,k}$$

Once user communities have been discovered, we can infer their characteristics and subjective preferences over products in the catalog, by detecting

significant items. An item i is considered significant for the community k if $P(i|z_k) > P(i|z_{k'}) \forall k' \neq k$, where the probability of observing an item given a topic z_k can be estimated at the end of the inference phase as:

$$p(i|z_k) \propto \sum_{\langle u,r \rangle \in \mathbf{R}} P(r|z_k, i) p(z_k|u) = \sum_{\langle u,r \rangle \in \mathbf{R}} \varphi_{k,i,r} \cdot \vartheta_{u,k} \quad (3.5)$$

One of the key differences between neighborhood and latent factor models relies on the ability of detecting hidden relationships among users and/or items. In fact, while Neighborhood based models are good at detecting strong but local relationships due to the direct estimate of the similarity between objects, latent factor models are able to detect weak but global patterns, defined as topic-specific multinomial distribution over rating values. As consequence, using the direct estimate of similarity coefficients, neighborhood models induce a more intuitive clustering (they are based on an intuitive notion of similarity), while the latter approaches achieve a better prediction accuracy because cluster and their corresponding probability distributions are obtained employing a likelihood maximization approach.

Assume that we are given external evidence of strong relationships between users; we can employ a soft constraints regularization procedure to push the Gibbs Sampling towards a clustering solution that preserves neighborhoods. This background information can be specified as a set of soft constraints in the form $\langle u, \mathcal{N}_u^K \rangle$, where \mathcal{N}_u^K is the set of the K most similar users to u according to a specified similarity measure (Pearson, Cosine and Jaccard are the most used).

The key assumption in the following approach is that if two user are similar, then they will likely associate the same hidden topic z to the same item. We can adopt the procedure summarized in Algorithm 2 to regularize hidden topic assignments.

Given two user neighbors u and u' who have rated the same item i , the hidden topic corresponding to the observation of i can be computed by sampling from the parameters of the user who have been purchased more items, because he is better represented in the topic space. Topic regularization is performed for a limited number of observations; when the prediction error computed after the regularization exceeds significantly the one's computed before, the regularization is rejected. An alternative regularization schema could be obtained according to a simulated annealing procedure, where regularization steps which deteriorate prediction accuracy could be also accepted and the effect of regularization decreases as the number of iteration increases. Moreover, the same protocol can be used to perform regularization on items, by assuming that similar items will be likely to receive the same topic assignment by the same user.

To measure the effect of the regularization, we can compute for each user u the percentage of his neighbors mapped in the same cluster as:

$$\frac{\sum_{u' \in \mathcal{N}_u^K} \delta(u, u')}{K}$$

Algorithm 2 performRegularization

```

1:  $E \leftarrow \text{computeError}()$ 
2:  $\text{regularizedPairs} \leftarrow 0$ 
3: while  $\text{regularizedPairs} < \text{max}$  do
4:   choose a random pair  $\langle u, i \rangle$ 
5:   choose  $u' \in N_u^K : r_i^{u'} \neq 0$ 
6:   if  $|\mathcal{I}(u')| > |\mathcal{I}(u)|$  then
7:      $z_{u,i} \leftarrow \text{sampleTopic}(u', i, r_i^{u'})$ 
8:   else
9:      $z_{u',i} \leftarrow \text{sampleTopic}(u, i, r_i^{u'})$ 
10:  end if
11:   $\text{regularizedPairs} \leftarrow \text{regularizedPairs} + 1$ 
12: end while
13:  $\text{updateMultinomials}()$ 
14:  $E' \leftarrow \text{computeError}()$ 
15: if  $E' - E > 10e - 5$  then
16:    $\text{undoRegularization}()$ 
17: end if

```

where $\delta(u, u') = 1$ iff $\text{cluster}(u) = \text{cluster}(u')$, zero otherwise. Then, the average percentage of neighbors preserved can be computed by averaging over users.

3.3.2 Experimental Evaluation.

We evaluate the effectiveness of the proposed approach considering two perspectives:

- *Prediction Accuracy*: we are interested in evaluating whether the URP model with Gibbs Sampling procedure for parameter estimation achieves a competitive prediction accuracy with respect to the variational version and considering other similar probabilistic approaches;
- *Effect of Regularization*: we are interesting in measuring how much the clustering solution induced by the Regularized-URP model maintains the neighborhoods provided as external evidence and evaluating possible effects of regularization on prediction accuracy.

Prediction accuracy is measured as the RMSE and we performed a MonteCarlo 5-fold validation on MovieLens-1M data, where for each fold the training-set contains about the 80% of overall ratings. Information about the number of users, items, sparseness coefficient, and average rating per users/items regarding the first fold are summarized in Table 3.6. All the considered approaches have been trained over the training-data, employing an early stopping criterion, i.e. the learning phase is terminated as soon as the prediction error measured on a held-out data (%1 of the training) increases. The Gibbs-Sampler burn in period is fixed at 300 iterations; after that period the procedure samples multinomial parameters with an interval of 10

	Training Set	Test Set
Users	6,040	6,032
Items	3,706	3,444
Ratings	800,729	199,480
Avg ratings (user)	132	33
Avg ratings (item)	216	57
Sparseness Coeff	96%	99%

Table 3.6. Summary of the Data used for validation (1 Fold)

iteration (sampling lag). Hyperparameters α and β were initialized to 2 and 0.5, respectively. Rating predictions are computed as the expected value of $P(R = r|u, i)$.

Figure 3.11 shows the average prediction error achieved by different models on validation folds, varying the number of hidden topics from 2 to 10, while the best result for each model is summarized in Table 3.7. The URP-Gibbs offers a significant advantages in term of prediction accuracy over all the considered competitors, for all number of topics. As reported in [38], the variational pa-

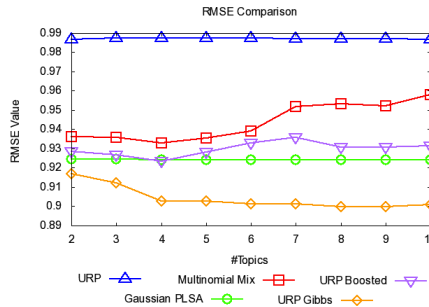


Fig. 3.11. Predictive Accuracy

Approach	Best RMSE	#Topics
<i>Mixture of Multinomials</i>	0.9328	4
<i>Gaussian PLSA</i>	0.9241	10
<i>URP</i>	0.9868	10
<i>URP-Boosted</i>	0.9235	4
URP-Gibbs	0.8997	9

Table 3.7. Predictive Accuracy Summary

parameter estimation of the URP model with randomly initialized parameters

does not perform well, and better results can be obtained by initializing the model with the parameters obtained by fitting a multinomial mixture model with the same number of topics (URP-Boosted). Following this approach, the variational estimation is able to improve the prediction accuracy of the Multinomial Mixture Model, which follows more or less the same trend. The best result achieved by the Gibbs URP is obtained employing 9 topics. The prediction accuracy is not the only advantage of the Gibbs-Sampling inference: as shows in Figure 3.12, the learning time corresponding to variational inference is larger than the one required by the sampling procedure and the gain is more marked for a number of topics greater than 5. It is worth noticing that the

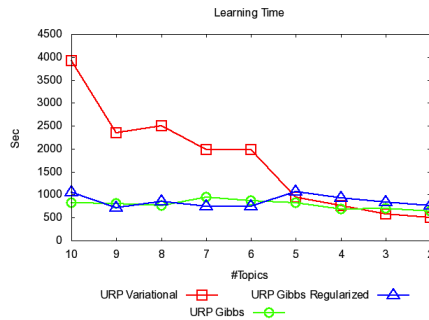


Fig. 3.12. Learning Time

prediction accuracy of URP model is still far from state-of-art collaborative filtering techniques (*Probabilistic Matrix Factorization* [44] achieves on the same data an average RMSE of 0.8655 employing 10 hidden factors). On the other hand, the URP model defines a full generative semantic which is more easy to understand and can be used to generalize to novel user profiles.

The evaluation of the effects of regularization has been performed by considering the capabilities of the model in reproducing neighborhood with maximum dimension of 20, obtained employing Jaccard similarity:

$$sim_{u,v} = \frac{|\mathcal{I}(u) \cap \mathcal{I}(v)|}{|\mathcal{I}(u) \cup \mathcal{I}(v)|}$$

Using such similarities coefficients, the neighborhood model, which stores for each user the first $K = 20$ neighbors, does not take into account rating information but its computation requires less resources. As a preliminary test, regularization has been performed exclusively during the burn-in period with a lag of 5 iterations. Typically, only few (from 5 to 10) regularization steps are performed before the regularization starts to jeopardize prediction accuracy. As shown in Figure 3.13, the influence of the regularization on prediction accuracy is limited. This procedure does not produce significant reduction

on RMSE and also the learning time Figure 3.12 of Regularized URP and Gibbs-URP is similar. The most interesting result is related at the analysis

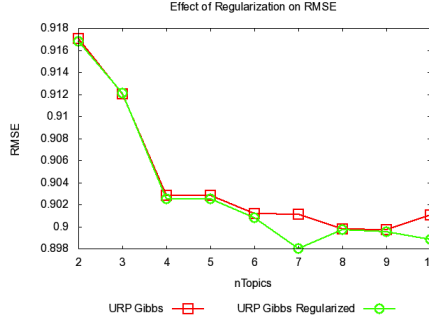


Fig. 3.13. Effect of Regularization on RMSE

of the percentage of neighbors which are mapped to the same cluster of the corresponding user by the regularization. Some results are summarized in Table 3.8, which compares the percentage of neighbors considering regularization with the one achieved without it, varying the size of the neighborhood. The gain achieved with regularization is significant, proving the effectiveness of the proposed approach even if the effective number of iterations that include regularization is limited.

Approach	K=5	K=10	K=15	K=20
<i>URP (5)</i>	24.33	23.82	23.64	23.53
<i>Regularized-URP (5)</i>	36.50	36.33	36.28	36.13
<i>URP (10)</i>	28.19	27.74	27.61	27.41
<i>Regularized-URP (10)</i>	38.54	38.37	38.36	38.34

Table 3.8. % of Neighbors Preserved

Finally, Table 3.9 shows the 5 most significant items for 10 user communities, computed according to Eq. 3.5, their corresponding prior probability (in square brackets) and underlying topics, which are obtained considering the genres associated to each movie. Although we did not directly perform a quantitative comparison between clusters obtained employing regularization and the ones obtained without it, because it would require the definition of a measurement of cluster purity and thus a more deep analysis, a qualitative evaluation shows the regularization procedure helps in enforcing the similarity between relevant items within the communities.

Cluster 1 [0.0472] Crime,Thriller	Cluster 2 [0.0298] Comedy,Romance	Cluster 3[0.1432] Action, Adventure	Cluster 4[0.0540] Action, Sci-fi	Cluster 5[0.4181] Comedy, Drama
Fargo GoodFellas Taxi Driver Psycho Reservoir Dogs	Shakespeare in Love Groundhog Day When Harry Met Sally American President Sleepless in Seattle	Abyss Arachnophobia Batman Returns Logan's Run Heavy Metal	Terminator 2 Star Wars: Return of the Jedi Jurassic Park Matrix Men in Black	Big Chill Risky Business Waking Ned Devine Prizzi's Honor Midnight in the Garden of Good and Evil
Cluster 6 [0.0453] Comedy	Cluster 7 [0.0071] Comedy	Cluster 8 [0.0061] Action, Sci-fi	Cluster 9 [0.2402] Crime, Thriller	Cluster 10 [0.0082] Childrens
Airplane! Clerks Big Lebowski High Fidelity There's Something Ab. Mary	Austin Powers American Pie Caddyshack Animal House Wedding Singer	Terminator Fly StarTrek: Wrath of Khan Heat Star Trek	Silence of the Lambs Godfather Sixth Sense Pulp Fiction Usual Suspects	Toy Story E.T. the Extra-Terrestrial Willy Wonka and the Chocolate Factory Muppet Movie Snow White and the Seven Dwarfs

Table 3.9. User communities and relevant movies

Enhancing Recommendation Accuracy

4.1 Introduction

To improve the accuracy of CF-based recommendation engines, researchers have focused on the development of accurate techniques for rating prediction. The recommendation problem has been interpreted as a missing value prediction problem [13], in which, given an active user, the system is asked to predict her preference for a set of items. Since a user is more prone to access items for which she will likely provide a positive feedback, a recommendation list can be hence built by drawing upon the (predicted) highly-rated items.

Under this perspective, a common approach to evaluate the predictive skills of a recommender systems is to minimize statistical error metrics, such as the *Root Mean Squared Error (RMSE)*. The common assumption is that small improvements in RMSE would reflect into an increase of the accuracy of the recommendation lists [59]. This assumption, however does not necessarily hold. In [50], the authors review the most common approaches to CF-based recommendation, and compare them according to a new testing methodology which focuses on the accuracy of the recommendation lists rather than on the rating prediction accuracy. Notably, cutting-edge approaches characterized by low RMSE values achieves performances comparable to naive techniques, whereas simpler approaches, such as the pure SVD, consistently outperforms the other techniques. In an attempt to find an explanation, the authors impute the contrasting behavior with a “limitation of RMSE testing, which concentrates only on the ratings that the user provided to the system” and consequently “misses much of the reality, where all items should count, not only those actually rated by the user in the past” [50].

The point is that pure SVD rebuilds the original rating matrix in terms of latent factors, rather than trying to minimize the error on observed data. In practice, the underlying optimization problem is quite different, since it takes into account the whole rating matrix considering both observed and unobserved preference values. To summarize, it is likely to better identify the latent factors and the hidden relationships between both factor/users and factors/items. It is natural then to ask whether more sophisticated latent factor models confirm this trend, and are able to guarantee better results in terms of recommendation accuracy, even when they provide poor RMSE performances.

Among the state-of-the art latent factor models, probabilistic techniques offer some advantages over traditional deterministic models: notably, they do not minimize a particular error metric but are designed to maximize the likelihood of the model given the data which is a more general approach; moreover, they can be used to model a distribution over rating values which can be used to determine the confidence of the model in providing a recommendation; finally, they allow the possibility to include prior knowledge into the generative process, thus allowing a more effective modeling of the underlying data distribution. However, previous studies on recommendation accuracy do not take

into consideration such probabilistic approaches to CF, which instead appear rather promising under the above devised perspective.

In this chapter we adopt the testing methodology proposed in [50], and discuss also other metrics [60] for assessing the accuracy of the recommendation list. Based on these settings, we perform in Sec. 4.2 an empirical study of some paradigmatic probabilistic approaches to recommendation and propose in Sec. 4.3 a novel model to estimate both the item selection and relevance, based on the idea “Play and Enjoy”.

4.2 Probabilistic Methods for Top-N Recommendation

In this section we perform an empirical study of some paradigmatic probabilistic approaches to recommendation. We study different techniques to rank items in a probabilistic framework, and evaluate their impact in the generation of a recommendation list. Before analyzing and discussing the application of the probabilistic framework for the generation of personalized recommendation, we will briefly review in Sec. 4.2.1 a general framework for evaluating the recommendation accuracy. A key role in the process of generating accurate recommendation lists is played by the schemes with which to rank items candidate for recommendation, which will be discussed in Sec. 4.2.2. Finally, we will discuss an empirical evaluation of probabilistic approaches in Sec. 4.2.3.

4.2.1 Evaluating Recommendation Accuracy

The general framework for the generation of a recommendation list can be modeled as follows. We will denote by \mathcal{L}_u^j the recommendation list provided by the system to the user u during a generic session j . Then, the following protocols applies:

- Let \mathcal{C}_u^j a list of D candidate random items unrated by the user u in the past sessions $1, \dots, j - 1$;
- Associate to each item $i \in \mathcal{C}_u^j$ a score $p_i^{u,j}$ which represents the user’s interest for i in session j ;
- Sort \mathcal{C}_u^j in descending order given the values $p_i^{u,j}$;
- Add the first N items from \mathcal{C}_u^j to \mathcal{L}_u^j and return the latter to the user.

Simple scoring functions can be obtained considering non-personalized baseline models which take into account the popularity or the average rating of an items. More specifically, *Top Popular (Top-Pop)* recommends items with the highest number of ratings, while *Item Average (Item-Avg)* selects items with the highest average rating. Each RS provides a specific scoring $p_i^{u,j}$, thus

the testing methodology basically relies on the evaluation of capability of the RS in providing higher scores for the items of interest in \mathcal{C}_u^j .

A first, coarse-grained approach to evaluation, can be obtained by employing standard classification-based accuracy metrics such as precision and recall, which require the capability to distinguish between relevant and not relevant recommendations. For example, if an explicit preference value is available, we can consider as relevant all those items which have received a rating greater than the average, otherwise the item will be considered as non-relevant. Given a user, we assume a unique session of recommendation, and we compare the recommendation list of N items provided by the RS, according to the protocol described above, with those relevant items in $\mathcal{I}_S(u)$. In particular, assuming we can identify a subset $\mathcal{T}_u^r \subseteq \mathcal{I}_S(u)$ of relevant items, we can compute precision and recall as specified in Eq. 1.5. However, these definitions aim at evaluating the amount of useful recommendations in a single session. A different perspective can be considered by assuming that a recommendation meets *user satisfaction*, if the user can find at least a *hit*, i.e. an interesting (best rated) item in the recommendation list. Starting from a redefinition of the set of relevant items,

$$\mathcal{T}_u^{rr} = \{i \in \mathcal{I}_S(u) | (u, i, r_i^u) \in \mathbf{S}, r_i^u = V\}$$

the following testing protocol can be applied to assess user satisfaction:

- For each user u and for each item $i \in \mathcal{T}_u^{rr}$:
 - Generate the candidate list \mathcal{C} by randomly drawing from $\mathcal{I}_R(u) - (\mathcal{I}_T(u) \cup \{i\})$.
 - Add i to \mathcal{C} .
 - Associate each item within \mathcal{C} with a suitable score and sort \mathcal{C} in descending order of item scores.
 - Consider the position of the item i in the ordered list: if i belongs to the top- k items, there is a *hit*; otherwise, there is a *miss*.

According to this protocol, [50] defines the *US-Precision* and *US-Recall*.

$$US-Recall(u, k) = \frac{\#hits}{|\mathcal{T}_u^{rr}|}, \quad US-Precision(u, k) = \frac{\#hits}{k \cdot |\mathcal{T}_u^{rr}|}$$

The final values can be obtained by averaging over all users. Notice that the above definition of precision does not penalize false positives: the recommendation is considered successful if it matches at least an item of interest. However, neither the amount of non-relevant “spurious” items, nor the position of the relevant item within the top- k is taken into account.

4.2.2 Probabilistic Modeling of Item Ranking

In this section we discuss how the above described models can be used to provide the ranking p_i^u for a given user u and an item i in the protocol described in Sec. 4.2.1.

Predicted Rating.

The most intuitive way to provide item ranking in the recommendation process relies on the analysis of the distribution over preference values $P(r|u, i)$ (assuming that we are modeling explicit preference data). Given this distribution, there are several methods for computing the ranking for each pair $\langle u, i \rangle$; the most commonly used is the expected value $E[R|u, i]$, as it minimizes the MSE and thus the RMSE:

$$p_i^u = E[R|u, i] \quad (4.1)$$

We will show in Sec. 4.2.3 that this approach fails in providing accurate recommendation and discuss about potential causes.

Item Selection.

For co-occurrence preference approaches, the rank of each item i , with regards to the user u can be computed as the mixture:

$$p_i^u = P(i|u) = \sum_z P(z|u)P(i|z) \quad (4.2)$$

where $P(i|z)$ is the probability that i will be selected by users represented by the abstract pattern z . This distribution is a key feature of co-occurrence preference approaches and models based on free-prediction. When $P(i|z)$ is not directly inferred by the model, we can still estimate it by averaging on all the possible users who selected i :

$$P(i|z) \propto \sum_u \delta(u, i)_{\mathbf{T}} P(z|u)$$

where $\delta_{\mathbf{T}}(u, i) = 1$ if $\mathbf{T}_i^u \neq 0$.

Item Selection And Relevance.

In order to force the selection process to concentrate on relevant items, we can extend the ranking discussed above, by including a component that represents the “predicted” relevance of an item with respect to a given user:

$$\begin{aligned} p_i^u &= P(i, r > \bar{r}_{\mathbf{T}}|u) \\ &= P(i|u)P(r > \bar{r}_{\mathbf{T}}|u, i) = \sum_z P(z|u)P(i|z)P(r > \bar{r}_{\mathbf{T}}|i, z) \end{aligned} \quad (4.3)$$

where $P(r > \bar{r}_{\mathbf{T}}|i, z) = \sum_{r > \bar{r}_{\mathbf{T}}} P(r|i, z)$. In practice, an item is ranked on the basis of the value of its score, by giving high priority to the high-score items.

4.2.3 Evaluation

In this section we experiment the testing protocols presented in Sec. 4.2.1 on the most known probabilistic approaches. We use the MovieLens-1M dataset, which consists of 1,000,209 ratings given by 6,040 users on approximately 3,706 movies, with a sparseness coefficient 96% and an average number of ratings 132 per user, and 216 per item. In the evaluation phase, we adopt a MonteCarlo 5-folds validation, where for each fold contains about the 80% of overall ratings and the remaining data (20%) is used as test-set. The final results reported by averaging the values achieved in each fold.

In order to make our results comparable with the ones reported in [50], we consider *Top-Pop* and *Item-Avg* algorithms as baseline, and *Pure-SVD* as a main competitor. Notice that there are some differences between our evaluation and the one performed in the above cited study, namely: (i) we decided to employ bigger test-sets (20% of the overall data vs 1.4%) and to cross-validate the results; (ii) for lack of space we concentrate on MovieLens only, and omit further evaluations on the Netflix data (which however, in the original paper [50], confirm Pure-SVD as the top-performer); (iii) we decided to omit the “long tail” test, aimed at evaluating the capability of suggesting non-trivial items, as it is out of the scope of this paper.¹

In the following we study the effects of the ranking function on the accuracy of the recommendation list. The results we report are obtained by varying the length of the recommendation list ² in the range 1 – 20 and the dimension of the random sample is fixed to $D = 1000$. In a preliminary test, we found the optimal number of components for the *Pure-SVD* to be set to 50.

4.2.4 Predicted Rating

We start our analysis from the evaluation of the recommendation accuracy achieved by approaches that model explicit preference data, namely PMF[44], MMM[30], URP[38, 61], UCM[62] and G-PLSA[24], where the predicted rating is employed as ranking function. First of all, the following table summarizes the RMSE obtained by these approaches:

Approach	RMSE	#Latent Factors
<i>Item Avg</i>	0.9784	-
<i>MMM</i>	1.0000	20
<i>G-PLSA</i>	0.9238	70
<i>UCM</i>	0.9824	10
<i>URP</i>	0.8989	10
<i>PMF</i>	0.8719	30

¹ Notice, however, that it is still possible to perform an indirect measurement of the non-triviality and correctness of the discussed approaches by measuring the gain in recommendation accuracy wrt. the Top-Pop recommendation algorithm.

² With an abuse of notation, we denote the length of the recommendation list by N in the following plots

The results about Recall and Precision are given in Figure 4.1, where the respective number of latent factors is given in brackets. Considering user satisfaction, almost all the probabilistic approaches fall between the two baselines. Pure-SVD outperforms significantly the best probabilistic performers, namely URP and PMF. The trend for probabilistic approaches does not change considering Recall and Precision, but in this case not even the Pure-SVD is able to outperform Top-Pop, which exhibits a consistent gain over all the considered competitors.

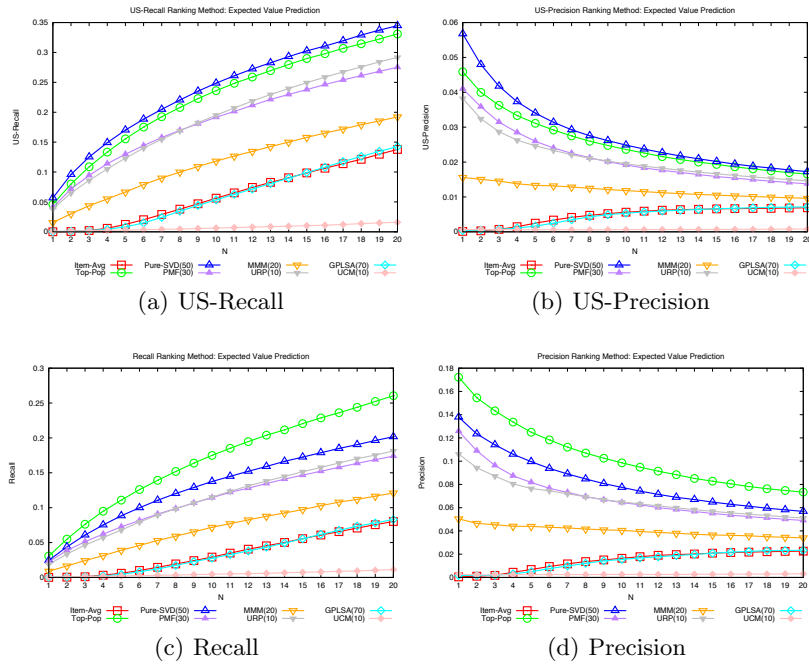


Fig. 4.1. Recommendation Accuracy achieved by probabilistic approaches considering $E[r|u, i]$ as ranking function

A first summary can be obtained as follows. First, we can confirm that there is no monotonic relationship between RMSE and recommendation accuracy. All the approaches tend to have a non-deterministic behavior, and even the best approaches provide unstable results depending on the size N . Further, ranking by the expected value exhibits unacceptable performance on the probabilistic approaches, which reveal totally inadequate in this perspective. More in general, any variant of this approach that we do not report here for space limitations) does not substantially change the results.

4.2.5 Item Selection and Relevance

Things radically change when item occurrence is taken into consideration. Figure 4.2 show the recommendation accuracy achieved by probabilistic models which employ Item-Selection (LDA[32],PLSA[52],UCM and URP) and Item-Selection&Relevance (UCM and URP). The LDA approach significantly outperforms all the available approaches. Surprisingly, UCM is the runner-up, as opposed to the behavior exhibited with the expected value ranking, it is clear that the component $P(i|z)$ here plays a crucial role, that is further strengthened by the relevance ranking component.

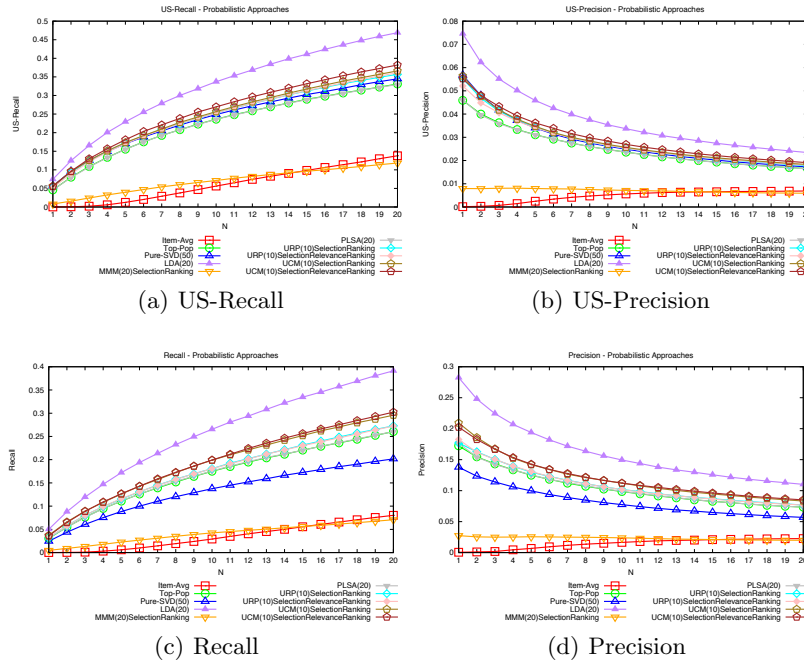


Fig. 4.2. Recommendation Accuracy achieved by probabilistic approaches considering $P(i|u)$ or $P(i, r > 3|u)$ as ranking functions

Also surprising is the behavior of URP, which still achieves a satisfactory performance compared to Pure-SVD. However, it does not compare to LDA. The reason can be found in the fact that the inference procedure in the LDA directly estimates $P(i|z)$, whereas such a component in the URP model is approximated a-posteriori. This is also proved by the unsatisfactory performance of the MMM approach which falls short of the expectations. Since the UCM is an extension of the MMM, it is clear that explicitly inferring the ϕ component in the model helps in achieving a stronger accuracy.

The PLSA model also seems to suffer from the overfitting issues, as it is not able to reach the performances of the Pure-SVD. On the other side, if user satisfaction is not taken into account, the PLSA outperforms the Pure-SVD, as it follows the general trend of the Top-Pop model. More in general, models equipped with Item-Selection&Relevance outperform their respective version which make recommendation basing only on the Item-Selection component.

We also perform an additional test to evaluate the impact of the size of the random sample in the testing methodology employed to measure user satisfaction. Results achieved by LDA, Pure-SVD, UCM/URP (Selection&Relevance Ranking) are given in Figure 4.3. Probabilistic approaches outperform systematically Pure-SVD for each value of D .

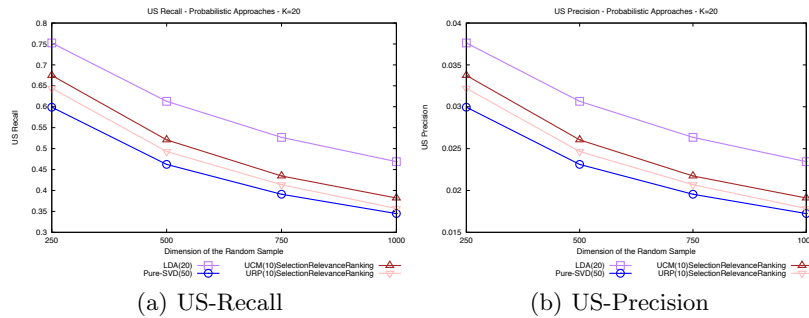


Fig. 4.3. Recommendation Accuracy achieved by probabilistic approaches considering $K=20$ and varying the dimension of the random Sample

4.2.6 Discussion

There are two main considerations in the above figures. One is that rating prediction fails in providing accurate recommendations. The second observation is the unexpected strong impact of the item selection component, when properly estimated.

In an attempt to carefully analyze the rating prediction pitfalls, we can plot in Figure 4.4(a) the contribution to the RMSE in each single evaluation in \mathcal{V} by the probabilistic techniques under consideration. Item-Avg acts as baseline here. While predictions are accurate for values 3 – 4, they result rather inadequate for border values, namely 1, 2 and 5. This is mainly due to the nature of RMSE, which penalizes larger errors. This clearly supports the thesis that low RMSE does not necessarily induces good accuracy, as the latter is mainly influenced by the items in class 5 (where the approaches are more prone to fail). It is clear that a better tuning of the ranking function should take this component into account.

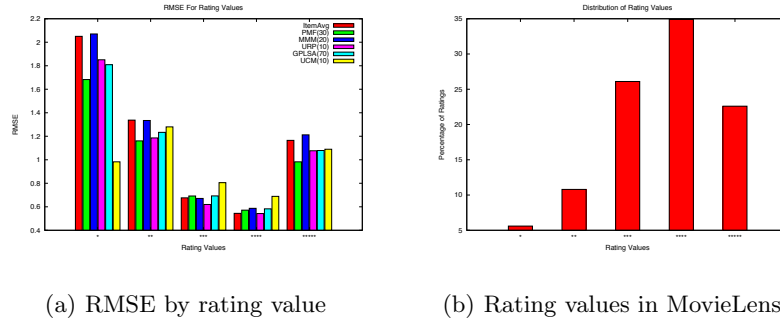
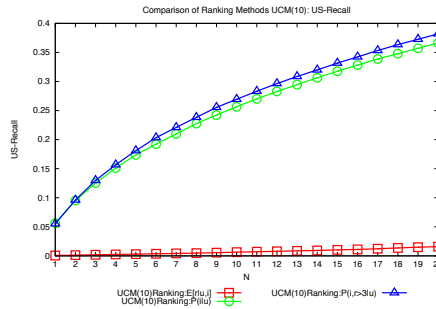


Fig. 4.4. Analysis of Prediction Accuracy

Also, by looking at the distribution of the rating values, we can see that the dataset is biased towards the mean values, and more in general the low rating values represent a lower percentage. This explains, on one side, the tendency of the expected value to flatten towards a mean value (and hence to fail in providing an accurate prediction). On the other side, the lack of low-rating values provides an interpretation of the dataset as a *Like/DisLike* matrix, for which the item selection tuning provides a better modeling.

By the way, the rating information, combined with item selection, provides a marginal improvement, as testified by Figure 4.2.6. Here, a closer look at the UCM approach is taken, by plotting three curves relative to the three different approaches to item ranking. Large recommendation lists tend to be affected by the rating prediction. Our experiments have shown that item selection component plays the most important role in recommendation ranking. However, better results can be achieved by considering also a rating prediction component, as we will show in the next section.

Fig. 4.5. Comparison of Different Ranking Function on UCM(10)



4.3 Modeling Item Selection and Relevance

A key role in the process of generating accurate recommendation lists is played by the schemes with which to rank items candidate for recommendation. In the previous section, we provided a comparative analysis of three possible such schemes, and studies their impact in the accuracy of the recommendation list. The results of such study can summarized as follows.

- Lower RMSE values do not necessarily imply improvements in recommendation accuracy. Cutting-edge probabilistic approaches, such as PMF [44], equipped with *expected-value* ($p_i^u = E[R|u, i]$) item-ranking schemes have been shown to perform poorly in terms of recommendation accuracy.
- Probabilistic CF methods were shown to outperform state-of-the-art competitors in terms of recommendation accuracy when equipped with the item selection scheme $p_i^u = P(i|u) = \sum_z P(z|u)P(i|z)$.

In the light of foregoing considerations, in this section we propose a new probabilistic approach to recommendation for explicit preference data, referred to as the *Bayesian User-Community Model (BUCM)*, which is based on a mix of *item selection and relevance ranking*. BUCM introduces a generative process, which takes into account both item selection and rating emission to gather into communities those users who experience the same items and tend to adopt the same rating pattern. Each user is modeled as a random mixture of topics, where each topic is characterized by a distribution modeling the popularity of items within the respective user-community and by a distribution over preference values for those items. The proposed model can be associated with a novel *item-relevance* ranking criterion, which is based both on item popularity and user's preferences. A key difference with respect to conventional probabilistic approaches to recommendation is that Bayesian UCM allows *free-prediction* and is, thus, more suitable for the estimation of selection and preference parameters. While most of the conventional probabilistic techniques focus on *forced-prediction*, which explicitly requires to predict the preference value for each observed user-item pair, the goal of Bayesian UCM is to model item selection and rating prediction simultaneously.

4.3.1 The Bayesian User Community Model

Bayesian UCM relies on a generative process, which takes into account both item selection and rating emission. Each user is modeled as a random mixture of topics, where the individual topic is then characterized both by a distribution modeling item-popularity within the considered user-community and by a distribution over preference values for those items.

The main difference between the proposed Bayesian UCM model and the state-of-art probabilistic approaches to CF is the former is a free-prediction model. In fact, while most of the models accord with a missing-value perspective and, hence, are focused on the prediction of a preference value r_i^u

given the pair $\langle u, i \rangle$, the Bayesian UCM model tries to also infer the tendency of a user to experience some items over others independent of her/his rating values. The Bayesian UCM model assumes that this tendency is influenced by implicit and hidden factors which characterize each user community. To elucidate, a user may be pushed to experience a certain item because she/he belongs to a community in which the category of that item occurs with an high probability, although this has no impact on the rating assigned to the aforesaid item category. The probability of observing an item is independent from the rating assigned, given the state of the latent variables. Moreover, free-prediction models are focused on both the estimation of a rating behavior and the popularity of an item within each user community. An item which has received high ratings and has been experienced few times by the users belonging to the considered community could not have better chances of being recommended with respect to a popular item within the same community, which has received only ratings around the average.

The generative process behind the Bayesian UCM can be summarized as follows:

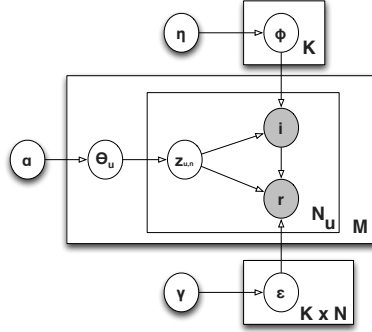
1. For each user $u \in \mathcal{U}$ sample user community-mixture components $\boldsymbol{\vartheta}_u \sim Dir(\boldsymbol{\alpha})$
2. For each topic (or equivalently user community) $z = \{1, \dots, K\}$,
 - a) Sample item selection components $\boldsymbol{\varphi}_z \sim Dir(\boldsymbol{\beta})$
 - b) Sample rating probabilities $\boldsymbol{\varepsilon}_z \sim Dir(\boldsymbol{\gamma})$
3. Sample the number of items for the user u , $N_u \propto Poisson(\mathcal{K})$
4. For $n = 1$ to N_u
 - a) Choose a user attitude $z_{u,n} \sim Discrete(\boldsymbol{\vartheta}_u)$
 - b) Choose an item $i_n \sim Multi(\boldsymbol{\varphi}|z_{u,n})$
 - c) Generate a rating value for the chosen item according to the distribution $P(r|\boldsymbol{\varepsilon}_{z_{u,n}, i_n})$

The corresponding graphical model is illustrated in Figure 4.6

4.3.2 Parameter Estimation

We here introduce inference and parameter estimation within the devised Bayesian UCM. The notation used in our discussion is summarized in Table 4.3.2. Given the hyperparameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, the joint distribution of the data \mathbf{R} , the user-topic mixtures Θ , the item-selection components Φ , the rating probabilities Γ and the observation-topic assignments \underline{Z} , can be computed as:

$$P(\mathbf{R}, \underline{Z}, \Theta, \Phi, \Gamma | \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = P(\mathbf{R} | \underline{Z}, \Phi, \Gamma) P(\underline{Z} | \Theta) \cdot P(\Theta | \boldsymbol{\alpha}) P(\Phi | \boldsymbol{\beta}) P(\Gamma | \boldsymbol{\gamma})$$

Fig. 4.6. Bayesian User Community Model


The complete data likelihood can be obtained by integrating over Θ , Φ and Γ :

$$P(\mathbf{R}, \underline{Z} | \alpha, \beta, \gamma) = \int \int \int P(\mathbf{R} | \underline{Z}, \Phi, \Gamma) P(\underline{Z} | \Theta) P(\Theta | \alpha) P(\Phi | \beta) P(\Gamma | \gamma) d\Theta d\Phi d\Gamma$$

which, due to the conditional independence $\mathbf{R} \perp\!\!\!\perp \alpha | \underline{Z}$, can be factored as:

$$P(\mathbf{R}, \underline{Z} | \alpha, \beta, \gamma) = \int P(\underline{Z} | \Theta) P(\Theta | \alpha) d\Theta \int \int P(\mathbf{R} | \underline{Z}, \Phi, \Gamma) P(\Phi | \beta) P(\Gamma | \gamma) d\Phi d\Gamma$$

By rearranging the components and grouping the conjugate distributions, the complete data likelihood can be expressed as:

$$P(\mathbf{R}, \underline{Z} | \alpha, \beta, \gamma) = \prod_{u=1}^M \frac{1}{\Delta(\alpha)} \int \prod_{k=1}^K \vartheta_{u,k}^{n_u^k + \alpha_k - 1} d\vartheta_u \prod_{i=1}^N \frac{1}{\Delta(\beta)\Delta(\gamma)} \int \prod_{k=1}^K \varphi_{i,k}^{n_i^k + \beta_i - 1} d\varphi_k \\ \cdot \int \prod_{k=1}^K \prod_{r=1}^V \epsilon_{k,i,r}^{n_{i,r}^k + \gamma_r - 1} d\epsilon_{k,i}$$

The latter is the starting point for the inference of all the topics underlying the generative process, as the conditioned distribution on \underline{Z} can be written as:

$$P(\underline{Z} | \mathbf{R}, \alpha, \beta, \gamma) = \frac{P(\underline{Z}, \mathbf{R} | \alpha, \beta, \gamma)}{P(\mathbf{R} | \alpha, \beta, \gamma)}$$

This formula is however intractable, mainly because the computation of the denominator requires a summation over an exponential number of terms. Gibbs Sampling addresses this problem by defining a Markov chain, in which at each step inference can be accomplished by exploiting the full conditional $P(Z_n | \underline{Z}_{-n}, \mathbf{R}, \alpha, \beta, \gamma)$. In the latter, n denotes a single rating observation

$n = \{u, i, r_i^u\}$, Z_n is the cell of the matrix \underline{Z} which corresponds to this observation, and \underline{Z}_{-n} denotes the remaining topic assignments. The chain is hence defined by iterating over all the available n states. The Gibbs Sampling algorithm estimates the probability of assigning the topic k to the n -th observation, given the assignment corresponding to all the other rating observations:

$$p(Z_n = k | \underline{Z}_{-n}, \mathbf{R}) \propto \frac{n_u^k + \alpha_k - 1}{\sum_{k'=1}^K (n_u^{k'} + \alpha_{k'}) - 1} \cdot \frac{n_i^k + \beta_i - 1}{\sum_{i'=1}^N (n_{i'}^k + \beta_{i'}) - 1} \cdot \frac{n_{i,r}^k + \gamma_r - 1}{\sum_{r'=1}^V (n_{r',i}^k + \gamma_{r'}) - 1} \quad (4.4)$$

Given the state of the markov chain, denoted my $\mathcal{M} = \mathbf{R}, \underline{Z}$, where \underline{Z} encodes the topic assignment for each pair $\langle u, i \rangle \in \mathbf{R}$, we can obtain the multinomial parameters Φ and Θ and Γ noticing that, by algebraic manipulations, they reduce to Dirichlet distributions and can hence been estimated as the underlying expectations [63]:

$$\vartheta_{u,k} = \frac{n_u^k + \alpha_k}{N_u + \sum_{k=1}^K \alpha_k} \quad (4.5)$$

$$\varphi_{i,k} = \frac{n_i^k + \beta_i}{\sum_{i=1}^N n_i^k + \beta_i} \quad (4.6)$$

$$\epsilon_{k,i,r} = \frac{n_{i,r}^k + \gamma_r}{\sum_{r'=1}^V n_{i,r'}^k + \gamma_{r'}} \quad (4.7)$$

Algorithm 3 The Gibbs-sampling procedure for parameter estimation within Bayesian UCM

Require: The sets $\mathcal{U} = \{u_1, \dots, u_M\}$ and $\mathcal{I} = \{i_1, \dots, i_N\}$
the rating matrix \mathbf{R} , the number of latent topics K , initial hyperparameters α, β and γ .

```

1: initializeTopicAssignments() {Randomly assign topics}
2: iteration  $\leftarrow$  0
3: converged  $\leftarrow$  false
4: while iteration < nMaxIterations and  $\neg$ converged do
5:   for all  $\langle u, i, r \rangle \in \mathbf{R}$  do
6:      $z'_{u,i} \leftarrow$  sampleTopic(u, i, r) {According to Eq. 5.10};
7:     update counts using the new topic for the observation  $\langle u, i, r \rangle$ 
8:   end for
9:   updateHyperParams()
10:  if (iteration > burnin) and (iteration % sampleLag = 0) then
11:    sampleUserTopicsMixingProbabilities() {According to Eq. 5.17 };
12:    sampleItemSelectionProbabilities() {According to Eq. 5.18 };
13:    sampleRatingProbabilities() {According to Eq. 5.19 };
14:    converged  $\leftarrow$  checkConvergence()
15:  end if
16:  iteration  $\leftarrow$  iteration + 1
17: end while

```

SYMBOL	DESCRIPTION
M	#Users
N	# Items
\mathbf{R}	$M \times N$ Rating Matrix
K	# topics/user communities
α	K - vector, Dirichlet priors on user communities
β	N -vector, Dirichlet priors on items
γ	V -vector, Dirichlet priors on rating values
Θ	matrix of parameters ϑ_u
ϑ_u	mixing proportion of communities for the user u
z	topic variable
Φ	matrix of parameters φ_k
φ_k	mixing proportion of items for the community k
N_u	# preference observations for user u
Γ	matrix of parameters ε_k
ε_k	vector of rating distributions $\varepsilon_{k,i}$ for topic k
$\varepsilon_{k,i}$	distribution over rating values for the item i and the community k
n_i^k	# times that the item i has been assigned to topic k
$n_{i,r}^k$	# times that the rating r has been assigned to the item i when the topic is k
n_u^k	# times an item evaluated by u has been assigned to topic k
z_n	topic assignment for the observation $n = \langle u, i \rangle$
z_{-n}	topic assignment for all other observations except the current observation $n = \langle u, i \rangle$

Table 4.1. Summary of notation

Algorithm 6 shows the pseudocode for the inference phase: the Gibbs Sampling procedure starts with a random initialization; then topic assignments are estimated till convergence or till the number of performed iteration reaches the maximum value. Hyperparameters can be updated employing iterative approximation (see [58] for further details). The convergence criteria checks whether the increase in likelihood (measured on held-out data) is above a predefined threshold.

4.3.3 Experimental Evaluation

In this section we comparatively evaluate the recommendation performance of Bayesian UCM. The experiments are aimed at assessing the quality of the proposed model in two different perspective:

- From the *forced-prediction* viewpoint, we show that the *predictive accuracy* (i.e., the prediction error) exposed by the Bayesian UCM over unobserved ratings is comparable to other state-of-the-art probabilistic approaches.
- Conversely, from the *free-prediction* viewpoint, we show that Bayesian UCM is the top-notch approach in term of *recommendation accuracy*, i.e., the accuracy of the recommendation list generated.

According to the empirical results originally found in [50] and subsequently confirmed in [64], there is no monotonic relationship between prediction error (or, equivalently, accuracy) and recommendation accuracy. Therefore, a low prediction error does not necessarily imply a satisfactory recommendation performance. The latter is better evaluated in terms of the accuracy of the recommendation lists provided to the users. Therefore, the findings in this section will state the superiority of the Bayesian UCM in providing accurate recommendations.

We perform the above evaluations on two reference benchmark data sets, namely MovieLens-1M and a sample of Netflix data. The main features of these datasets are summarized in the table below:

	Netflix		MovieLens	
	Training Set	Test Set	Training Set	Test Set
Users	435,656	389,305	6,040	6,040
Items	2,961	2,961	3,706	3,308
Ratings	5,714,427	3,773,781	800,168	200,041
Avg ratings (user)	13.12	9.69	132.47	33,119
Avg ratings (item)	1929.90	1274.50	215.91	60.47
Sparseness Coeff	0.9956		0.9643	

As far as Predictive Accuracy is concerned, Bayesian UCM provides the following estimation for user preference:

$$P(R = r|u, i) = \sum_z P(z|\vartheta_u)P(r|z, i) = \sum_k \vartheta_{u,k} \epsilon_{k,i,r}$$

We evaluate the *RMSE* of Bayesian UCM over the MovieLens data set and compare its predictive accuracy against a selection of state-of-art probabilistic competitors, namely, Mixture of Multinomials [30], G-PLSA [24], URP-Boosted [38], URP-Gibbs [61], UCM [62] and PMF [44]. Results are summarized in Table 4.2, wherein column **#Topics** indicates the number of latent factors taken into account within each individual probabilistic model. The minimum *RMSE* value is highlighted in bold.

Empirical evidence reveals that the predictive accuracy of Bayesian UCM is lower than that of G-PLSA, URP-Boosted, URP-Gibbs and PMF. This is not a surprising finding, since the generative process of the aforesaid competitors is focused on the prediction accuracy. By looking at the results in Table 4.2, Bayesian UCM is superior to both the variants of UCM. In particular, Bayesian UCM outperforms significantly the UCM variant equipped with multinomial rating distribution.

Approach	Best RMSE	#Topics
<i>Mixture of Multinomials</i>	0.9328	4
<i>G-PLSA</i>	0.9241	10
<i>URP-Boosted (Variational)</i>	0.9235	4
<i>URP-Gibbs</i>	0.8997	9
<i>PMF</i>	0.8655	10
<i>UCM (Multinomial)</i>	0.9638	4
<i>UCM (Gaussian)</i>	0.9359	2
<i>Bayesian UCM</i>	0.9263	30

Table 4.2. Summary of predictive competitor accuracy over the Movielens dataset

It is worth providing an insight into the difference in predictive accuracy between URP-Gibbs and Bayesian UCM, since both are Bayesian probabilistic approaches based on a Gibbs sampling procedure for approximated model inference. The observed *RMSE* discrepancy is essentially due to the nature of the underlying mathematical models. Indeed, URP-Gibbs is a forced-prediction approach meant to increase the likelihood of those communities, in which a similar rating behavior is observed across the respective users. This is clearly preferable for predictive accuracy. Instead, Bayesian UCM is a free-prediction approach, that considers not only the rating behavior but also the frequency of item selection in the identification of user communities. As a matter of fact, the generic community gathers those users who tend to assign similar ratings to items that are frequently experienced within the same community. Therefore, as it has been already anticipated, an item which has received high ratings from few users of a community could not have better chances of being recommended with respect to a popular item within the same community, which has received only ratings around the average. In other words, combining rating behavior (which is the only component in forced-prediction) with item selection for free prediction tends to have a negative impact on the resulting predictive accuracy.

As already discussed, the results are significantly different when recommendation accuracy is taken into account. Here, Bayesian UCM is compared against a selection of heterogeneous competitors, namely Top-Pop and Item-Avg [50], Pure-SVD [50], LDA [32], PLSA [28], URP-Gibbs and UCM. Item ranking in the context of the aforesaid probabilistic approaches, apart from UCM, exclusively relies on item selection. The UCM is the only competitor that can combine both item selection and rating emission for item ranking, as it directly supports a free-prediction approach. Also, It is worth noticing that, though being the top-performer in terms of predictive accuracy, PMF is not considered here, because previous tests performed in [64] have shown that its recommendation accuracy is low. The results are summarized in Fig. 4.7 in which Bayesian UCM is referred to as BUCM for convenience. The latter achieves the best performance against all the competitors, and in general the

two datasets confirm the same trend.³ Notice that, the performances of both Pure-SVD and PLSA over the Netflix data set are very close to Top-Pop.

These plots show the results achieved by the selected competitors over the MovieLens data set, when the size of their recommendation lists varies from 1 to 20. It is evident that all probabilistic approaches, with the only exception of PLSA, outperform both the baseline methods, namely Top-Pop and Item-Avg, as well as Pure-SVD. This confirms the effectiveness of probabilistic modeling: Bayesian UCM outperforms all competitors both in (standard and US) precision and recall.

The gain in accuracy with respect to LDA is more significant when US-precision and US-recall are taken into account (recall 0.5 vs 0.468 when $k = 20$), mainly because in this test item ranking benefits from the component of predicted relevance.

Notably, the discrepancy between the recommendation accuracy of Bayesian UCM and UCM is consistently large. This confirms the advantages of the Bayesian approach. Also, it is worth noticing how URP, though exhibiting a higher predictive accuracy than Bayesian UCM, poorly performs in terms of recommendation accuracy with respect to the latter. Such an empirical evidence confirms the importance of the selection component in the recommendation process.

The US precision and recall of Bayesian UCM are further (comparatively) investigated over the Movielens data set, when the size of the random sample of candidate recommendations is varied in the testing protocol from 250 to 1000. The results shown in Figure 4.8 prove the superiority of Bayesian UCM. Finally, it is important to evaluate whether the Bayesian UCM introduced a significant performance degradation. In principle, the increase in recommendation accuracy comes at a cost of a more complex model which in turn provides a more complex inference procedure. In fig. 4.9 we compare the execution times of the Bayesian UCM to those of the URP model. The two models exhibit a similar generative process, the difference lying in the explicit modeling (and inference) of the item selection component. The latter difference however, only yields a reasonable overhead.

³ We omitted some models on NetFlix data to ease readability of the overlapping curves.

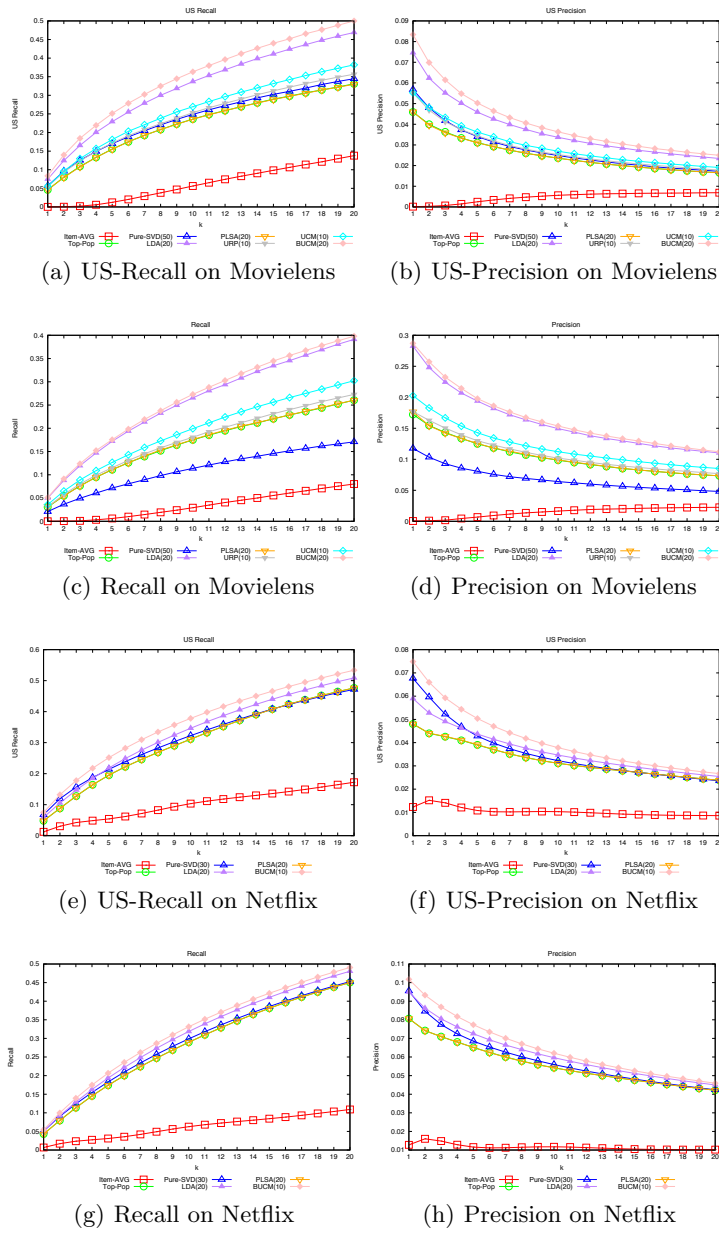


Fig. 4.7. Precision and recall over the MovieLens and Netflix data sets

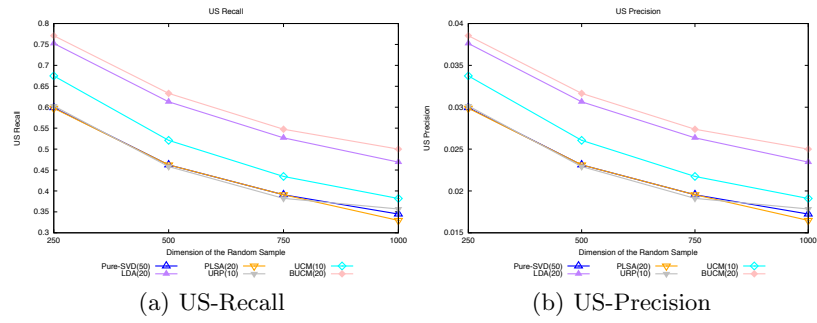


Fig. 4.8. Recommendation accuracy over the MovieLens data set with the increasing size of the random sample and $K = 20$

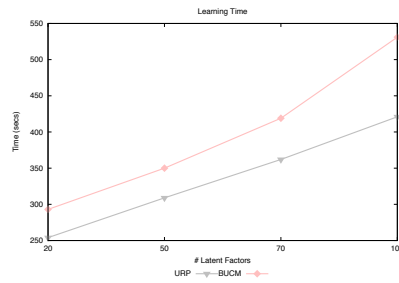


Fig. 4.9. Execution time: BUCM vs. URP on MovieLens.

Hierarchical Co-clustering of Users' Preference Data

5.1 Introduction

Collaborative filtering data exhibit global patterns (i.e. tendencies of some products of being ‘universally’ appreciated) as well significant local patterns (i.e tendency of users belonging to a specific community to express similar preference indicators on the same items). Local preferences affect the performance of RS especially when the number of users and items grows, and their importance has been acknowledged by the current CF literature [65]. This schema, in which similarity coefficients between users are computed by taking into account the item-category, allows two users to agree perfectly on one topic while disagree completely on one other.

Typically, local patterns can be better detected by means of co-clustering approaches [55, 35, 37, 66, 67, 68]. Unlike traditional CF techniques, which try to discover similarities between users or items using clustering techniques or matrix decomposition methods, co-clustering approaches aim to partition data into homogenous blocks enforcing a simultaneous clustering on both the dimensions of the preference data. This highlights the mutual relationships between users and items: similar users are detected by taking into account their ratings on similar items, which in turn are identified considering the ratings assigned by similar users.

However, a main weakness of the current approaches to co-clustering is the static structure enforced by fixed row/column blocks where both users and items have to fit. For example, the movies “Titanic” and “Avatar”, are typically associated with different categories: the former is about romance, whereas the latter can be considered an action, sci-fi movie. Assuming a global and unique partition on the item-set, we can expect to see the movies into different partitions. However, that structure would fail to recognize a group of users who are really into the movies of James Cameron (who is the director for both the movies). Analogously, any method associating the two movies with the same partition would fail in identifying the difference in genre.

The issue in the previous example is that different user groups can infer different interpretations of item categories. A more flexible structure, where item categories are conditioned by user categories, would better model such situation, by e.g., allowing “Titanic” and “Avatar” to be observed in the same item category within the “Cameron” group, and in different categories outside. Notice that traditional clustering approaches are not affected by this problem, as they only concentrate on local patterns in one dimension of the rating matrix. The drawback, however, is that they ignore structural information in the other dimension, which by the converse can be exploited both for more accurate prediction and user profiling.

In this chapter we present and discuss probabilistic hierarchical approaches which are able to discover both global and local trends in data, allowing different user communities to show different preference values on distinct groups of items. The underlying structure differs from the previously proposed co-clustering approaches to CF data because it does not assume the existence of

a unique partition on the item-set: each user community is characterized by having its own set of topics involving items and user preferences.

5.2 A 2-phase Probabilistic Hierarchical Co-clustering Approach

The starting point in our approach is the observation that different communities can infer different evaluations of the same item. Specific groups of users tend to be co-related according to different subsets of features. However, though semantically-related, two users with (possibly several) differences in their item ratings would hardly be recognized as actually similar by any global model imposing a fixed structure for item categories. Individual user can be intended as a mixture of latent concepts, each of which being a suitable collection of characterizing features. Accordingly, two users are considered as actually similar if both represent at least a same concept. Viewed in this perspective, the identification of *local patterns*, i.e. of proper combinations of users and items, would lead to the discovery of natural clusters in the data, without incurring into the aforesaid difficulties. Consider the toy example in Figure 5.1, where homogenous blocks exhibiting similar rating patterns are highlighted. There are 7 users clustered into two main communities. Community 1 is characterized by 3 main topics (with groups $d_{11} = \{i_1, i_2, i_3\}$, $d_{12} = \{i_4, i_5, i_6, i_7\}$ and $d_{13} = \{i_8, i_9, i_{10}\}$), whereas community 2 includes 4 main topics (with groups $d_{21} = \{i_1, i_4, i_5\}$, $d_{22} = \{i_2, i_3, i_7\}$, $d_{23} = \{i_6, i_{10}\}$ and $d_{24} = \{i_8, i_9\}$). The novelty is that different communities group the same items differently. This introduces a topic hierarchy which in principle increases the semantic power of the overall model.

		i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	
Community 1	u_1	1		1	5		4	5		2	2	$d_1 = \{i_1, i_2, i_3\}$
	u_2	1	1			5	4	4	5	2	2	$d_2 = \{i_4, i_5, i_6, i_7\}$
	u_3	1	1	1	4	5			5	2		$d_3 = \{i_8, i_{10}\}$
	u_4		1	1		5	4	5	4		2	
Community 2	u_5	5		4	5	5	1	4	3		1	$d_1 = \{i_1, i_4, i_5\}$
	u_6		4	4	5	5	1	4	3	3	1	$d_2 = \{i_2, i_3, i_7\}$
	u_7	5	4		5		1	4	3	3		$d_3 = \{i_6, i_{10}\}$ $d_4 = \{i_8, i_9\}$

Fig. 5.1. Example of Local Pattern in CF Data

The generative model for the proposed scheme can be summarized as follows:

1. Select a user community c_k according to the probability distribution π_k ;
2. select a user u with probability $P_k(u) = P(u|c_k)$ and an item i with probability $P_k(i) = P(i|c_k)$;
3. Choose a topic d_h with probability $P(d_h|i, c_k)P(d_h|u, c_k)$;
4. produce the rating r with probability $\phi_h(r) = P(r|d_h)$.

Formally, we can assume that the probability of a triplet $\langle u, i, r \rangle$ is

$$P(u, i, r) = \sum_{k=1}^K \pi_k P_k(u) P_k(i) P(r|i, u, c_k) \quad (5.1)$$

where the latter correspond to a “local” probabilistic latent semantic analysis, provided that the user communities are known:

$$P(r|i, u, c_k) = \sum_{h=1}^{H_k} \phi_h(r) P_k(d_h|i) P_k(d_h|u) \quad (5.2)$$

The idea, in the above formula, is learning latent communities from the data as well as a collection of characterizing concepts for each community. In particular, each rating can be seen as the outcome of a mixture of various concepts, where some concepts are more or less probable according to the cluster where the user fits. Hence, a data tuple can be thought as the outcome of the following generative model: firstly pick a distribution over latent clusters; next, choose the concepts associated and finally generate the individual values. Also, notice the role of the prior probabilities (π_k) in the generative process. In practice, they model the assumption that observing a pair $\langle u, i \rangle$ is not totally random, but it is instead the result of the grouping of users into communities.

Due to the strong coupling between the user community latent variable c and the one corresponding to local patterns d , the exact inference for the model characterized by the joint probability in Eq. 5.1, which would maximize both the user community cohesion and the local topic similarity, is difficult to solve analytically. Hence, we adopt an approximated solution, based on a hard clustering policy for user communities, such that the inference of the parameters can be performed efficiently without compromising the generative semantic and the flexibility of the model.

We devise a hierarchical approach to the estimation of the components involved into Eq. 5.1. In practice, our approach consists in a preliminary discovery structure, where user communities are detected. Next, for each user community, a topic model is investigated, and the most prominent topics are discovered and properly modeled.

The general scheme of the algorithm is shown in Algorithm 4 and could be summarized as follows: given a rating matrix \mathbf{R} , discover k user communities; then, for each of those communities, according to an hard clustering approach, select from \mathcal{U} a subset of users that belong to the considered community and generate a set of H_k topic models for their ratings.

Algorithm 4 HMbuild

Require: The sets $\mathcal{U} = \{u_1, \dots, u_M\}$ and $\mathcal{I} = \{i_1, \dots, i_N\}$
and the corresponding rating matrix \mathbf{R} ;
Ensure: a set $\mathcal{C} = \{c_1, \dots, c_K\}$ of user community models
and a subset $\mathcal{D}^k = \{d_1^{(k)}, \dots, d_{H_k}^{(k)}\}$ for each user community k

- 1: $\mathcal{C} \leftarrow \text{GenerateUserCommunities}(\mathbf{R})$;
- 2: **for all** community model $c_k, k = 1, \dots, K$ **do**
- 3: let $\mathcal{U}_k = \{u \in \mathcal{U} | p(c_k|u) \geq p(c_j|u), j = 1, \dots, K\}$, and \mathbf{R}_k the corresponding submatrix of \mathbf{R} ;
- 4: $\mathcal{D}^k \leftarrow \text{GenerateTopicModels}(\mathbf{R}_k)$;
- 5: **end for**

The hierarchical model for users' ratings consists in a set of K user community models and for each of them a set of H_k topic models which represent local preference patterns for the member of the considered community. The user community level specifies the probabilities $\gamma_{uk} = P(c_k|u)$ with $k = 1, \dots, K$, which measure how much the ratings given by the user u fit the preference behavior underlined by each of the communities. For example, the following user profile ("The Good, the Bad and the Ugly", 5), ("Once Upon a Time in the West", 5) would presumably fits better the patterns of the "western" user community rather than the ones typical of the "romance" community.

The probability of observing the rating r for the pair (u, i) can be computed considering two schema, summarized in Algorithm 5:

- *Hard-Clustering Prediction:*

$$P(r|i, u) = \sum_{h=1}^{H_k} \phi_h(r) P_k(d_h|i) P_k(d_h|u) \quad (5.3)$$

where $k = \underset{j=1, \dots, K}{\operatorname{argmax}} \gamma_{uj}$ is the cluster that better represents the previously observed rating of the user u . This prediction rule relies exclusively on the information given by the topic model corresponding to the user's cluster; thus it might produce low quality predictions if the user's community is not identified with enough confidence.

- *Soft-Clustering Prediction:*

$$P(r|i, u) = \sum_k \gamma_{uk} \cdot \tilde{P}(r|i, u, c_k) \quad (5.4)$$

where the probabilities γ_{uk} act as mixture weights and the distribution over rating values corresponding to the community c_k is computed taking into account both global and local patterns:

$$\tilde{P}(r|i, u, c_k) = \begin{cases} P(r|i, u, c_k) & \text{if } u \in \mathcal{U}_k \\ P(r|i, c_k) & \text{otherwise} \end{cases} \quad (5.5)$$

Note that if $u \in \mathcal{U}_k$ then γ_{uk} is the dominant mixing weight and the distribution over ratings is refined by considering the corresponding set of topic models; in the opposite case the distribution over ratings can be estimated by considering the probability of observing each rating given an item within the considered community.

Algorithm 5 HMcomputeRatingsProbability

Require: a pair $\langle u, i \rangle$

Ensure: a probability $P(R = r|u, i)$ for each rating value r

```

1: let  $c = \operatorname{argmax}_{j=1, \dots, K} p(c_k|u)$ 
2: for all  $r = 1$  to  $V$  do
3:   if Hard-Clustering then
4:      $P(R = r|u, i) = \sum_{h=1}^{H^k} \phi_h(r) P_k(d_h|i) P_k(d_h|u)$ 
5:   else
6:     for all community model  $c_k, k = 1, \dots, K$  do
7:       if  $k = c$  then
8:          $prob \leftarrow \mathcal{D}^k.getRatingProbability(r, u, i)$ 
9:       else
10:         $prob \leftarrow c_k.getRatingProbability(r, i)$ 
11:       end if
12:        $P(R = r|u, i) \leftarrow P(R = r|u, i) + \gamma_{uk} \times prob$ 
13:     end for
14:   end if
15: end for

```

5.2.1 Modeling User Communities.

The discovery of the communities is accomplished essentially via a model-fitting procedure based on a maximum-likelihood estimation. In practice, we assume that the rating matrix \mathbf{R} is modelled as a set of user vectors, where each vector is characterized by the preferences of the user. Formally, this means that we can model the probability $P(r, i|u)$ for each triplet $\langle u, i, r \rangle$.

The corresponding probability of observing a user hence corresponds to the joint probability of observing all his ratings, that is

$$P(u|\Theta, \mathbf{R}) = \prod_{i=1}^N \prod_{r=1}^V (P(i|\Theta) \cdot P(r|i, \Theta))^{\delta(u, i, r)}$$

where

$$\delta(u, i, r) = \begin{cases} 1 & \text{if } r_i^u = r \\ 0 & \text{otherwise} \end{cases}$$

This modeling allows us to adopt a maximum likelihood approach to the estimation of the Θ parameters characterizing the $P(i|\Theta)$ and $P(r|i, \Theta)$. For example, we can characterize $P(i|\Theta)$ via a Bernoulli pdf parameterized by α_i , and $P(r|i, \Theta)$ as a multinomial (with factors σ_{ri}). Within a ML framework, the estimation of the above probabilities would produce

$$\alpha_i = \frac{\sum_{u=1}^M \sum_{r=1}^V \delta(u, i, r)}{\sum_{u=1}^M \sum_{i'=1}^N \sum_{r=1}^V \delta(u, i', r)} \quad \sigma_{ri} = \frac{\sum_{u=1}^M \delta(u, i, r)}{\sum_{u=1}^M \sum_{r'=1}^V \delta(u, i, r')} \quad (5.6)$$

A first effect of the above estimates is to adjust the soft-clustering prediction formula Eq. 5.5 as

$$P(r|i, u) = \beta \sum_k \gamma_{uk} \cdot P(r|u, i, c_k) + (1 - \beta) \sigma_{ri}$$

where β is a weighting factor proportional to the number of ratings $|\mathcal{I}(u)|$. In practice, the estimate σ_{ri} provides a higher contribution when the number of ratings given by a user is low (and hence it acts as a prior).

The component $P(r, u, i, c_k)$ and the posteriors γ_{uk} can be estimated by assuming the existence of a set of communities, where each community models specific user attitudes. In particular, the probability of observing a user is given by the mixture

$$P(u|\mathcal{C}) = \sum_{j=1}^K P(u|c_j) \pi_j = \sum_{j=1}^K \prod_{i=1}^N \prod_{r=1}^V \pi_j (\alpha_{ij} \cdot \sigma_{rij})^{\delta(u, i, r)}$$

where a single community c_j is characterized by the parameters $\alpha_{ij} = P(i|c_j)$ and $\sigma_{rij} = P(r|c_j, i)$. The expected log likelihood can hence be defined as:

$$\mathcal{Q}(\mathbf{R}; \mathbf{\Gamma}) = \sum_{u=1}^M \sum_{j=1}^K \gamma_{uj} \cdot \left[\sum_{i=1}^N \sum_{r=1}^V \delta(u, i, r) \cdot (\log \alpha_{ij} + \log \sigma_{rij}) + \log \pi_j \right]$$

Estimating the parameters by means of an EM procedure yields the following equations:

E-Step:

$$\gamma_{uj} = P(c_j|u) = \frac{P(u|c_j) \cdot \pi_j}{\sum_{j'=1}^K P(u|c_{j'}) \cdot \pi_{j'}}$$

M-Step:

$$\pi_j = \frac{\sum_{u=1}^M \gamma_{uj}}{M}$$

$$\alpha_{ij} = \frac{\sum_{u=1}^M \gamma_{uj} \sum_{r=1}^V \delta(u, i, r)}{\sum_{u=1}^M \gamma_{uj} \sum_{i'=1}^N \sum_{r=1}^V \delta(u, i', r)} \quad \sigma_{rij} = \frac{\sum_{u=1}^M \gamma_{uj} \cdot \delta(u, i, r)}{\sum_{u=1}^M \sum_{r'=1}^V \gamma_{uj} \cdot \delta(u, i, r')}$$

A further advantage of the above formalization is the possibility of exploiting the above model for prediction purposes as well as for structure discovery. A prediction function in fact can be defined as

$$\hat{r}_i^u = E[R|u, i] = \sum_{r=1}^V r \cdot \sum_k \sigma_{rik} \cdot \gamma_{uk} \quad (5.7)$$

and used as a baseline for the special case described in step 10 of algorithm 2. We shall see in the following that the resulting baseline function is even competitive with state-of-the art approaches. The above formalization also allows an alternative gaussian model

$$P(r|i, c_j) = N(v_u^r; \mu_{ij}, \sigma_{ij}) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp \left[-\frac{(v_u^r - \mu_{ij})^2}{2\sigma_{ij}^2} \right]$$

where v_u^r is the Z-score normalization of r with regards to user u :

$$v_u^r = \frac{r - \mu_u}{\sigma_u}$$

and the means and the variances are estimated as proposed in [24], using a smoothing schema:

$$\mu_u = \frac{\sum_{i \in \mathcal{I}_u} (r_i^u + q\bar{\mu})}{|\mathcal{I}(u)| + q} \quad \sigma_u = \frac{\sum_{i \in \mathcal{I}_u} (r_i^u + q\bar{\mu})}{|\mathcal{I}(u)| + q}$$

The rating prediction for the pair (u, i) can be hence computed as:

$$\hat{r}_i^u = \mu_u + \sigma_u \left(\sum_{k=1}^K \gamma_{uk} \cdot \mu_{ik} \right) \quad (5.8)$$

Assuming a gaussian model for $P(R = r|\theta_j, i)$, the likelihood takes the following form:

$$ll(\Theta|U) = \sum_{u=1}^M \sum_{j=1}^K \gamma_{uj} \left[\log \pi_j + \sum_{i=1}^N \sum_{r=1}^V \delta(u, i, r) \cdot \left(\log \alpha_{ij} - \frac{1}{2} \log 2\pi - \frac{1}{2} \log \sigma_{ij} - \frac{(v - \mu_{ij})^2}{2\sigma_{ij}^2} \right) \right]$$

and the **M-Steps** can be rewritten as:

$$\mu_{ik} = \frac{\sum_u^M \sum_r^V \gamma_{uk} \cdot \delta(u, i, r) \cdot v_u^r}{\sum_u^M \sum_r^V \gamma_{uk} \delta(u, i, r)} \quad \sigma_{ik}^2 = \frac{\sum_{u=1}^M \sum_r^V \gamma_{uk} \cdot \delta(u, i, r) (v_u^r - \mu_{ik})^2}{\sum_{u=1}^M \sum_r^V \gamma_{uk} \delta(u, i, r)}$$

5.2.2 Local Community Patterns via Topic Analysis.

The approach to the discovery of local community patterns is based again on a EM procedure which aims at maximizing the likelihood of the $\mathbf{R}_k = \{\langle r, u, i \rangle | P(c_k|u) \geq P(c_j|u), j = 1, \dots, K\}$ rating matrix associated to a community model c_k . In practice, we can define the expected log-likelihood

$$\mathcal{Q}(\mathbf{R}_k; \Psi) = \sum_u^M \sum_i^N \sum_r^V \sum_h^{H_k} \psi_k(h; r, i, u) \cdot [\log \phi_h(r) + \log P_k(d_h|i) + \log P_k(d_h|u)]$$

where $\psi_k(h; r, i, u) = P(d_h|r, i, u, c_k)$. The EM algorithm can hence be defined in terms of the following formulas:

- **E-Step:**

$$\psi_k(h; r, i, u) = \frac{\phi_h(r) P_k(d_h|i) P_k(d_h|u)}{\sum_j \phi_j(r) P_k(d_j|i) P_k(d_j|u)}$$

- **M-Steps:**

$$\begin{aligned} P_k(d_h|i) &= \frac{\sum_u^M \sum_r^V \psi_k(h; r, i, u)}{\sum_{h'} \sum_u^M \sum_r^V \psi_k(h'; r, i, u)} \\ P_k(d_h|u) &= \frac{\sum_i^N \sum_r^V \psi_k(h; r, i, u)}{\sum_{h'} \sum_i^N \sum_r^V \psi_k(h'; r, i, u)} \\ P(r|d_h) &= N(r; \mu_{d_h}, \sigma_{d_h}) \end{aligned}$$

where

$$\begin{aligned} \mu_{d_h} &= \frac{\sum_u^M \sum_i^N \sum_r^V \psi_k(h; r, i, u) \delta(u, i, r) \cdot r}{\sum_u^M \sum_i^N \sum_r^V \psi_k(h; r, i, u) \delta(u, i, r)} \\ \sigma_{d_h} &= \frac{\sum_u^M \sum_i^N \sum_r^V \psi_k(h; r, i, u) \delta(u, i, r) \cdot (r - \mu_{d_h})^2}{\sum_u^M \sum_r^V \sum_i^N \psi_k(h; r, i, u) \delta(u, i, r)} \end{aligned}$$

5.2.3 Computational aspects.

Once the parameters of the hierarchical model have been estimated, the on-line complexity for computing predictions scales with the number of user communities and corresponding topics, while the off-line phase requires more resources. In fact, the complexity of the learning phase is determined by the complexity of discovering user communities, which is linear with the number of observed ratings.

To avoid overfitting, which could deteriorate the predictive skills of the models on unobserved data we adopt an *Early Stopping* criterion: a fraction

of the data has been retained as held-out dataset and the models have been trained on the remaining part of the data until the accuracy on the held-out data begins to increase.

The estimation of the correct number of clusters is accomplished by resorting to a Cross-Validation approach based on a penalized Log-Likelihood principle, as described below. Given a set D of observations (in our case, the rating matrix \mathbf{R} and its subsets \mathbf{R}_k), we aim at finding the model parameters Θ maximizing the probability $P(\Theta|D)$. In logarithmic terms,

$$\begin{aligned}\log(P(\Theta|D)) &= \log P(D|\Theta) + \log P(\Theta) \\ &= \log(\mathcal{L}(\Theta|D)) + \log P(\Theta)\end{aligned}$$

The idea in the above formula is to counterbalance two opposing requirements: the fitting of the data and the complexity of the model. By modeling $P(\Theta)$ can be modeled as an exponential distribution w.r.t the size of Θ , we can rewrite the above as

$$\log(P(\Theta|D)) \approx \log(\mathcal{L}(\Theta|D)) - m \log n$$

where m is the size of Θ (i.e., the number of model parameters), and n is the size of D . The evaluation strategy hence consists in computing $\log(P(\Theta|D))$ for each possible Θ , and in choosing the model where it is maximal. In particular, the strategy can be summarized as follows:

1. fix the values K_{min} and K_{max} ;
2. choose the number C of cross-validation trials;
3. for each trial c :
 - a) sample a subset D_{train} from D ;
 - b) for k ranging from K_{min} and K_{max} :
 - c) compute $\log(P(\Theta_k|D_{train}))^c$;
4. for each K , average the values $\log(P(\Theta_k|D_{train}))^c$ over c ;
5. choose the value k^* such that $\log(P(\Theta_{k^*}|D_{train}))^{avg}$ is maximal.

Discussion

There are several major differences between state-of-the art models and the above formalization. Considering pLSA, the hidden variable z there is used to discover similar trends in the rating behavior and encourages grouping users into user communities. The prediction relies solely on $P(r|i, z)$ and does not consider item hierarchies and, hence boosted predictions triggered by similar items. By contrast, the proposed hierarchical approach aims to discover local patterns for each user community. Also, there are two further components which boost the prediction accuracy of the underlying user community model.

First, the multinomial prior π_j for each user community j , which helps in preventing overfitting by counterbalancing the contribute of each user u in γ_{uj} . The π_j component can be interpreted as a laplacian smoothing based on uniform Dirichlet priors. Clearly, explicit modeling of such priors via Bayesian estimation, in the style of [38], can be adopted. However, as discussed in the next section, the computational cost would leverage significantly. Also, the α_{ij} component explicitly models the likelihood that item i has been rated within community j . The latter also is a major difference, at the user community level, with respect to the multinomial mixture and the User Rating Profile models, discussed in [30].

Also, notice that the co-clustering techniques discussed in the previous section, like the Flexible Mixture Model, assume the existence of a fixed partition both for user communities and for item categories. In our case instead, each user community is characterized by its own partition over the item-set with a flexible number of topics. In addition, co-clustering models only produce prediction on the basis of local contribution $P(r|c_k, d_h)$. By contrast, according to Eq. 5.5, our prediction benefits from both local and global information.

A final remark is concerned with the possibility of considering the proposed approach symmetrical. Our model starts with user communities and then generates topics. In theory a dual scheme could be viable as well, by first generating item categories and then specific user communities conditioned to item categories. However, duality only holds if the number of rows and columns of the rating matrix are of the same order of magnitude. In fact, the number of model parameters in an item-based mixture grows linearly to the number of users. If the number of items is significantly less than the number of users, this would cause the generation of few categories characterized by too many parameters (and as a consequence the resulting model would be prone to overfitting).

5.2.4 Evaluation

We evaluate the effectiveness of the proposed approach in two different respects:

- To measure the effectiveness of the User community model adopted in the first stage in discovering communities fitting the training data. Since each community should be able to model a user's preferences, it is interesting to measure the prediction accuracy of Eq. 5.7 and Eq. 5.8, which exploit the community mixtures.
- To measure the overall prediction accuracy of the hierarchical approach, and to compare it to other well-known approaches in the literature.

Additionally, as a paradigmatic example, we shall inspect the informative content of the structures discovered by the hierarchical mode proposed so far. We will show how the resulting community/topic model can provide significant informative content about the communities and the relevant topics discovered.

	Nextflix		MovieLens	
	Training Set	Test Set	Training Set	Test Set
Users	435,656	389,305	6,040	6,040
Items	2,961	2,961	3,706	3,308
Ratings	5,714,427	3,773,781	800,168	200,041
Avg ratings (user)	13.12	9.69	132.47	33,119
Avg ratings (item)	1929.90	1274.50	215.91	60.47
Min ratings (user)	1	1	5	12
Min ratings (item)	5	1	1	1
Max ratings (user)	957	691	2266	62
Max ratings (item)	64492	42780	2229	1199
Sparseness Coeff	0,9956		0,9643	

Table 5.1. Summary of the Data used for validation.

We used two popular benchmark datasets (Netflix and MovieLens) for rating prediction to validate the predictive performance of the proposed approach. In short, Netflix dataset contains over 100 million of ratings given by 480,189 users on a set of 17,770 movies, collected between October 1998 and December 2005. The Netflix Prize dataset has been the reference data for empirical comparisons of Collaborative Filtering algorithms during the last years, mainly for 3 reasons: (i) size of dataset and sparseness coefficient; (ii) availability of results from competitive algorithms; (iii) availability of a baseline score for the prediction error, achieved by a real RS (the Netflix Cinematch algorithm) on the same dataset. We exploited a subsample of the whole Netflix data, and partition it into training and test set, where the latter contains ratings given by a subset of the users in the training set over the same set of items. Info about this dataset are summarized in Table 5.1.

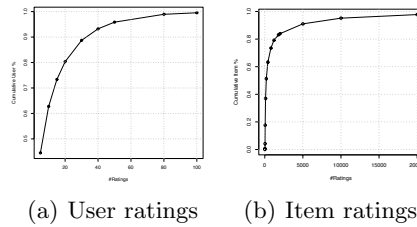


Fig. 5.2. ECDF for user and item ratings on NetFlix.

Figure 5.2 shows the empirical cumulative densities for both user and item ratings within the subsample adopted here. There are some major differences between the original Netflix dataset and the subsample used here. For example, we can see from Figure 5.2(a) that over 60% of the users have less than 10 ratings and the average number of evaluations given by users is 13 (whereas the original dataset exhibits an average 200 ratings). In addition, figure Figure 5.2(b) shows that over 50% of the items have received less than 200

ratings, with an average value of 1929. Again, the average ratings in the original dataset were 5000. In practice, the subsample we exploit is more difficult than the original dataset.¹

Predictive Accuracy.

We compare our approach with the most known latent factor approaches for rating prediction. In particular, we directly implemented the Regularized SVD [23], pLSA [28], FMM [51], Multinomial Mixture Model [30] and URP [38]. The summary of our results can be found in tables 5.2(a) and 5.2(b). Other algorithms not listed here will be discussed separately in the end of the section.

In a first set of experiments, we evaluate the performance achieved by the User Community Models, considering both the Multinomial and the Gaussian version and performed a suite of experiments varying the number of user communities and compared the obtained RMSE values with the ones achieved by the Gaussian pLSA algorithm on the same data.

Experiments on the three models were performed by retaining the 10% of the training (user,item,rating) triplets as held-out data; finally 10 attempts have been executed to determine the best initial configurations. Predictions for the User Community Models are generated according to Eq. 5.5, because preliminary experiments have shown that it outperforms the Hard-Clustering prediction rule. Performance results of the two User Communities Models and pLSA are shown in Figures 5.3(a) and 5.3(b).

Considering Netflix, the multinomial User Community approach and the pLSA do not produce a significant improvement over the Cinematch base, which is close to 0.95; for both these models the best RMSE values is achieved by considering 150 user communities. The average RMSE for the pLSA model is 0.9474 and only minor improvements on this result are observed varying the number of clusters. The gaussian User Community version outperforms both the multinomial model and pLSA, achieving the best RMSE value of 0.9280 when 30 user communities are employed. The learning phase corresponding to the best model takes about 30 minutes on a INTEL XEON E5520 at 2.27 Ghz, with an average of 6 iterations needed to reach convergence. We were not able to extensively report on FMM and URP on Netflix, due essentially to the high computational resources needed by these models. Table 5.2(a) shows the best result we were able to compute for FMM. For URP, the only model we were able to compute required 17,340secs and did not exhibit significant results. We were able, however, to thoroughly experiment on MovieLens with these models as well.

Surprisingly, the multinomial User Community model has a significant worsening on MovieLens. While the Gaussian model is still competitive, due essentially to the z-score normalization exploited in Eq. 5.8, the multinomial model seems to suffer more the skewness of the dataset.

¹ This also explains the difference between the values declared in the original papers by the competitors and the values we were able to reproduce on our subsample.

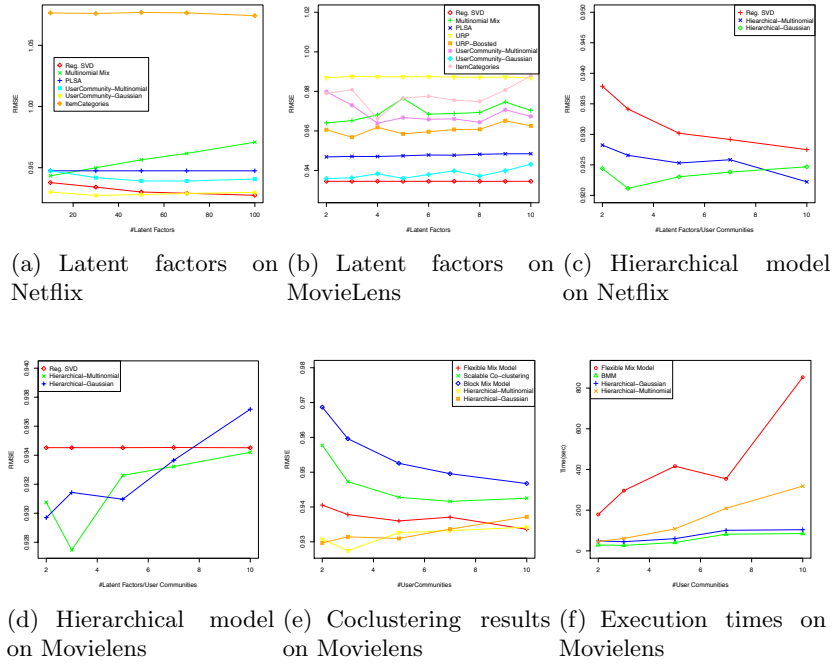


Fig. 5.3. Performance results.

The hierarchical schema allows us to obtain more refined results. This approach has been evaluated by considering both the multinomial and the gaussian version on the first layer clustering, and adopting the procedure for the dynamic estimation of the number of topics described in Sec. 5.2.3. Figure 5.3(d) and Figure 5.3(c) show the performances achieved by the two version and the ones achieved by a natural competitor based on latent factors: the regularized SVD. In both the cases, the hierarchical approach produces a significant improvement over the first clustering layer, outperforming the SVD model. On Netflix, hierarchical approach produces RMSE values 0.9222 (multinomial model) and 0.9211 (gaussian model), while the best result achieved by the SVD model is 0.9275. This situation is also reflected in MovieLens where the Reg. SVD produced 0.9345. Again, it's a surprise to see that in this case the multinomial hierarchical approach (0.9274) outperforms the Gaussian hierarchical (0.9296). This result is even more surprising, if we consider that the multinomial user communities didn't perform very well in the first level. It seems that the adoption of specific item categories boosts the performance significantly.

Figure 5.3(e) compares all the probabilistic approaches to co-clustering on MovieLens data. Here we compare our approaches with FMM, Bregman

(a) NextFlix Data

Approaches	Best RMSE	Parameters
Overall Mean	1.0839	
User Avg	1.0368	
Item Avg	1.009	
Knn Simple	1.0066	$K = 15$
Scalable Coclustering	0.9862	$K = 3, H = 5$
Weighted Centering	0.9707	$\alpha = 0.6$
Knn with Double Cen. Baseline	0.9637	$K = 20$
Flexible Mixture Model	0.9540	$K = 10, H = 70$
Block Mixture model	0.9477	$K = 30, H = 30$
PLSA	0.9474	$K = 100$
Knn with user effect baseline	0.9453	$K = 20$
Multinomial Mixture Model	0.9434	$K = 10$
User Communities Multinomial	0.9391	$K = 70$
Regularized SVD	0.9275	$\#features = 100$
User Communities Gaussian	0.9274	$K = 30$
KNN Relationship model	0.9258	$K = 20$
Hierarchical model Multinomial - Fixed	0.9251	$K = 50, H = 100$
Hierarchical model Multinomial - Flexible	0.9222	$K = 100$
Hierarchical Model Gaussian - Fixed	0.9212	$K = 50, H = 100$
Hierarchical Model Gaussian - Flexible	0.9211	$K = 30$

(b) MovieLens-1M Data

Approaches	Best RMSE	Parameters
Overall Mean	1.1150	
User Avg	1.0462	
URP	0.9869	$K = 10$
Item Avg	0.9862	
User Communities Multinomial	0.9638	$K = 4$
Multinomial Mix	0.9640	$K = 2$
Weighted Centering	0.961	$\alpha = 0.7$
URP - Boosted	0.9568	$K = 3$
PLSA	0.9468	$K = 2$
Regularized SVD	0.9345	$\#features = 8$
Block Mixture model	0.9467	$K = 10, H = 7$
Scalable Coclustering	0.9416	$K = 7, H = 5$
User Communities Gaussian	0.9359	$K = 2$
Flexible Mixture Model	0.9335	$K = 10, H = 10$
Hierarchical Model Gaussian - Fixed	0.9297	$K = 2, H = 2$
Hierarchical Model Gaussian - Flexible	0.9296	$K = 2$
Hierarchical model Multinomial - Fixed	0.9278	$K = 2, H = 3$
Hierarchical model Multinomial - Flexible	0.9274	$K = 3$

Table 5.2. Summary of the comparative analysis (K and H represent respectively the number of user communities and item topics)

Co-clustering [55] and Block Mixture Model [67]. Again, the hierarchical approaches outperform the other co-clustering approaches. This gives evidence that conditioning item categories to user communities provides better structures. Finally, Figure 5.3(f) shows the execution times of these co-clustering approaches. Here, we employ 10 item categories and vary the number of user communities.

A final validation qualitatively compares our approach with some among the most popular and effective approaches for making recommendations. We focused on single techniques rather than ensembles or combinations of multiple predictors.

Results on Netflix data show that the prediction accuracy achieved by the proposed model is competitive to the ones achieved by other popular recent

approaches, such as *PMF* [44], *Bi-LDA* [43] and *SVD++* [15]: the first one is reported to achieve on a sample of 1M ratings of Netflix data an RMSE equals to 0.9253; the latter achieves 0.9333 on the overall Netflix dataset. As far as the *SVD++* is concerned, although it achieves a 0.904 RMSE value on the considered dataset, the problem with such an approach is that it takes advantage of implicit information contained in the test-set.

The model also compares with *fLDA* [25] and the *Regression-based latent factor models* [69], which integrate user/item features and on a 75% – 25% split of the MovieLens-1M achieve 0.9381 and 0.9258 RMSE values.

Structure Discovery.

The hierarchical model can be used for classical pattern discovery tasks, such as the identification of the most appreciated items for each user community, as well a new kind of analysis, in which we focus on different topics and their impacts on the rating behavior of users within the same community. Table 5.3 shows a selection from the most significant items for 10 user communities and their topics. We show 5 communities only, and the 5 most relevant topics within them. An item i is considered significant with respect to a topic h within the community k if $P_k(d_h|i) > P_k(d_{h'}|i) \forall h' \neq h$. For each community we register its prior probability (in square brackets) and the a-posteriori interpretation of its topics.

For instance, user community #2 is characterized by the topics: “Fantasy”, “Sci-Fi”, “Live-Music Performance” “Action” and “Drama”. It is worth noticing how the informative content in the hierarchy allows to better discriminate among topics and tendencies. By focusing on the first level only, the same community would exhibit a global attitude towards action movies (as “Gladiator”, “Die Hard” and “Terminator 2” are the most probable items here).

5.3 A Bayesian Hierarchical Model for Preference Data

Cocustering approaches are based on the idea that similar users can be detected by taking into account their ratings on similar items, which in turn are identified considering the ratings assigned by similar users. A main weakness in these approaches is the static structure enforced by fixed row/column blocks where both users and items have to fit. In the previous chapter, we approached this problem and proposed the Hierarchical User Community Model (HUCM in the following) which is based on a dynamic hierarchy between user communities and item categories. In practice, there is a dependency relationship between latent factors on items and latent factors on users, which effectively overcomes the limits of a static structure enforced by fixed row/column blocks where both users and items have to fit.

When focusing on user communities only, HUCM is incidentally capable of explicitly modeling item selection, i.e., the probability that an item is actually

	Community 1 [0.09]	Community 2 [0.05]	Community 3 [0.17]	Community 4 [0.06]	Community 5 [0.04]
Topic 1	Curb Your-Enthusiasm The Office: Series 2 The Office Special Monty Python's Flying Circus	Star Wars: EplV The Incredibles The Princess Bride Lord of the Rings: The Two Towers	Gladiator The Shield Star Wars: EplV Saving-Private Ryan	The Best-of Friends Friends: S6 Gilmore Girls Friends: S5	It's a-Wonderful Life Star Wars: EplV Ben-Hur Gone with-the Wind
Interpr.:	Comedy	Fantasy	Action, war	Sitcom	Classic
Topic 2	Bruce Springsteen: Anthology 1978-2000 Karajan: Mozart: Don Giovanni Music of the Heart Music for Montserrat	Doctor Who: Pyramids of Mars Doctor Who: The Ribos Operation Battlestar Galactica Last Exile	Knowing Me-Knowing You Shag Aladdin Side Out	The Life and-Times of Frida Kahlo Birth of the-Blues / Blue Skies Julius Caesar American Dream	Blue's Clues: Shapes and Colors Yu-Gi-Oh! Sesame Street Black Beauty
Interpr.:	Music	Sci-Fi	Comedy	Documentary	For children
Topic 3	Glengarry-Glen Ross JFK Bataan Changing Lanes	Harry Connick Jr.: Only You Donna Summer: Live Ben Harper: Live Mozart: Don Giovanni	The Secret-Lives of Dentists Proof of Life The Ice Storm Body Story	Reservoir Dogs Get Shorty The Naked Gun SMM	Gone in-60 Seconds Intolerable Cruelty Confidence The Naked Gun
Interpr.:	Drama	Live performances	Drama	Crime	Crime
Topic 4	Highlander The Recruit Ali Rambo: First Blood	Robin Hood: Prince of Thieves Proof of Life Mission Impossible II Vanilla Sky	Equilibrium Ladder 49 Bad Company Waking Life	Amelle Victor / Victoria Princess Mononoke Sophie's Choice	A Midsummer-Night's Dream Chances Are Fools Rush In Mighty Aphrodite
Interpr.:	Action	Action, famous actors	Thriller	Romance	Comedy, Fantasy
Topic 5	Men in Black Alien Resurrection Spider-Man 2 X-Men: Evolution	Love Story Coffee and Cigarettes A Walk in the Clouds Hannah and-Her Sisters	13 Going on 30 Planet of the Apes Men in Black Rosemary-and Thyme	All the Pretty Horses Romeo Must Die Great Expectations The Manchurian-Candidate	The Parallax View Waterworld Romeo Must Die Swimming-with Sharks
Interpr.:	Action, Sci-Fi	Drama, romance	Fantasy, Comedy	Drama	Thriller

Table 5.3. User communities and relevant topics

selected by a user. While most of the conventional probabilistic techniques focus on forced-prediction, which explicitly requires to predict the preference value for each observed user-item pair, the non-hierarchical version of HUCM (referred to as UCM in the following, see Sec. 5.2.1) is capable to model item selection and rating prediction simultaneously.

To summarize, previous research devised two major contributions to the current literature. First, hierarchical probabilistic structures based on latent factor models can better model the underlying hidden relationships at the basis of users' behaviors. This allows to boost the prediction accuracy of such probabilistic models. Second, explicit modeling of item selection plays a crucial role with accurate recommendation lists. As shown in Chap. 4, a combined use of items selection and ranking prediction is crucial for providing accurate recommendation lists.

There is an apparent mismatch between these two situations. The explicit modeling of item selection boosts the accuracy of recommendation lists, yet it negatively impacts on prediction accuracy. The point is that exploiting item selection for ranking prediction in a (hierarchical) co-clustering model yields too many parameters to estimate, and consequently the risk of overfitting increases. As a matter of fact, the models achieving better prediction accuracy [38, 44, 43, 62] ignore the item selection components, whereas the models exhibiting the highest recommendation accuracy (such as *Pure-SVD* [50], *pLSA* [28], *LDA* [32] and *UCM*) provide poor performance in ranking prediction, or do not support it at all.

In this section we propose a new Bayesian Hierarchical latent factor model (*BH* in the following) which combines the advantages of both hierarchical modeling and item selection, and comparatively investigate both its recommendation accuracy and prediction error. *BH* relies on a generative process, which can take into account both item selection and rating emission, so that those users who experience the same items and tend to adopt the same rating pattern are gathered into communities. Individual users are modeled as a random mixture of communities, where the individual community is characterized again by a mixture of topics modeling both the popularity of items and the distribution over item ratings.

BH reinterprets the former HUCM in a Bayesian modeling setting, that is better suited to the sparseness of the preference data and less susceptible to overfitting. Additionally, *BH* allows a simpler and more elegant procedure for the estimation of model parameters through Gibbs sampling [63].

5.3.1 The Bayesian Hierarchical Model

In the following we extend the framework proposed in Sec. 5.2 by relaxing some basic conditions:

- users can exhibit diverse “dynamic” behaviors (in the style of [35]). That is, for each user there is no fixed community. Rather, the local behavior is picked randomly among the most probable.
- Analogously, items are dynamically associated with topics according to an underlying probability law.
- The overall process is governed by Bayesian priors thus allowing a more controlled modeling of data sparseness.

The key idea is that there exists a set of user communities, each one describing different tastes of users and their corresponding rating patterns. Each user community is then modeled as a random mixture over latent topics, which can be interpreted as item-categories. Given a user u , we can foresee his/her preferences on a set of items \mathcal{I}_u by choosing an appropriate user community z and then choosing an item category w for each item in the list. The choice of the item category w actually depends on the selected user community z . Finally the preference value is generated by considering the preference of users belonging to the group z on items of the category w . This local modeling of items is the main difference in the generative semantics with respect to state-of-the-art LDA based co-clustering approaches [43].

A first coarse-grained generative process directly derived from [62] can be devised as an adaptation of the well-know LDA-based models [32, 38], and is graphically depicted in Figure 5.4:

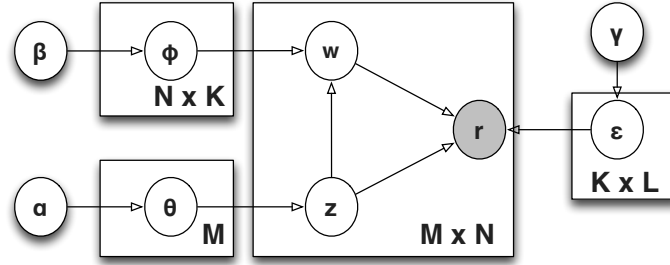


Fig. 5.4. *BH-Forced* Generative model

1. For each user $u \in \mathcal{U}$ sample user community-mixture components $\vartheta_u \sim Dir(\alpha)$;
2. For each item $i \in \mathcal{I}$ and user community $z \in \{1, \dots, K\}$ sample the mixture components $\varphi_{z,i} \sim Dir(\beta)$
3. For each topic $w \in \{1, \dots, L\}$ and user community $z = \{1, \dots, K\}$, sample rating probabilities $\varepsilon_{z,w} \sim Dir(\gamma)$
4. For each active pair $n = \langle u, i \rangle$ in \mathbf{R} :
 - a) Choose a user attitude $z_n \sim Discrete(\vartheta_u)$
 - b) Choose a topic $w_n \sim Multi(\varphi_{z_n,i})$
 - c) Generate a rating value for the chosen item according to the distribution $P(r|\varepsilon_{z_n,w_n})$

With respect to HUCM, that relies on maximum likelihood estimation with multinomial priors for model inference, the new Bayesian formulation (*BH-Forced* in the following) is both better suited to the sparsity of the rating matrix and less susceptible to overfitting. Moreover, it allows the development of a simpler and more elegant procedure for approximated parameter estimation based on Gibbs sampling [63]. Notice that, in the following, we model $P(r|\varepsilon_{z_n,w_n})$ as a multinomial over the parameter vector ε_{z_n,w_n} . Different choices can be made, in the style of [24], which are omitted here for lack of space.

Figure 5.5 shows how the rating matrix described in Figure 5.1 can be modeled according to *BH-Forced*. The figure summarizes a setting of the probability distributions for a *BH-Forced* Co-Clustering model compatible with the data represented in the previous example. By applying the generative process described above, the interested reader can easily verify that each observed rating can be replicated by drawing upon the corresponding distribution. For example, let us consider the observation $\langle u_5, i_5 \rangle$. According to the devised generative process, we first pick user community 2 for u_5 , exploiting table *c*. Next, we assign item category 1 to item i_5 , by drawing upon the available

categories according to the probability in table e. Finally, given the cocluster $\langle 2, 1 \rangle$, we observe rating 5 by picking randomly according to the related rating distribution in table f.

Again, it is worth noticing that the Bayesian Hierarchical model is more powerful, as it allows the modeling of complex relationships in a more dynamic scenario. As a matter of fact, users (resp. items) are not necessarily statically bound to a single community (resp. topic), but their membership can be dynamically modeled. In particular, for each pair $\langle u, i \rangle$ diverse user communities and item categories can be picked, according to the associated multinomial priors.

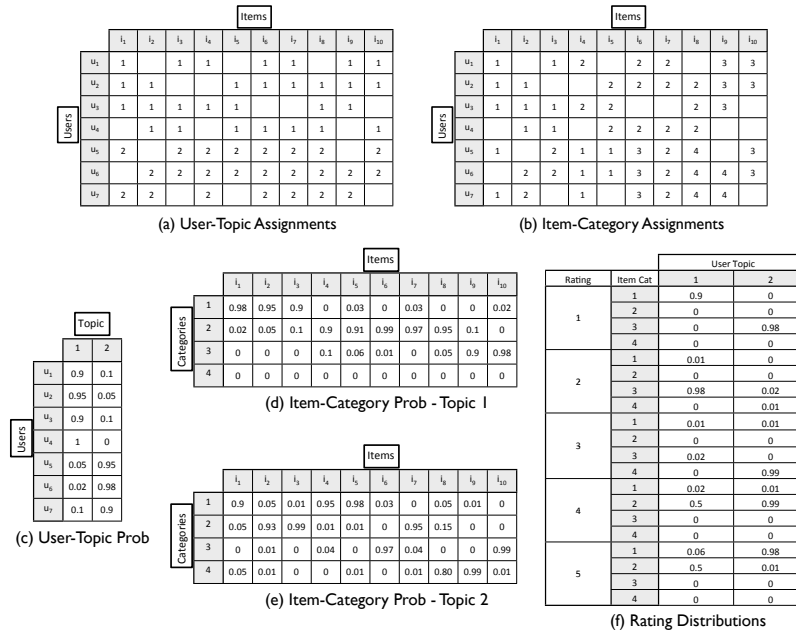


Fig. 5.5. Probabilistic modeling of local patterns

Modeling Free Prediction.

A problem with the BH model introduced so far is its focus on forced-prediction. That is, the model concentrates on the prediction of preference values for each observed user-item pair, and does not explicitly take into account item selection. As already mentioned, this component plays a crucial role in the generation of the recommendation list. Hence, it is likely to expect poor recommendation accuracy for this model.

The point is that the components in the *BH-Forced* model do not provide a direct support to the computation of $P(r, i|u)$. Thus, the only possibility for

BH-Forced is to generate a recommendation list by resorting to the expected-value.

We fix this issue by accommodating the hierarchical scheme in Figure 5.4 with an explicit item selection component. Specifically, each user is modeled as a random mixture of topics, where the individual topic is then characterized both by a distribution modeling item-popularity within the considered user-community and by a distribution over preference values for those items. In particular, the distribution of items given the topic variable w depends on the choice of the user community: this enforces an explicit modeling of item popularity both within a category and within a community, and hence provides a high degree of flexibility. Further, the rating prediction components maintain almost the same structure as in the *BH-Forced* model, and hence even the accuracy is almost the same.

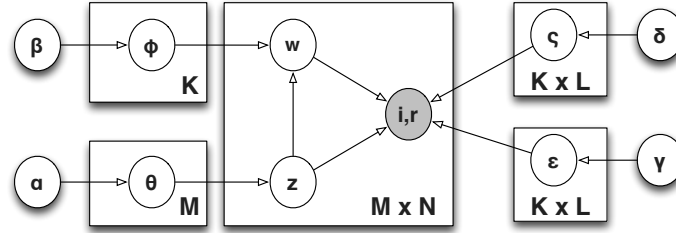


Fig. 5.6. *BH-Free* Model

The generative process for the new *BH-Free* model, whose corresponding graphical scheme is shown in Figure 5.6, is as follows:

1. For each user $u \in \mathcal{U}$ sample user community-mixture components $\vartheta_u \sim Dir(\alpha)$;
2. For each user community $z \in \{1, \dots, K\}$ sample the mixture components $\varphi_z \sim Dir(\beta)$
3. For each topic $w \in \{1, \dots, L\}$ and user community $z = \{1, \dots, K\}$,
 - a) Sample item selection components $\varsigma_{z,w} \sim Dir(\delta)$
 - b) Sample rating probabilities $\varepsilon_{z,w} \sim Dir(\gamma)$
4. For each $u \in \mathcal{U}$
 - a) Sample the number of items for the user u , $N_u \propto Poisson(\mathcal{K})$
 - b) For $n = 1$ to N_u
 - i. Choose a user attitude $z_{u,n} \sim Discrete(\vartheta_u)$
 - ii. Choose a topic $w_{u,n} \sim Multi(\varphi_{z_{u,n}})$
 - iii. Choose an item $i_n \sim Multi(\varsigma_{z_{u,n},w_{u,n}})$

- iv. Generate a rating value for the chosen item according to the distribution $P(r|\boldsymbol{\varepsilon}_{z_u,n,w_u,n})$.

BH-Free tries to infer the tendency of a user to experience some items over others independent of her/his rating values. The model assumes that this tendency is influenced by implicit and hidden factors which characterize each user community. To elucidate, a user may be pushed to experience a certain item because she/he belongs to a community in which the category of that item occurs with a high probability, although this has no impact on the rating assigned to the aforesaid item category. The probability of observing an item is independent from the rating assigned, given the state of the latent variables. This is a major difference with respect to most of the (co-clustering) models, which instead approach the problem from a *matrix approximation* perspective (as they focus on the prediction of r_i^u). By contrast, free-prediction models are focused on both the estimation of a rating behavior and the popularity of an item within each user community. An item which has received high ratings and has been experienced few times by the users belonging to the considered community could not have better chances of being recommended with respect to a popular item within the same community, which has received only ratings around the average.

It is worth noticing that support to free prediction was already included in the UCM model. And in fact, *BH-Free* can be considered as a substantial extension of the UCM model, in that it (i) adds a hierarchical co-clustering structure, thus complying to the originary idea of modeling local patterns; (ii) accommodates a Bayesian modeling which allows better control on data sparseness.

Inference and parameter estimation.

The inference process is similar for both *BH-Forced* and *BH-Free*. Concerning the *BH-Free* model, there's a small overhead due to the explicit modeling of item selection. Hence, in the following we shall only sketch the derivation of the sampling equations for this model. The equations for *BH-Forced* can be derived by resorting to similar techniques.

The notation used in our discussion is summarized in Table 5.4. Given the hyperparameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\delta}$ and $\boldsymbol{\gamma}$, the joint distribution of the data \mathbf{R} , the user-community mixtures Θ , the item-topic components Φ , the item and rating probabilities Σ and Γ and the observation-community/topic assignments Z, W , can be computed as:

$$P(\mathbf{R}, Z, W, \Theta, \Phi, \Sigma, \Gamma | \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}) = P(\mathbf{R} | Z, W, \Gamma, \Sigma) \cdot P(Z | \Theta) P(\Theta | \boldsymbol{\alpha}) \cdot P(W | Z, \Phi) P(\Phi | \boldsymbol{\beta}) \cdot P(\Gamma | \boldsymbol{\gamma}) \cdot P(\Sigma | \boldsymbol{\delta}) \quad (5.9)$$

SYMBOL	DESCRIPTION
M	#Users
N	# Items
\mathbf{R}	$M \times N$ Rating Matrix
K	# topics/user communities
L	# item categories
Θ	matrix $M \times K$ of parameters ϑ_u
ϑ_u	K -vector: mixing proportion of user-communities for the user u
Φ	Matrix of parameters φ_k
φ_k	L -vector: mixing proportion for the item category l and the user-topic k
Γ	Matrix of parameters $\epsilon_{k,l}$
$\epsilon_{k,l}$	V -vector: distribution over rating values for the co-cluster k, l
Σ	Matrix of parameters $\varsigma_{k,l}$
$\varsigma_{k,l}$	N -vector: mixing proportion for each item i in the co-cluster k, l
z	user-topic variable
Z	$M \times N$ matrix: user-topic assignments for each rating observation
w	item-categories topic variable
W	$M \times N$ matrix: item-categories assignments for each rating observation
α	K - vector: Dirichlet priors on user communities
β	L -vector: Dirichlet priors on item categories
γ	V -vector: Dirichlet priors on rating values
δ	N -vector: Dirichlet priors on items
n_u^k	# evaluation of the user u which have been assigned to the user topic k
$n_r^{k,l}$	# times that the rating r has been assigned to each observation when the user topic is k and the item category is l
$n_i^{k,l}$	# times that the item category l has been assigned to observations of the item i when the user topic is k
n_u	# observations for the user u ($ \mathcal{I}(u) $)
n_k	# observations associated with community k
$n_{k,l}$	# times that the category l has been assigned to observations whose user topic is k
\mathbf{n}_u	$\{n_u^k\}_{k=1}^K$
\mathbf{n}_k	$\{n_k^l\}_{l=1}^L$
$\mathbf{n}_k^{(V)}$	$\{n_r^{k,l}\}_{r=1}^V$
$\mathbf{n}_{k,l}^{(N)}$	$\{n_i^{k,l}\}_{i=1}^N$

Table 5.4. Summary of notation

The complete data likelihood can be obtained by integrating over Θ, Φ, Σ and Γ which, due to the conditional independence $\mathbf{R} \perp\!\!\!\perp \alpha, \beta | Z, W$ can be factored as:

$$P(\mathbf{R}, Z, W | \alpha, \beta, \gamma, \delta) = \int P(Z|\Theta)P(\Theta|\alpha)d\Theta \int P(W|Z\Phi)P(\Phi|\beta)d\Phi \int \int P(\mathbf{R}|Z, W, \Sigma, \Gamma)P(\Sigma|\delta)P(\Gamma|\gamma)d\Sigma d\Gamma$$

By rearranging the components and grouping the conjugate distributions, the complete data likelihood can be expressed as:

$$P(\mathbf{R}, Z, W | \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\sigma}) = \prod_{u=1}^M \frac{\Delta(\mathbf{n}_u + \boldsymbol{\alpha})}{\Delta(\boldsymbol{\alpha})} \cdot \prod_{k=1}^K \frac{\Delta(\mathbf{n}_k + \boldsymbol{\beta})}{\Delta(\boldsymbol{\beta})} \\ \cdot \prod_{k=1}^K \prod_{l=1}^L \frac{\Delta(\mathbf{n}_{k,l}^{(V)} + \boldsymbol{\gamma})}{\Delta(\boldsymbol{\gamma})} \cdot \prod_{k=1}^K \prod_{l=1}^L \frac{\Delta(\mathbf{n}_{k,l}^{(N)} + \boldsymbol{\delta})}{\Delta(\boldsymbol{\delta})}$$

The latter is the starting point for the inference of all the topics underlying the generative process, as the conditioned distribution on Z, W can be written as:

$$P(Z, W | \mathbf{R}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}) = \frac{P(Z, W, \mathbf{R} | \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta})}{P(\mathbf{R} | \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta})}$$

This formula is however intractable, mainly because the computation of the denominator requires a summation over an exponential number of terms. Gibbs Sampling [63] addresses this problem by defining a Markov chain, in which at each step inference can be accomplished by exploiting the full conditional $P(z_n = k_n, w_n = l_n | Z_{-n}, W_{-n}, \mathbf{R}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$. In the latter, z_n (resp. w_n) is the cell of the matrix Z (resp. W) which corresponds to this observation, and Z_{-n} (W_{-n}) denotes the remaining topic assignments. The chain is hence defined by iterating over the available states n . The Gibbs Sampling algorithm estimates the probability of assigning the pair k_n, l_n to the n -th observation, given the assignment corresponding to all the other rating observations:

$$P(z_n = k_n, w_n = l_n | Z_{-n}, W_{-n}, \mathbf{R}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \propto \\ \frac{n_{u_n}^{k_n} + \alpha_{k_n} - 1}{\sum_{k'=1}^K (n_{u_n}^{k'} + \alpha_{k'}) - 1} \cdot \frac{n_{k_n, l_n} + \beta_{l_n} - 1}{\sum_{l'=1}^L (n_{k_n}^{l'} + \beta_{l'}) - 1} \quad (5.10) \\ \cdot \frac{n_{r_n}^{k_n, l_n} + \gamma_{r_n} - 1}{\sum_{r=1}^V (n_r^{k_n, l_n} + \gamma_r) - 1} \cdot \frac{n_{i_n}^{k_n, l_n} + \delta_{i_n} - 1}{\sum_{i=1}^N (n_i^{k_n, l_n} + \delta_i) - 1}$$

Given the state of the Markov chain, denoted by $\mathcal{M} = (\mathbf{R}, Z, W)$, we can obtain the multinomial parameters Φ and Θ and Γ noticing that, by applying Bayes's rule and then by algebraic manipulations and the properties of the Dirichlet distribution [63]. This ultimately yields the following estimations:

$$\vartheta_{u,k} = \frac{n_u^k + \alpha_k}{n_u + \sum_{k=1}^K \alpha_k} \quad (5.11)$$

$$\varphi_{k,l} = \frac{n_{k,l} + \beta_l}{n_k + \sum_{l=1}^L \beta_l} \quad (5.12)$$

$$\epsilon_{k,l,r} = \frac{n_r^{k,l} + \gamma_r}{n_{k,l} + \sum_{r'=1}^V \gamma_{r'}} \quad (5.13)$$

$$\varsigma_{k,l,i} = \frac{n_i^{k,l} + \delta_i}{n_{k,l} + \sum_{i'=1}^N \delta_{i'}} \quad (5.14)$$

Finally, given the pair $\langle u, i \rangle$ we compute the probability of observing the rating value r in a *free prediction* context:

$$p(R = r, i|u) = \sum_{k=1}^K \sum_{l=1}^L \vartheta_{u,k} \cdot \varphi_{k,l} \cdot \varsigma_{k,l,i} \cdot \epsilon_{k,l,r} \quad (5.15)$$

Notice the explicit reference, in Eq. 5.15 to the $\varsigma_{k,l,i}$ component that models the probability of i being selected within co-cluster k, l . Clearly, such a component biases the ranking towards relevant items, thus providing the required adjustment that makes the model suitable for both prediction and recommendation accuracy.

The sampling equations for the *BH-Forced* model can be derived in a similar fashion:

$$P(z_n = k_n, w_n = l_n | Z_{-n}, W_{-n}, \mathbf{R}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \propto \frac{n_u^{k_n} + \alpha_{k_n} - 1}{\sum_{k'=1}^K (n_u^{k'} + \alpha_{k'}) - 1} \cdot \frac{n_{k_n, i}^{l_n} + \beta_{l_n} - 1}{\sum_{l'=1}^L (n_{k_n, i}^{l'} + \beta_{l'}) - 1} \cdot \frac{n_{r_n}^{k_n, l_n} + \gamma_{r_n} - 1}{\sum_{r=1}^V (n_{r_n}^{k_n, l_n} + \gamma_r) - 1} \quad (5.16)$$

with parameters

$$\vartheta_{u,k} = \frac{n_u^k + \alpha_k}{N_u + \sum_{k=1}^K \alpha_k} \quad (5.17)$$

$$\varphi_{k,l,i} = \frac{n_i^{k,l} + \beta_i}{N_{k,l} + \sum_{i=1}^N \beta_i} \quad (5.18)$$

$$\epsilon_{k,l,r} = \frac{n_r^{k,l} + \gamma_r}{N_{k,l} + \sum_{r'=1}^V \gamma_{r'}} \quad (5.19)$$

and prediction

$$p(R = r|u, i) = \sum_{k=1}^K \sum_{l=1}^L \vartheta_{u,k} \cdot \varphi_{k,l,i} \cdot \epsilon_{k,l,r} \quad (5.20)$$

A pseudo-code for the learning of the parameters is given in Algorithm 6.

5.3.2 Evaluation

In this section we comparatively evaluate the performance of the two BH models. The experiments are aimed at assessing the quality of the models in two different perspectives:

- From the *forced-prediction* viewpoint, we show that the *predictive accuracy* (i.e., the prediction error) exposed by both the *BH-Forced* and *BH-Free* models over unobserved ratings is comparable and in some cases even better than other state-of-the art probabilistic approaches.

Algorithm 6 The Gibbs-sampling procedure for parameter estimation within Bayesian Hierarchical Model (Forced)

Require: The sets $\mathcal{U} = \{u_1, \dots, u_M\}$ and $\mathcal{I} = \{i_1, \dots, i_N\}$
the rating matrix \mathbf{R} , the number of latent topics K , initial hyperparameters α, β and γ .

- 1: *initializeTopicAssignments()* {Randomly assign topics}
- 2: *iteration* \leftarrow 0
- 3: *converged* \leftarrow *false*
- 4: **while** *iteration* $<$ *nMaxIterations* **and** \neg *converged* **do**
- 5: **for all** $\langle u, i, r \rangle \in \mathbf{R}$ **do**
- 6: $z'_{u,i} \leftarrow \text{sampleTopic}(u, i, r)$ {According to Eq. 5.10};
- 7: update counts using the new topic for the observation $\langle u, i, r \rangle$
- 8: **end for**
- 9: *updateHyperParams()*
- 10: **if** (*iteration* $>$ *burnin*) **and** (*iteration* $\% \text{sampleLag} = 0$) **then**
- 11: *sampleUserTopicsMixingProbabilities()* {According to Eq. 5.17};
- 12: *sampleItemSelectionProbabilities()* {According to Eq. 5.18};
- 13: *sampleRatingProbabilities()* {According to Eq. 5.19};
- 14: *converged* \leftarrow *checkConvergence()*
- 15: **end if**
- 16: *iteration* \leftarrow *iteration* + 1
- 17: **end while**

- Conversely, from the *free-prediction* viewpoint, we show that *BH-Free* is the top-notch approach in terms of *recommendation accuracy*

We use two reference benchmark data sets, namely MovieLens-1M and a sample of Netflix data. Both datasets contain explicit preference data: ratings fall within the range 1 to 5, where the latter denotes the highest preference value. The main features of these datasets are summarized in Table 5.5.

We compare both models with some state-of-the-art competitors for CF recommendation, and in particular with co-clustering approaches. For the latter aspect, we compare with *LDCC* [40] (which extends the Bayesian co-clustering model proposed in [42] and it is based on a collapsed Gibbs sampling algorithm to perform parameter estimation and inference); with *Bregman-CC* proposed in [55] (which is based on the Bregman co-clustering algorithm); with *Bi-LDA* [43] (which extends the standard URP model [38] in both the user and item dimensions). Both models are implemented through a collapsed Gibbs sampling procedure, where we use uniform priors to initialize the hyperparameters of the models. We set the maximum number of iterations to 1000, where the first 100 as burn-in and a sample lag of 10 iterations. All models have been trained by retaining the 1% of the training data as held out to perform *early stopping* and avoid overfitting.

We also compare with the User community models previously defined: UCM, HUCM [62], and BUCM [70]. Whereas HUCM is a natural choice for comparison, (as the BH models represent a direct extension of such a model),

	Nextflix		MovieLens	
	Training Set	Test Set	Training Set	Test Set
Users	435,656	389,305	6,040	6,032
Items	2,961	2,961	3,706	3,444
Ratings	5,714,426	3,773,781	800,729	199,480
Avg ratings (user)	13	9	132	33
Avg ratings (item)	1929	1274	216	57
Min ratings (user)	1	1	11	1
Min ratings (item)	5	1	1	1
Max ratings (user)	957	691	1849	465
Max ratings (item)	64492	42780	2738	690
Sparseness Coeff	0,9957		0,9642	
% of *	4.55	4.53	5.62	5.58
% of **	10.06	10.06	10.76	10.74
% of ***	28.82	28.87	26.11	26.11
% of ****	33.33	33.39	34.89	34.89
% of *****	23.21	23.13	22.61	22.69

Table 5.5. Summary of the Data used for validation.

the UCM (and its Bayesian redefinition) explicitly model item selection and relevance ranking, and hence represent a reference comparison for the *BH-Free* model.

Predictive Accuracy.

We start our analysis from the evaluation of the prediction accuracy achieved by the algorithms. Table 5.6 summarizes the best RMSE obtained on both the considered datasets, together with the associated settings. To assess the effectiveness of all the considered approaches in rating prediction, we compare them with *Probabilistic Matrix Factorization (PMF)* [44], a cutting-edge probabilistic approach.

Approach	MovieLens		Netflix	
	Best RMSE	#Topics	Best RMSE	#Topics
<i>PMF</i>	0.8655	10	0.9309	100
<i>HUCM</i>	0.9278	2-3	0.9212	50-10
<i>Bregman-CC</i>	0.9023	10-20	0.9873	3-5
<i>Bi-LDA</i>	0.9033	30-20	0.9362	30-15
<i>LDCC</i>	0.9074	5-5	0.9419	5-10
<i>BH-Forced</i>	0.9041	15-3	0.9320	10-3
<i>BH-Free</i>	0.9073	30-5	0.9256	30-5
<i>BUCM</i>	0.9292	30	0.9431	10

Table 5.6. Summary of predictive accuracy over the MovieLens and Netflix datasets

As a general remark, both *BH-Forced* and *BH-Free* exhibit similar RMSE as other co-clustering approaches. *BH-Free* even outperforms all the other approaches on the NetFlix data, and is the runner-up winner after *HUCM* which, however, exhibits a marginal advantage. Minimal differences can also

be noticed on MovieLens, where *PMF* achieves the best RMSE score (as expected). In both datasets, *BUCM* is overcome by all other co-clustering methods: this proves that a hierarchical structure provides substantial information for boosting the accuracy of prediction.

Since the dependency between item categories and user communities tends to produce more complex structures with respect to traditional co-clustering approaches, it is important to evaluate the scalability of the *BH* models in this respect. Figure 5.7 shows how the RMSE scales with the number of item categories for the two *BH* models. *BH-Forced* globally achieves a lower RMSE, but tends to overfit the data with a larger number of such categories. This is clearly due to the huge number of parameters that the model induces: *BH-Forced* estimates the matrix $\{\varphi_{k,i,l}\}_{k=1,\dots,K;i=1,\dots,N;l=1,\dots,L}$ which is one order of magnitude bigger than the same matrix in the other co-clustering models (like *Bi-LDA* or *BH-Free*).

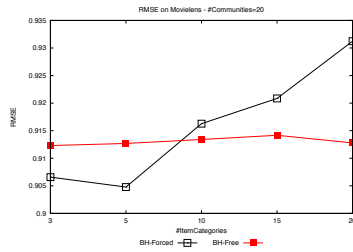


Fig. 5.7. RMSE on MovieLens data - Bayesian Hierarchical Model (#usercommunities=20)

As shown in Figure 5.8, the learning time of the *BH* models introduced a reasonable overhead with respect to the learning time of *LDCC*, when the number of item categories is less than 20.

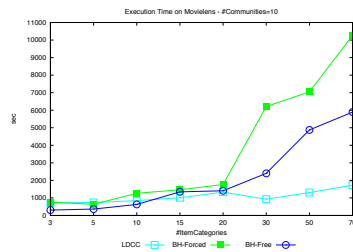


Fig. 5.8. Execution Time on MovieLens data

Recommendation Accuracy.

Things change substantially when considering the precision and recall accuracy metrics. Based on the results in [31, 70] (discussed in the previous chapter), we consider here also *LDA* model, which has been identified as one of the top-performers in terms of recommendation accuracy. Notice that *LDA* was not included in the analysis of predictive accuracy, as it does not explicitly support a way to compute rating prediction.²

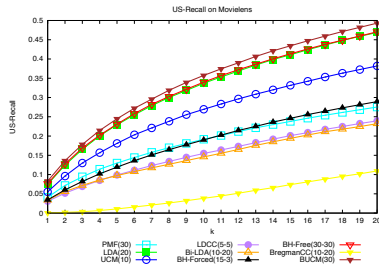
The recommendation list for traditional probabilistic approaches based on forced prediction is computed by sorting items according to the expected value. As far as *HUCM* is concerned, even if the overall model does not specify item-selection probabilities, these components are modeled explicitly by the simplified non-hierarchical (*B*)*UCM* versions. To summarize, we equip *LDA* with item selection ranking, and *UCM*, *BUCM* and *BH-Free* with item selection and relevance ranking. All the other approaches are based on the expected value.

Figure 5.3.2 shows the results of recommendation accuracy on MovieLens and Netflix data, when the size k of the list varies from 1 to 20. Probabilistic models equipped with item-selection achieve the best results in both datasets. On MovieLens data, *BH-Free* follows the same trend as *LDA* for user satisfaction, and exhibits a minimal worsening on standard recall (0.39 vs 0.37) and precision (0.11 vs 0.10). *BH-Forced* does not compare with item-selection methods, but achieves competitive results with the remaining probabilistic co-clustering approaches, outperforming them in user satisfaction recall. Notably, the discrepancy between the recommendation accuracy of Bayesian approaches and the non-bayesian ones is consistently large. In particular, both *BUCM* and **BH-Free** outperform *UCM*. This confirms the advantages of the Bayesian approach.

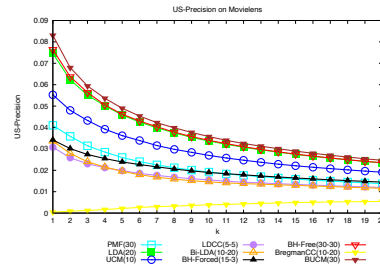
The trends are confirmed and even strengthened on Netflix data: approaches equipped with item-selection and relevance ranking, and in particular *BH-Free*, tend to outperform all the other approaches. *BH-Free* achieves the best recommendation accuracy and exhibits a global gain over both *UCM* and *BUCM*.

The outperformance of *BUCM* over *BH-Free* in MovieLens can be explained by the different distribution of these data with respect to Netflix. In this latter case, in fact, the huge volume of data is more likely to exhibit local patterns, which are better modeled by *BH-Free*. By converse, MovieLens exhibits both less users and less ratings, and hence the simpler *BUCM* model can easily fit the data.

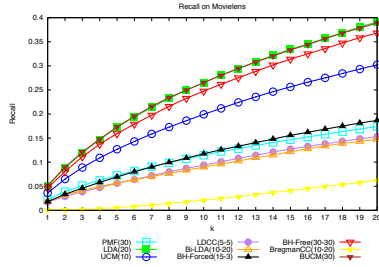
² As a matter of fact, extensions explicitly modeling such feature [38] have been experimentally shown in [31] to perform worse than *Ruslan:2008*.



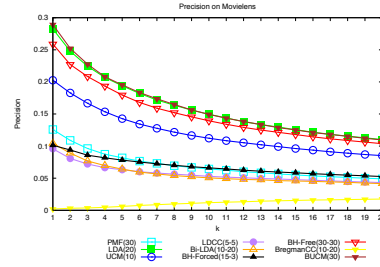
(a) US-Recall on MovieLens



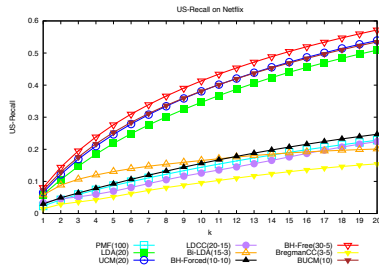
(b) US-Precision on MovieLens



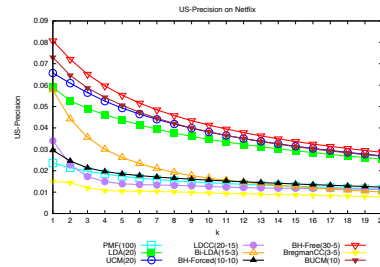
(c) Recall on MovieLens



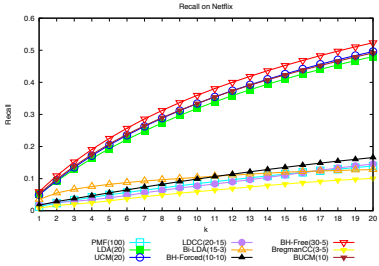
(d) Precision on MovieLens



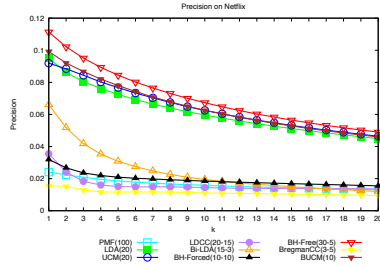
(e) US-Recall on Netflix



(f) US-Precision on Netflix



(g) Recall on Netflix



(h) Precision on Netflix

In this thesis, we presented and discussed about the main applications of probabilistic approaches to the recommendation problem. The probabilistic framework provides several advantages over other techniques, both in terms of modeling and accuracy of the results. The main advantages rely in their flexibility, as they allow to model different loss functions, and their capacity to model a distribution over preference values, which can be used to determine the confidence of the model in providing a recommendation. Moreover, since the recommendation task is essentially a prediction problem, the use of bayesian inference techniques is better suited to the sparsity of the preference data, as it better prevents overfitting. Besides the above mentioned, there are other significant advantages in the adoption of probabilistic models for recommendation. Recent studies pointed out that there is more in recommendation than just rating prediction. A successful recommendation should answer to the simple question ‘What is the user actually looking for?’ which is strictly tied with dynamic user profiling.

The recommendation problem has been traditionally interpreted as a missing value prediction problem, where we are asked to estimate the numerical users’ preference value on unseen items. The prediction accuracy is commonly used as a proxy metric to measure the effectiveness of the RS. However, recent studies have pointed out many limitations of this approach, showing that higher prediction accuracy does not imply higher recommendation accuracy. Motivated by these premises, we have investigated the performance of the main probabilistic approaches in terms of recommendation accuracy. In this study, we have shown that probabilistic models, equipped with the proper ranking function, exhibit competitive advantages over state-of-the-art RSs in terms of precision and recall of the recommendation list. In particular, strategies based on item selection guarantee significant improvements, and we have investigated the motivations behind the failure of prediction-based approaches.

Item selection plays the most important role in the design of an accurate personalized ranking function. However, the recommendation accuracy can be further boosted by computing the recommendation list according to the item selection and relevance ranking function, which estimates the probability that a user will play and like a given item. This approach combines the benefits of the modeling of preference values (explicit feedback) and of the implicit users’ selection of items (implicit feedback). Thus, we proposed a novel probabilistic model which is based on a free selection generative process. The Bayesian UCM takes into account both item selection and rating emission for the purpose of gathering those users who experience the same items and tend to adopt the same rating pattern into communities. By exploiting this joint modeling, the Bayesian UCM outperforms consistently state-of-art approaches to recommendation in terms of recommendation accuracy, while the item selection component tends to have a negative impact on the prediction accuracy.

The high dimensional preference data exhibit both global and local patterns, which can be identified and characterized by employing a hierarchical coclustering approach. In general, coclustering techniques are better suited to model the mutual relationship between users and items: similar users are detected by taking into account their ratings on similar items, which in turn are identified considering the ratings assigned by similar users. Hierarchical coclustering approaches overcome the limitation of enforcing fixed row/column blocks within the data, by allowing a more flexible structure, where item categories are conditioned by user communities. Due to the strong coupling between user communities and item categories, the exact inference for this hierarchical model is analytically and computationally compelling. We proposed two different solutions to the problem of learning the hierarchical coclustering structure. In the first proposal, we adopted an approximated solution, based on a hard clustering policy for user communities. This is a two-phase process in which we initially detect homogenous user communities, and then, for each of them, we detect a corresponding set of item categories. The second solution, is based on a hierarchical Bayesian approach for preference data, which allows the dynamic associations of user and item categories. Two versions of the general schema were proposed, namely *BH-Forced* and *BH-Free*, respectively based on the forced and free-prediction semantics. The experimental evaluation highlights the effectiveness of the proposed *BH* models, which represent the most satisfactory compromise between prediction and recommendation accuracy.

The proposed techniques can be extended in several directions, which we will discuss next.

Combining collaborative and content features

First of all, we are interested in combining in the same bayesian framework both collaborative and content features. This is expected to increase the accuracy of the recommendations provided by the system and the background content information can be used to provide personalized recommendations in cold-start scenarios. Moreover, side information can be exploited for a better identification and interpretation of user communities and item categories pattern. Finally, tag annotations define a general way to model each item/content/information. Hence, personalized techniques that are based on these information can be used in several context, from news and tweets, to pictures and video recommendations.

Social Recommender Systems

Users' behavior on web is more and more influenced by their social interactions with other users, and *social recommender systems* [71, 72] are emerging as a powerful combination of both recommendation and social networking features. In these systems, people *share* contents and information (generally indicated as memes), *interact* with their contacts, and *express* evaluations on

them, typically in the form of thumbs up/down. In this context, we are interested in providing an extension of the proposed framework which takes into account both users' past preferences and explicit people relationships to enhance recommendations. This implies a radical change in perspective: while in traditional personalized approaches we model users selection as a process that depends on her past purchase history, here we have to model users' choices as *social process*. The main differences with respect to traditional techniques for recommendation relies on the following points:

- Users' Influence : traditional approaches to recommendations assume that users' behavior is independent and identically distributed, thus ignoring social factors and common activities/interests among users. Users may be pushed to purchase an item under the influence of friends who have purchased the same item. Thus, at any given time t , each user u is subjected to her neighbors influence which may push/inhibit the selection of a generic item. Social influence [73, 74, 75] is believed to play an important role in users' behavior on SN, thus it will be indeed interesting to analyze and quantify its role in the item selection and evaluation process.
- Temporal Aspect : few efforts have been directed towards the modeling of temporal dimension in recommendations [76, 77, 78]. The temporal dimension is necessary to detect causal relationships between friends' actions: a user may be pushed to perform a generic action following her own behavior and preferences or under the influence of one of her friends. The analysis of the temporal aspect is crucial to determine and distinguish between social influence or simple correlation. Finally, users' taste and interests are expected to change in time. Dynamic topic models [79, 80] can provide an effective tool for the understanding of the evolution of users preference in time, as well for the detection of emerging trends and topics [81, 82].
- High Dynamism : New users/information join the network and this compromise the application of most of the traditional recommendation techniques, which are based on a finite set of users/items. SNs can be better represented as an high dynamic stream of both users and information. While several techniques have been proposed to deal with the "cold start" problem from a user perspective (recommendations to new users), few efforts have been directed towards the design and implementation of personalization techniques for *new items*.
- Heterogeneity : Almost all current recommender systems are designed for specific domains and applications without explicitly addressing the heterogeneity of the implicit and explicit preference information available on SNs. In particular, according to the Collaborative Filtering approach, state-of-art techniques for recommendations take into account only the past users' feedback to generate personalized suggestions. On the other hand, Social Network Recommender Systems are heterogeneous networks[83,

84], based on textual (reviews/tags), rating information and on social relationships.

Social recommendation over activity streams

Activity streams are one of the most important component in SNs. They allow us to learn about other users's recent behavior, activities, generally defined as feed. However, the flood on activity updates in current SN is huge and noisy. As direct consequence, users lose interest in following and reading the activity stream, and this is the first step towards the walking out from the SN.

The traditional technique to filter the activity stream is to select only the activities generated by friends but, as the number of friends increases with time, this approach does not alleviate the noise problem. Some friends can produce many non-interesting activities, dominating the stream and making difficult to find interesting items. Identifying and ranking relevant feed items is without any doubts a promising direction of research [85, 86, 87, 88], which combines both personalization and the modeling of users' influence and expertise.

Ranking

The probabilistic framework can be extended to model directly personalized ranking functions which take into account different factors, such as serendipity and novelty. The key idea is to maximize a ranking objective function instead of minimizing the conventional prediction error. Previous works in this direction have focused on learning a personalized ranking from from implicit feedback data [89, 90], while the application of the ranking based optimization criterion to explicit preference data needs still needs to be addressed.

References

1. Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.
2. Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web, WWW '07*, pages 271–280, 2007.
3. J. Ben Schafer, Joseph A. Konstan, and John Riedl. E-commerce recommendation applications. *Data Min. Knowl. Discov.*, 5(1-2):115–153, January 2001.
4. James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD, 2007*.
5. Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
6. George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management, CIKM '01*, pages 247–254, 2001.
7. Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. 2011.
8. David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35:61–70, 1992.
9. H. Lieberman. Letizia: An Agent that Assists Web Browsing. In *Proc. of Int. Joint Conf. on Artificial Intelligence*, pages 924 – 929, 1995.
10. Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2007.
11. John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52. Morgan Kaufmann, 1998.
12. Robert M. Bell and Yehuda Koren. Improved neighborhood-based collaborative filtering. In *KDD-Cup and Workshop, 2007*.

13. Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, pages 285–295, 2001.
14. Jon Herlocker, Joseph A. Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Inf. Retr.*, pages 287–310, 2002.
15. Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.
16. Robert M. Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM '07: Proc. of the 2007 Seventh IEEE International Conference on Data Mining*, pages 43–52. IEEE Computer Society, 2007.
17. Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD '07: Proc. of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104. ACM, 2007.
18. Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.
19. Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Rev.*, 37(4):573–595, 1995.
20. Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *ICML '98: Proc. of the Fifteenth International Conference on Machine Learning*, pages 46–54. Morgan Kaufmann Publishers Inc., 1998.
21. Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender systems case study. In *ACM WebKDD Workshop*, 2000.
22. Simon Funk. Netflix update: Try this at home. URL: <http://sifter.org/~simon/Journal/20061211.html>, 2006.
23. Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proc. of KDD Cup and Workshop*, 2007.
24. Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266. ACM, 2003.
25. Deepak Agarwal and Bee-Chung Chen. flda: matrix factorization through latent dirichlet allocation. In *WSDM '10*, pages 91–100, 2010.
26. David H. Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In *WWW '09*, pages 111–120, 2009.
27. C. Lawrence Zitnick and Takeo Kanade. Maximum entropy for collaborative filtering. In *UAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 636–643. AUAI Press, 2004.
28. Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.
29. Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

30. Benjamin Marlin. Collaborative filtering: A machine learning perspective. Technical report, Department of Computer Science University of Toronto, 2004.
31. Nicola Barbieri, Giuseppe Manco, and Ettore Ritacco. A probabilistic hierarchical approach for pattern discovery in collaborative filtering data (extended abstract). In Giansalvatore Mecca and Sergio Greco, editors, *SEBD*, pages 239–246, 2011.
32. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.
33. Mark Steyvers and Tom Griffiths. Probabilistic topic models. In T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum, 2006.
34. David M. Blei. Introduction to probabilistic topic models. *Communications of the ACM*, 2011.
35. Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693. Morgan Kaufmann Publishers Inc., 1999.
36. Thomas Hofmann. Learning what people (don't) want. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pages 214–225. Springer-Verlag, 2001.
37. L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *ICML*, pages 704–711. AAAI Press, 2003.
38. Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In *In NIPS*, 2003.
39. David M. Blei and Jon D. McAuliffe. Supervised topic models. In *NIPS '08*, 2008.
40. Pu Wang, Carlotta Domeniconi, and Kathryn Blackmond Laskey. Latent dirichlet bayesian co-clustering. In *ECML PKDD '09*, pages 522–537, 2009.
41. Hanhuai Shan and Arindam Banerjee. Bayesian co-clustering. In *ICDM '08*, pages 530–539, 2008.
42. M. Mahdi Shafiei and Evangelos E. Milios. Latent dirichlet co-clustering. In *ICDM '06*, pages 542–551, 2006.
43. Ian Porteous, Evgeniy Bart, and Max Welling. Multi-hdp: a non parametric bayesian model for tensor factorization. In *AAAI'08*, pages 1487–1490, 2008.
44. Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2008.
45. Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887, 2008.
46. Hanhuai Shan and Arindam Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM*, 2010.
47. Ryan Prescott Adams, George E. Dahl, and Iain Murray. Incorporating side information in probabilistic matrix factorization with gaussian processes. In *UAI*, pages 1–9, 2010.
48. Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM*, pages 403–414, 2012.
49. S.M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1097–1101, 2006.

50. Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46, 2010.
51. Xin Jin, Yanzan Zhou, and Bamshad Mobasher. Web usage mining based on probabilistic latent semantic analysis. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 197–205. ACM, 2004.
52. Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196, January 2001.
53. Govaert Gerard and Nadif Mohamed. Clustering with block mixture models. *Pattern Recognition*, 36(2):463–473, 2003.
54. Gerard Govaert and Mohamed Nadif. An em algorithm for the block mixture model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(4):643–647, 2005.
55. Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM*, pages 625–628, 2005.
56. Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Trans. Inf. Syst.*, 26:13:1–13:37, June 2008.
57. C. E. Shannon. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 30:50–64, 1951.
58. Thomas P. Minka. Estimating a dirichlet distribution. Technical report, Microsoft Research, 2000.
59. Y. Koren. How useful is a lower rmse?, 2007.
60. Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *ACM RecSys*, pages 257–260, 2010.
61. Nicola Barbieri. Regularized gibbs sampling for user profiling with soft constraints. In *ASONAM*, pages 129–136. IEEE Computer Society, 2011.
62. Nicola Barbieri, Giuseppe Manco, and Ettore Ritacco. A probabilistic hierarchical approach for pattern discovery in collaborative filtering data. In *SDM*, pages 630–621. SIAM / Omnipress, 2011.
63. Gregor Heinrich. Parameter Estimation for Text Analysis. Technical report, University of Leipzig, 2008.
64. Nicola Barbieri and Giuseppe Manco. An analysis of probabilistic methods for top-n recommendation in collaborative filtering. In Dimitrios Gunopoulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *ECML/PKDD (1)*, volume 6911 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2011.
65. Paul Resnick et al. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994.
66. Mohammad Khoshneshin and W. Nick Street. Incremental collaborative filtering via evolutionary co-clustering. In *RecSys*, pages 325–328, 2010.
67. Nicola Barbieri, Massimo Guarascio, and Giuseppe Manco. A block mixture model for pattern discovery in preference data. *2010 IEEE International Conference on Data Mining Workshops*, 0:1100–1107, 2010.
68. Nicola Barbieri, Gianni Costa, Giuseppe Manco, and Ettore Ritacco. Characterizing relationships through co-clustering - a probabilistic approach. In Joaquim Filipe and Ana L. N. Fred, editors, *KDIR*, pages 64–73. SciTePress, 2011.
69. Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.

70. Nicola Barbieri, Gianni Costa, Giuseppe Manco, and Riccardo Ortale. Modeling item selection and relevance for accurate recommendations: a bayesian approach. In *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys*, pages 21–28, 2011.
71. Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management, CIKM '08*, pages 931–940, 2008.
72. Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems, RecSys '10*, pages 135–142, 2010.
73. David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. 2003.
74. Jure Leskovec, Ajit Singh, and Jon M. Kleinberg. Patterns of influence in a recommendation network. In *Proc. of the 10th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, (PAKDD'06)*, 2006.
75. Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. Learning influence probabilities in social networks. In *Third ACM Int. Conf. on Web Search and Data Mining (WSDM'10)*, 2010.
76. Zhengdong Lu, Deepak Agarwal, and Inderjit S. Dhillon. A spatio-temporal approach to collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems, RecSys '09*, pages 13–20, 2009.
77. Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, pages 89–97, 2010.
78. Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10*, pages 210–217, 2010.
79. David M. Blei and John D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning, ICML '06*, pages 113–120, 2006.
80. Xuerui Wang and Andrew McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '06*, pages 424–433, 2006.
81. Shiva Prasad Kasiviswanathan, Prem Melville, Arindam Banerjee, and Vikas Sindhwani. Emerging topic detection using dictionary learning. In *Proceedings of the 20th ACM international conference on Information and knowledge management, CIKM '11*, pages 745–754, 2011.
82. Ankan Saha and Vikas Sindhwani. Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization. In *Proceedings of the fifth ACM international conference on Web search and data mining, WSDM '12*, pages 693–702, 2012.
83. Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Synthesis Lectures on Data Mining and Knowledge Discovery. 2012.
84. Lu Liu, Jie Tang, Jiawei Han, and Shiqiang Yang. Learning influence from heterogeneous social networks. *Data Min. Knowl. Discov.*, pages 511–544, 2012.

85. Jill Freyne, Shlomo Berkovsky, Elizabeth M. Daly, and Werner Geyer. Social networking feeds: recommending items of interest. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 277–280, 2010.
86. Ido Guy, Inbal Ronen, and Ariel Raviv. Personalized activity streams: sifting through the "river of news". In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 181–188, 2011.
87. Shlomo Berkovsky, Jill Freyne, Stephen Kimani, and Gregory Smith. Selecting items of relevance in social network feeds. In *Proceedings of the 19th international conference on User modeling, adaptation, and personalization*, UMAP'11, pages 329–334, 2011.
88. Shlomo Berkovsky, Jill Freyne, and Gregory Smith. Personalized network updates: increasing social interactions and contributions in social networks. In *Proceedings of the 20th international conference on User Modeling, Adaptation, and Personalization*, UMAP'12, pages 1–13, 2012.
89. Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, 2009.
90. Gábor Takács and Domonkos Tikk. Alternating least squares for personalized ranking. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 83–90, 2012.