

Latent Semantic Models for Collaborative Filtering

THOMAS HOFMANN
Brown University

Collaborative filtering aims at learning predictive models of user preferences, interests or behavior from community data, that is, a database of available user preferences. In this article, we describe a new family of model-based algorithms designed for this task. These algorithms rely on a statistical modelling technique that introduces latent class variables in a mixture model setting to discover user communities and prototypical interest profiles. We investigate several variations to deal with discrete and continuous response variables as well as with different objective functions. The main advantages of this technique over standard memory-based methods are higher accuracy, constant time prediction, and an explicit and compact model representation. The latter can also be used to mine for user communities. The experimental evaluation shows that substantial improvements in accuracy over existing methods and published results can be obtained.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*; I.5.3 [**Pattern Recognition**]: Clustering—*algorithms*

General Terms: Collaborative filtering, recommender systems, machine learning, mixture models, latent semantic analysis

1. INTRODUCTION

Content-based filtering and retrieval builds on the fundamental assumption that users are able to formulate queries that express their interests or information needs in term of intrinsic features of the items sought. In some cases, however, it may be difficult to identify suitable descriptors such as keywords, topics, genres, etc. that can be used to accurately describe interests. Yet in other cases, for example, in electronic commerce, users may be unaware or at least inattentive of their interest. In both cases, one would like to predict user preferences and recommend items without requiring the user to explicitly formulate a query.

Collaborative filtering is a technology that is complementary to content-based filtering and that aims at learning predictive models of user preferences,

This work was sponsored by NSF-ITR grants, award numbers IIS-0085836 and IIS-0085940.

Author's address: Department of Computer Science, Box 1910, Brown University, Providence, RI 02912.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or permissions@acm.org.

© 2004 ACM 1046-8188/04/0100-0089 \$5.00

interests or behavior from community data, that is, a database of available user preferences. Ideally, additional user input or interaction beyond the profile generated from previous interactions and observations is not necessary. Up to now, the dominant paradigm for performing collaborative filtering in recommender systems has been based on nearest neighbor regression or memory-based techniques. Virtually all first generation recommender systems have used the same fundamental two-step approach of first identifying users that are similar to some active user for which a recommendation has to be made, and then computing predictions and recommendations based on the preferences and judgments of these similar or like-minded users. The latter includes [Goldberg et al. 1992], the GroupLens (and MovieLens) project [Resnik et al. 1994; Konstan et al. 1997], Ringo [Shardanand and Maes 1995] as well as a number of commercial systems, most notably the systems deployed at Amazon.com and CDNow.com.

Memory-based methods have reached this level of popularity, because they are simple and intuitive on a conceptual level while avoiding the complications of a potentially expensive model-building stage. At the same time, they are deemed sufficient for many real-world problems. Yet there are a number of shortcomings, four of which we would like to point out here: (i) The accuracy obtained by memory-based methods may be suboptimal. Since recommendation accuracy is perhaps the most crucial factor from a user's perspective, improving accuracy is very important for most recommendation systems. (ii) Since no explicit statistical model is constructed, nothing is actually learned from the available user profiles and no general insight is gained. Hence, memory-based methods are only of limited use as data mining tools. (iii) Memory-based methods do not scale well in terms of their resource requirements (memory and computer time), unless further approximations—like subsampling—are made. (iv) It is difficult to systematically tailor memory-based algorithms to maximize the objective associated with a specific task.

This article deals with a model-based approach that addresses the above shortcomings and (i) achieves higher prediction accuracies, (ii) compresses the data into a compact statistical model that automatically identifies user communities, (iii) enables to compute preference predictions in constant time, and (iv) gives the system designer more flexibility in specifying the objectives of the application.

Model-based techniques have been investigated before, most notably Bayesian and non-Bayesian clustering techniques [Breese et al. 1998; Ungar and Foster 1998; Basu et al. 1998; Chien and George 1999], Bayesian networks [Breese et al. 1998], and dependency networks [Heckerman et al. 2000]. The approach proposed in this paper is a generalization of a statistical technique called probabilistic Latent Semantic Analysis (pLSA) [Hofmann 2001a] which was originally developed in the context of information retrieval [Hofmann 1999]. It bears some similarity with clustering methods such as distributional clustering [Pereira et al. 1993] in that latent variables for user communities are introduced, yet the communities can be overlapping and users are not partitioned into groups, not even probabilistically (cf. Hofmann and Puzicha [1999]). In fact, the probabilistic latent semantic models are in many ways closer related to dimension reduction methods and matrix decomposition techniques such as

Singular Value Decomposition (SVD) and Principal Component Analysis (PCA), which have been applied in information retrieval [Deerwester et al. 1990] as well as in the context of recommender systems [Sarwar et al. 2000; Goldberg et al. 2001; Canny 2002].

The main difference between our work and Bayesian or dependency networks is the fact that the latter learn a dependency structure directly on the observables, while our approach is based on a latent cause model that introduces the notion of user communities or groups of items. The main difference compared to PCA and SVD-based dimension reduction methods is that pLSA offers a probabilistic semantics and can build on statistical techniques for inference and model selection. However, our approach shares with all of the above techniques the assumption that predictions are computed in a “user-centric” view, whereas some more recent work has investigated item-based recommendation methods [Sarwar et al. 2001].

2. MODEL-BASED COLLABORATIVE FILTERING

2.1 Implicit and Explicit Ratings

The domains we consider consist of a set of persons or *users* $\mathcal{U} = \{u_1, \dots, u_m\}$, a set of *items* $\mathcal{Y} = \{y_1, \dots, y_m\}$ and a set of possible *ratings* \mathcal{V} . We assume observations are available for person/object pairs (u, y) , where $u \in \mathcal{U}$ and $y \in \mathcal{Y}$. In the most basic case, an observation will just be the co-occurrence of u and y , representing events like “person u buys product y ” or “person u clicks on link y ”, which is also sometimes called *implicit preference* data. Other cases may also provide an *explicit* rating $v \in \mathcal{V}$ as part of an observation. In the simplest case, this will be a binary response variable $v \in \{-1, 1\}$, modeling events like “person x likes/dislikes object y ”. In general, \mathcal{V} may be discrete or continuous, equipped with an ordinal or numerical (absolute) scale. For example, a five- or six-star rating scale as commonly used in movie recommendation systems such as MovieLens or EachMovie.

Rating data can be concisely summarized in table format as a n by m matrix \mathbf{A} , where each row will correspond to a user and each column to an item. In the case of implicit ratings, each entry a_{ij} represents a *count variable* of how often user u_i has selected item y_j or, more generally, how many pairs (u_i, y_j) have been observed. In the case of explicit ratings, each entry $a_{ij} \in \mathcal{V} \cup \{\emptyset\}$ will either correspond to a rating, $a_{ij} \in \mathcal{V}$ or will be unobserved, $a_{ij} = \emptyset$. Notice that the data matrix \mathbf{A} will typically be sparse in the sense that only a small fraction of pairs (u, y) are actually ever observed. Hence, the vast majority of entries a_{ij} will be 0 (implicit ratings) or \emptyset (explicit ratings).

2.2 Prediction Problems

We will consider two type of prediction problems. The first setting that we call *forced prediction* involves predicting a preference value for a particular item given the identity of the user, that is, one would like to learn a mapping $g : \mathcal{U} \times \mathcal{Y} \rightarrow \mathcal{V}$. More generally, one may be interested in the conditional probability $P(v|u, y)$ that user u will rate item y with v . Based on the

conditional probability one may also define a deterministic prediction function by $g(u, y) = \arg \max_v P(v|u, y)$. If v possesses a numerical scale, then it is more appropriate to define g via the expected rating, $g(u, y) = \sum_{v \in \mathcal{V}} vP(v|u, y)$ or $g(u, y) = \int_{\mathcal{V}} vP(v|u, y) dv$.¹ We call this setting forced prediction, because it mimics an experimental setup in which a user response is solicited for a particular item and the user has no choice on which item to vote. This is the relevant prediction mode in scenarios in which an item is presented to a user as a recommendation and one is interested in anticipating the user's response.

In the second setting, which we call *free prediction*, the item selection process is part of the predictive model and the goal is to learn probabilities $P(v, y|u)$ in order to predict both, the selected item y and (optionally) the associated rating v . By virtue of the chain rule, this can be rewritten as $P(v, y|u) = P(v|y, u)P(y|u)$, thus decomposing the problem into the prediction of the selected item (irrespective of the rating) and a prediction of the rating conditioned on the (hypothetically) selected item. This mimics a scenario in which the user is free to select an item of her or his choice and—in the case of explicit ratings—also provides a rating for it. The free prediction case is a generalization of what is commonly referred to as the “recommend” task, i.e. selecting a set of items to present to the user.

In the forced prediction case, the user is presented with a particular item and provides a rating for it. Here the selection of the item on which a user vote or response is solicited is part of the experimental design. In the free prediction case, the user is in control of the item selection and one is interested in predicting both, what a user will select and (optionally) how s/he will rate the item.

2.3 Loss and Risk Functions

Since we are pursuing a model-based approach to collaborative filtering, we will assume the availability of an adequate loss function. A loss function L is a function that quantifies how good or bad the prediction of a model is compared to a true outcome. We will denote the (parameterized) model space by \mathcal{H} and use a generic parameter θ to refer to a particular model in \mathcal{H} . Then, a *loss function* can be formally defined as a function $L : \mathcal{X} \times \mathcal{H} \rightarrow \mathfrak{R}$ where $\mathcal{X} = \mathcal{U} \times \mathcal{V} \times \mathcal{Y}$. Here \mathcal{V} is treated as void in the case of implicit ratings. Hence, for a given observation (u, v, y) , a loss function L will assign a score to every hypothesis θ under consideration. The smaller $L((u, v, y), \theta)$, the more compatible θ is believed to be with the observation.

In statistical inference, one often uses the (log-)likelihood as a criterion corresponding to a (negative) logarithmic loss

$$L^{\text{lg1}}((u, v, y), \theta) = -\log P(v|u, y; \theta), \quad \text{or} \quad L^{\text{lg2}}((u, v, y), \theta) = -\log P(v, y|u; \theta). \quad (1)$$

The first loss function in Eq. (1) is appropriate for the forced prediction scenario, since it conditions on the selected item y , while the second one corresponds to the free prediction mode in which y is part of the prediction.

¹For notational convenience, $P(v|u, y)$ denotes a probability mass function (discrete case) or a conditional probability density function (continuous case) dependent on the context.

Another popular choice for the case of discrete response variables and for models that make deterministic predictions is the zero-one loss,

$$L^{\text{zo}}((u, v, y), \theta) = 1 - \llbracket v = g(u, y; \theta) \rrbracket, \quad (2)$$

where $\llbracket \cdot \rrbracket$ denotes the indicator function of the enclosed predicate. If the prediction is correct, no loss is incurred, otherwise the loss is one. This loss function is also commonly used in supervised classification or pattern recognition. It can be generalized to the case of probabilistic models where one may define a loss via the probability of making an incorrect prediction, that is, the probability of error

$$L^{\text{pe}}((u, v, y), \theta) = \sum_{v' \neq v} P(v'|u, y; \theta) = 1 - P(v|u, y; \theta) \quad (3)$$

The logarithmic loss provides an upper bound on the probability of error, since

$$\begin{aligned} L^{\text{pe}}((u, y, v), \theta) &= 1 - P(v|u, y; \theta) \leq 1 - \log P(v|u, y; \theta) \\ &= 1 + L^{\text{lg}}((u, y, v), \theta). \end{aligned} \quad (4)$$

The zero-one loss is sometimes difficult to optimize directly, because it is a non-differentiable function of θ , in which case the use of the logarithmic loss can be advantageous for computational reasons. In fact, in this article, we focus exclusively on the use of logarithmic loss functions, which can be optimized (in approximation) with the well-known Expectation–Maximization (EM) algorithm for the proposed latent class models.

For numeric response variables, it is more common to use a metric-based loss function, for example, the absolute loss

$$L^{\text{abs}}((u, v, y), \theta) = |v - g(u, y; \theta)| \quad (5)$$

or the squared loss

$$L^{\text{sq}}((u, v, y), \theta) = (v - g(u, y; \theta))^2. \quad (6)$$

These loss functions have been used extensively for evaluating the accuracy of collaborative filtering methods, in particular memory-based methods, and we will also use them in our experiments. For completeness, we would like to mention the RankBoost method [Freund et al. 1998] which aims at minimizing an upper bound on the ranking loss. The latter uses a purely ordinal scale for the ratings and can be defined via the number of misordered item pairs.

A loss function scores models based on a single observation; however, we need to specify how to combine data consisting of several observations. Put differently, we need a sampling model to specify under which distribution $P(u, v, y)$ we would like to minimize the loss. This is usually called a *risk function* or *functional*, $\mathcal{R}(\theta) \equiv \sum_{u,v,y} P(u, v, y) L((u, v, y); \theta)$, where part of the sum has to be replaced by an integral in the case of continuous response variables v . A typical choice is to minimize the empirical loss, that is,

$$\mathcal{R}^{\text{emp}}(\theta) = \frac{1}{N} \sum_{(u,v,y)} L((u, v, y), \theta), \quad (7)$$

where angular brackets under a summation symbol are used as a shorthand notation to refer to all observation triplets and N denotes the total number of observed triplets. However, in collaborative filtering, it is a conceivable alternative to give the same weight to every user, irrespective of the number of implicit or explicit ratings available for that user. If we denote by n_u the number of observation triplets for user u , then this would correspond to the normalized empirical risk function

$$\tilde{\mathcal{R}}^{\text{emp}}(\theta) = \frac{1}{n} \sum_u L_u(\theta), \quad L_u(\theta) = \frac{1}{n_u} \sum_{(u', v, y): u'=u} L((u, v, y), \theta). \quad (8)$$

The choice of a normalized vs. nonnormalized risk function depends on the application. If we assume that users for which more data is available are more important, in the sense that it is more likely that we will have to make predictions for them again, then the unnormalized risk function in Eq. (7) may be more appropriate. Notice also that $\tilde{\mathcal{R}}^{\text{emp}}$ may put a lot of weight on individual observations, just because the data for some users may be sparse. Hence, we expect the normalized risk function to be more susceptible to overfitting, which has turned out to be disadvantageous in our experiments (cf. Section 5).

3. CO-OCCURRENCE LATENT SEMANTIC MODEL

3.1 Model Definition

We would like to discuss a simple model for co-occurrence data first, which is known as probabilistic latent semantic analysis (pLSA) [Hofmann 2001a; Hofmann 1999]. This can be thought of as a special case of collaborative filtering with implicit preference data [Hofmann and Puzicha 1999]. The data thus consists of a set of user-item pairs (u, y) which are assumed to be generated independently. The key idea of our approach is to introduce *hidden variables* Z with states z for every user-item pair, so that user u and item y are rendered conditionally independent. The possible set of states z is assumed to be finite and of size k . The resulting model is a mixture model that can be written in the following way

$$P(u, y; \theta) = \sum_z P(u, y, z) = \sum_z P(y|z) P(z|u) P(u), \quad (9)$$

where sums over z run over all possible k states. By applying Bayes' rule, one can alternatively use the equivalent parameterizations $P(u, y; \theta') = \sum_z P(z) P(u|z) P(y|z)$ and $P(u, y; \theta'') = \sum_z P(u|z) P(z|y) P(y)$. Since the more typical situation in collaborative filtering is to make personalized, that is, user-specific recommendations, we will mainly work with the conditional model

$$P(y|u; \theta) = \sum_z P(y|z) P(z|u). \quad (10)$$

In this model, the parameter vector θ summarizes the probabilities $P(z|u)$ which can be described by $(k-1) \times n$ independent parameters as well as $P(y|z)$ which requires $(m-1) \times k$ independent parameters, where again k denotes the number of possible states of the hidden variable.

Notice that if $k = 1$, then the model simply assumes that the selection of an item y does not depend on the identity of the user, $P(y|u) = P(y)$, resulting in non-personalized predictions. The user identity and the item identity are assumed to be marginally independent in this case. As the number of hidden states increases, the set of representable joint distribution over user–item pairs becomes less and less constrained until a fully saturated model is obtained, which can represent any probability mass function over user–item pairs. In practice, k has to be chosen in a way that adjusts model complexity in the light of the amount and sparseness of available data. Standard model selection techniques like cross-validation are available to that extend.

While we have not associated any a priori meaning with the states of the hidden variables, the hope though is to recover interesting structure in the data about user communities and groups of related items. Intuitively, the state z of a hidden variable Z associated with an observation (u, y) is supposed to model a hidden cause, that is, the fact that a person u selects item y “because of” z . Each z is intended to offer a hypothetical explanation for an implicit rating that is itself not directly observable. Since the number of possible states k is typically much smaller than the number of items and users, the model encourages to group users into user communities and items into groups of related items.

3.2 Expectation Maximization Algorithm

Following the maximum likelihood approach to statistical inference, we propose to fit the model parameters θ by maximizing the (conditional) log-likelihood, or equivalently, by minimizing the empirical logarithmic loss

$$\mathcal{R}(\theta) = -\frac{1}{N} \sum_{\langle u, y \rangle} \log P(y|u; \theta) = -\frac{1}{N} \sum_{i=1}^n \sum_{j=1}^m a_{ij} \log P(y_j|u_i; \theta) \quad (11)$$

where a_{ij} counts the number of times each pair (u_i, y_j) has been observed. Notice that for user–items pairs that have never been observed one gets $a_{ij} = 0$ and hence the number of terms in the double sum in Eq. (11) depends on the number of nonzero entries in the data matrix \mathbf{A} which is upper bounded by N and which can be far less than $n \times m$.

The Expectation Maximization (EM) algorithm [Dempster et al. 1977] is a standard method for statistical inference that can be used to (approximately) maximize the log-likelihood in mixture models like pLSA [Hofmann 2001a]. The first step in deriving an EM algorithm is to specify a complete data model. A complete data model treats the hidden variables *as if* they were actually observed, which in our case amounts to the assumption that for every observed pair (u, y) , we would in fact observe a triplet (u, y, z) . The complete data model corresponding to Eq. (10) is given by $P(y, z|u) = P(y|z)P(z|u)$ and the corresponding (negative) log-likelihood function can be written as

$$\mathcal{R}^c(\theta) = -\frac{1}{N} \sum_{\langle u, y, z \rangle} [\log P(y|z) + \log P(z|u)]. \quad (12)$$

Since the states of the latent variables are not known, we introduce a so-called *variational probability distribution* $Q(z; u, y)$ [Neal and Hinton 1998] for every

observed user item pair. Intuitively, the Q distribution will model our best knowledge about the states of the latent variables given the current parameters. If we identify the latter with user communities, then $Q(z; u, y)$ will denote the probability that the co-occurrence of (u, y) , that is, the selection of item y by user u , will be attributed to the fact that u is a member of community z .

Using Q one can define a family of risk functions (one risk function for every choice of Q)

$$\bar{\mathcal{R}}(\theta, Q) = -\frac{1}{N} \sum_{(u,y)} \sum_z Q(z; u, y) [\log P(y|z) + \log P(z|u)]. \quad (13)$$

Exploiting the concavity of the logarithm and using Jensen's inequality (cf. Cover and Thomas [1991]), it can be shown that every $\bar{\mathcal{R}}(\cdot, Q)$ defines an upper bound on $\mathcal{R}(\cdot)$ (up to a constant that only depends on Q),

$$\mathcal{R}(\theta) = -\frac{1}{N} \sum_{(u,y)} \log \sum_z Q(z; u, y) \frac{P(y|z)P(z|u)}{Q(z; u, y)} \quad (14a)$$

$$\leq -\frac{1}{N} \sum_{(u,y)} \sum_z Q(z; u, y) \log \frac{P(y|z)P(z|u)}{Q(z; u, y)} \quad (14b)$$

$$= \bar{\mathcal{R}}(\theta, Q) - \frac{1}{N} \sum_{(u,y)} H(Q(\cdot; u, y)), \quad (14c)$$

where $H(Q)$ refers to the entropy of a probability distribution Q .

The EM algorithm now consists of two steps that are performed in alternation: (i) computing the tightest bound for given parameters $\hat{\theta}$ and (ii) optimizing this bound with respect to θ . The first step consists of minimizing Eq. (14c) with respect to the variational distribution Q . This is called the E-step and amounts to computing the posterior probabilities of the hidden variables. Thus, for given parameters $\hat{\theta}$, the optimal Q^* —denoted by \hat{Q}^* —is given by

$$Q^*(z; u, y; \hat{\theta}) = P(z|u, y; \hat{\theta}) = \frac{\hat{P}(y|z)\hat{P}(z|u)}{\sum_{z'} \hat{P}(y|z')\hat{P}(z'|u)}. \quad (15)$$

A formal derivation using the technique of Lagrange multipliers is included in the appendix. The hat on probabilities in Eq. (15) denotes quantities parameterized by $\hat{\theta}$. Obviously, the posterior probabilities need only to be computed for user-item pairs (u, y) that have actually been observed. Averaging $\bar{\mathcal{R}}^c$ with respect to the posterior distribution calculated from Eq. (15) then yields the following upper bound on the negative log-likelihood function

$$\bar{\mathcal{R}}(\theta, \hat{\theta}) \equiv \bar{\mathcal{R}}(\theta, Q^*) = -\frac{1}{N} \sum_{(u,y)} \sum_z Q^*(z; u, y, \hat{\theta}) [\log P(y|z) + \log P(z|u)] \quad (16a)$$

$$= -\frac{1}{N} \sum_{(u,y)} \sum_z \frac{\hat{P}(y|z)\hat{P}(z|u)}{\sum_{z'} \hat{P}(y|z')\hat{P}(z'|u)} [\log P(y|z) + \log P(z|u)], \quad (16b)$$

which needs to be optimized with respect to the parameters θ in the Maximization (M) step of EM. The M-step requires to solve a constrained optimization

problem (cf. appendix for details) leading to the set of equations

$$P(y|z) = \frac{\sum_{\langle u, y \rangle: y'=y} Q^*(z; u, y, \hat{\theta})}{\sum_{\langle u, y \rangle} Q^*(z; u, y, \hat{\theta})} \quad (17a)$$

$$P(z|u) = \frac{\sum_{\langle u, y \rangle: u'=u} Q^*(z; u, y, \hat{\theta})}{\sum_z \sum_{\langle u, y \rangle: u'=u} Q^*(z; u, y, \hat{\theta})} = \frac{\sum_{\langle u, y \rangle: u'=u} Q^*(z; u, y, \hat{\theta})}{|\{\langle u, y \rangle : u' = u\}|}. \quad (17b)$$

The complete EM algorithm now proceeds by alternating the E-step in Eq. (15) with the M-step in Eq. (17).

3.3 Regularized Risk Functions

Learning statistical models with many parameters from a limited amount of data bears the risk of overfitting. Traditional model selection techniques would fit models by maximum likelihood and then determine the generalization performance of the model either analytically (typically in an asymptotic approximation) or via empirical evaluation using hold-out data or cross-validation. As an alternative approach we have proposed a technique called tempered EM [Hofmann 2001a] which minimizes a regularized risk function instead of the empirical risk (i.e. the negative log-likelihood in the case of maximum likelihood estimation). Formally, a β -parameterized family of regularized risk functions can be obtained by generalizing the upper bound in Eq. (14c)

$$\tilde{\mathcal{R}}_\beta(\theta, Q) \equiv \bar{\mathcal{R}}(\theta, Q) - \frac{1}{\beta} \sum_{\langle u, y \rangle} H(Q(\cdot; u, y)) \quad (18)$$

Notice that for $\beta = 1$ this reduces to maximum likelihood estimation via EM. For $\beta < 1$ more weight is put on the entropy of Q which avoids “over-confidence” in computing the posterior probabilities as can be seen from the solution of the generalized E-step (cf. appendix)

$$Q^*(z; u, y, \hat{\theta}) \propto (\hat{P}(z|u)\hat{P}(y|z))^\beta. \quad (19)$$

As a result, the optimal Q -distributions will be more smeared-out or *fuzzy* which counteracts overfitting as we will demonstrate in the experiments. Similar “tricks” have been used in speech recognition and other applications involving high-dimensional statistical models, in particular to compensate for simplifying assumptions about statistical independence. Notice that one of the advantages of tempered EM is the fact that the M-step is unaffected by the choice of β , hence one only has to modify the E-step.

A more rigorous framework is offered by the framework of Bayesian learning. As has been proposed in Blei et al. [2002] one can put Dirichlet priors on the multinomial distributions $P(z|u)$ and integrate them out, resulting in a model that has been called latent Dirichlet allocation (LDA) model. However, inference is significantly harder in this setting and further approximation are necessary to derive a tractable algorithm [Blei et al. 2002; Minka and Lafferty 2002].

3.4 Mixture Models, Clustering, and Dimension Reduction

It is important not to confuse the pLSA model with probabilistic or Bayesian clustering models [Breese et al. 1998; Chien and George 1999] in which it is assumed that each user belongs to exactly one user group. In a standard user clustering model, one introduces a *single* latent cluster membership variable for every user u , while the pLSA model associates a latent variable with every observation triplet (u, v, y) . Hence, different ratings of the same user can be explained by different latent causes in pLSA, whereas a user clustering model assumes that all ratings involving the same user are linked to the same underlying community. This can be stated more formally by computing and comparing the probability of a set of observations involving a particular user. The clustering model yields the following probability for the ratings of a fixed user u

$$P((v_1, y_1), \dots, (v_l, y_l)) = \sum_z P(z) \prod_i P(v_i, y_i|z). \quad (20)$$

In contrast, in the pLSA model each user is characterized by a distribution $P(z|u)$ and one gets a user-specific expression

$$P_u((v_1, y_1), \dots, (v_l, y_l)) = \prod_i \sum_z P(z|u) P(v_i, y_i|z). \quad (21)$$

By integrating out the mixture proportions $P(z|u)$ in a Bayesian manner, one can also define a generative model that does not have any reference to a specific user,

$$P((v_1, y_1), \dots, (v_l, y_l)) = \int_{\Theta} \left[\prod_i \sum_z \theta_z P(v_i, y_i|z) \right] p(\theta) d\theta. \quad (22)$$

Here, θ_z with $\theta_z \geq 0$ and $\sum_z \theta_z = 1$ takes the role of a latent parameter, which is averaged over using a prior probability density function $p(\theta)$. If the latter is chosen to be a Dirichlet distribution, one gets the LDA model of Blei et al. [2002]. In this article, we have focused on maximum likelihood estimation of $P(z|u)$, because statistical inference turns out to be significantly easier than in the fully Bayesian model.

4. LATENT SEMANTIC MODELS WITH RATINGS

4.1 Model Definition and Dependency Structures

Since many applications of collaborative filtering involve explicit user ratings, the pLSA model needs to be extended appropriately. We will focus first on the case, where ratings are predicted for fixed items (forced prediction). There are two different ways to augment the pLSA model with an additional random variable v for explicit ratings, as shown in Figures 1(e) and 1(f). The predicted rating will depend on the latent variable z and it will either depend directly on the item (variant (e)) or the user (variant (f)). The augmented pLSA model is hence no longer symmetric in the sense that both types of entities, users and items, are treated differently. We call the first variant the *community* version,

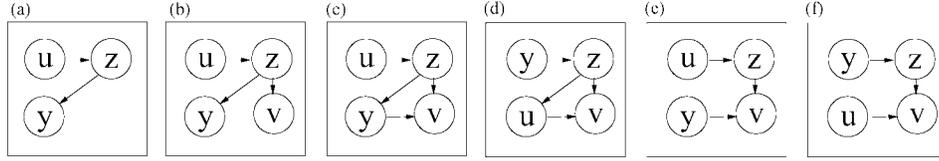


Fig. 1. Graphical model representation of possible extensions of the pLSA model to include a rating variable v . (a) Depicts the co-occurrence pLSA model. In (b), the rating only depends on the latent variable. (c) and (d) correspond to the free prediction mode with users and items interchanging their roles. (e) and (f) are derived from (c) and (d), respectively, by removing one arc, which is the manipulation corresponding to forced prediction.

since the user only influences the prediction mediated by z , but not directly. Correspondingly, the second variant will be called the *categorized* version, since items only impact the prediction through z which is supposed to model item categories or types. Similarly, two models can be derived for the free prediction mode. They are depicted in Figures 1(c) and 1(d). The model in Figure 1(b) is too restrictive to be useful for collaborative filtering.

4.2 Class Conditional Distributions

The proposed model has two ingredients, mixture coefficients—which in the *community* variant correspond to probabilities $P(z|u)$ —and class-conditional probability distributions $P(v|y, z)$. While the variables u and y are naturally assumed to be categorical, one of the key questions is how to take possible scales of the response variable v into account and how to parameterize the class-conditional distributions. In what follows, we will for concreteness focus on the community model, but the same argumentation applies to the categorized model variant.

If v is itself a categorical variable, for example, only taking binary values, $v \in \{-1, 1\}$, then one can simply introduce success probability parameters $\pi_{y,z} \in [0; 1]$ and define $P(v|y, z) \equiv \pi_{y,z}$. More generally, one can parameterize the conditional probability for categorical variables in the following manner (cf. Hofmann [2001b])

$$P(v|y, z) = \pi_{y,z}^v, \quad \text{where} \quad \sum_{v \in \mathcal{V}} \pi_{y,z}^v = 1. \quad (23)$$

In working with numerical (absolute) scales, we propose to introduce a location parameter $\mu_{y,z} \in \Re$ and a scale parameter $\sigma_{y,z} \in \Re^+$ for every community z and every item y which defines a Gaussian mixture model with user-specific mixing weights

$$P(v|u, y) = \sum_z P(z|u) P(v, \mu_{y,z}, \sigma_{y,z}), \quad (24a)$$

$$P(v; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(v - \mu)^2}{2\sigma^2}\right]. \quad (24b)$$

The assumption is that within each community the rating for each item possesses a typical value $\mu_{y,z}$, but that the observed ratings for individual users are noisy versions corrupted by normally distributed noise with variance $\sigma_{y,z}^2$.

Finally, notice that the expected response can be computed as

$$\begin{aligned} \mathbf{E}[v|u, y] &= \int_{\mathcal{V}} v P(v|u, y) dv = \sum_z P(z|u) \int_{\mathcal{V}} v P(v|y, z) dv \\ &= \sum_z P(z|u) \mu_{y,z}. \end{aligned} \quad (25)$$

It may be helpful to point out that Eq. (24) reduces to a standard Gaussian mixture model, in the degenerate case of a single user. In general, mixture proportions are user-specific though and the Gaussian pLSA model is very different from a standard Gaussian mixture model.

4.3 User Normalization

The models presented so far assume that all users express their ratings on a common scale. However, it is known that different users may associate subjectively different meanings with ratings and, for instance, a five-star rating may mean different things for different people. In memory-based methods, this is taken into account by similarity measures such as the Pearson or Spearman correlation coefficient [Herlocker et al. 1999]. One way to accommodate this in model-based approaches with numerical ratings is to normalize the raw user ratings appropriately. To that extend, we propose to transform ratings by (i) subtracting the user-specific mean rating μ_u and by (ii) normalizing the variance of ratings for each user to one. The first step accounts for individual differences in the overall “enthusiasm” of users and calibrates what should be considered as the neutral vote for every user. The second step makes the ratings more comparable across users by adjusting their dynamic range.

Formally, this is accomplished by performing the user-specific transformation of ratings

$$(u, v, y) \mapsto (u, v', y), \quad \text{with} \quad v' = \frac{v - \mu_u}{\sigma_u} \quad (26)$$

and where

$$\mu_u = \mathbf{E}[v|u], \quad \sigma_u^2 = \mathbf{E}[(v - \mu_u)^2 | u]. \quad (27)$$

For users with a small number of ratings, one has to be careful to perform appropriate smoothing in estimating the standard deviations (and to a lesser extend the means), since the empirical estimates derived from sample averages may be very unreliable due to sampling noise. We have thus used the following scheme to smooth the estimates of the variances,

$$\sigma_u^2 = \frac{\sum_{(u,v,y)} (v - \mu_u)^2 + q\bar{\sigma}^2}{n_u + q}, \quad (28)$$

where $\bar{\sigma}^2$ denotes the overall variance of ratings, n_u is the number of ratings available for user u and q is a free parameter controlling the smoothing strength (set to $q = 5$ in our experiments).

4.4 Maximum Likelihood Estimation: Discrete Case

Let us first discuss model fitting in the simplest case of the community or categorized models with a categorical response variable. Performing (approximate) maximum likelihood estimation can again be performed via the EM algorithm along exactly the same lines as outlined in Section 3. The only difference is that $P(y|z)$ is replaced by $P(v|y, z)P(y|z)$ (free prediction) or $P(v|y, z)$ (forced prediction), respectively. One thus arrives at the following E-step for the forced prediction case

$$Q^*(z; u, v, y, \hat{\theta}) = \frac{\hat{P}(z|u)\hat{P}(v|y, z)}{\sum_{z'} \hat{P}(z'|u)\hat{P}(v|y, z')} \quad (29)$$

and similarly for the free prediction case

$$Q^*(z; u, v, y, \hat{\theta}) = \frac{\hat{P}(z|u)\hat{P}(v|y, z)\hat{P}(y|z)}{\sum_{z'} \hat{P}(z'|u)\hat{P}(v|y, z')\hat{P}(z|z')} \quad (30)$$

The resulting M-step equations are Eq. (17) and

$$P(v|y, z) \propto \sum_{\substack{(u,v,y'): \\ v=v, y'=y}} Q^*(z; u, v, y, \hat{\theta}). \quad (31)$$

The details of the derivation can be found in the appendix. Comparing these equations with the standard pLSA equations in Section 3 shows little differences on a qualitative level.

4.5 Maximum Likelihood Estimation: Continuous Case

In the continuous case with Gaussian distributions, both the E-step and M-step need to be modified. The E-step equation can be obtained by replacing $P(v|y, z)$ with a Gaussian probability density function $P(v; \mu_{y,z}, \sigma_{y,z})$. The M-step update equations can be obtained by differentiating Eq. (16) with respect to the parameters $\mu_{y,z}$ and $\sigma_{\mu,z}^2$ (cf. appendix) which results in

$$\mu_{y,z} = \frac{\sum_{(u,v,y'): y'=y} v Q^*(z; u, v, y, \hat{\theta})}{\sum_{(u,v,y'): y'=y} Q^*(z; u, v, y, \hat{\theta})} \quad (32a)$$

$$\sigma_{y,z}^2 = \frac{\sum_{(u,v,y'): y'=y} (v - \mu_{y,z})^2 Q^*(z; u, v, y, \hat{\theta})}{\sum_{(u,v,y'): y'=y} Q^*(z; u, v, y, \hat{\theta})}. \quad (32b)$$

These are essentially the standard M-step equations of a Gaussian mixture model. The fact that mixing proportions are user-specific only enters in the computation of the posterior probabilities in the E-step. On an intuitive level, the community means $\mu_{y,z}$ and variances $\sigma_{y,z}^2$ are obtained by averaging over votes available for item y . The relative weight of the vote cast by user u though depends on the posterior probability of the latent “community” variable z .

4.6 Computational Complexity

The amount of data available in many practical applications of recommender systems can be enormous and the scalability of collaborative filtering

algorithms is a crucial factor for a successful system deployment. One has to distinguish between the offline and online computational complexity of an algorithm. The former accounts for computations that can be performed beforehand, that is, before actual predictions or recommendations for specific users have to be made. The latter deals with those computations that can only be performed in real-time during the interaction with a specific user, either because it is intractable to precompute all possible predictions or recommendations in advance, or because user profiles are changing dynamically in the course of an on-line session.

4.6.1 Offline Complexity. Analyzing the offline complexity of the proposed EM algorithm requires first of all to calculate the complexity of the E-step and M-step respectively. In the E-step, one needs to compute the optimal variational probability Q^* for each of the N observed user ratings. Each such Q^* consists of k numbers and requires a constant number of arithmetic operations to be computed, resulting in $\mathbf{O}(k \cdot N)$ operations for a single E-step. In the M-step, the posterior probabilities for each rating are accumulated to form the new estimates for $P(z|u)$, $P(v|y, z)$ and (in the free prediction model) $P(y|z)$. Notice that each $Q^*(z; u, v, y)$ is added to the accumulators for exactly one $P(z|u)$ and $P(y|z)$ as well as one of the accumulators for $P(v|y, z)$ (multinomial model) or $\mu_{y,z}$ and $\sigma_{y,z}^2$ (Gaussian model). Thus, the M-step also requires $\mathbf{O}(k \cdot N)$ operations. Typical values of k in our experiments have been in the range between 20 and 200. As far as memory requirements are concerned, we would like to point out that the E-steps and M-steps can be interleaved so that at any point we need to store the old value of the parameters, summarized in $\hat{\theta}$, as well as the same number of accumulator variables which are used internally to compute the new estimate of θ .

The number of EM-iterations that need to be performed cannot be easily estimated a priori, since it depends on properties of the specific data set. In the experiments, we have found that between 30–100 iterations are usually sufficient.

4.6.2 Online Complexity. More important for many applications is the online complexity of computing predictions in a dynamic environment. First of all, let us analyze the computational effort for computing a prediction $g(u, y)$, focusing on the Gaussian pLSA case for concreteness. From Eq. (25), we have that $g(u, y) = \sum_z P(z|u)\mu_{y,z}$. Since $\mu_{y,z}$ and $P(z|u)$ are assumed to be explicitly available as part of the statistical model, this requires $2k$ arithmetic operations. Moreover, since the number of communities k does not depend on the number of users n nor the number of items m , this amounts to a constant time prediction algorithm which has a computational complexity of $\mathbf{O}(k)$.

For new users u , we also have to compute $P(z|u)$ in the first place. Similarly, if additional ratings for user u become available, we would like to update the parameters $P(z|u)$ accordingly. We propose to ignore the effect on the community-specific parameters, since we expect the notion of a community to change on a much smaller time-scale. These changes can be taken into account by regular offline incremental EM updates or model retraining. From Eq. (17b), one sees

that computing $P(z|u)$ only involves posterior probabilities $P(v|u, y)$ for ratings of the same user u . Hence we propose to perform a limited EM iteration in which the E-step computes the posterior probabilities for all n_u ratings of the active user, which can be done in $\mathbf{O}(n_u \cdot k)$ operations and the M-step updates are restricted to the $P(z|u)$ parameters, which can also be carried out in time $\mathbf{O}(n_u \cdot k)$. This operation has also been called fold-in [Hofmann 2001a]. Typically, 20–40 restricted EM iterations are sufficient to compute $P(z|u)$. Notice that the computational complexity is independent of the number of users and items, but depends on the number of items that have been rated by the active user.

5. EXPERIMENTS

5.1 Data Set

The data we have used in our experiments is the EachMovie data set [EachMovie]. The data has been collected by Digital Equipment Research Center from 1995 through 1997. There are 1,623 items (movies) in this data set and 61,265 user profiles with a total of over 2.1 million ratings. Consequently, the average number of ratings per user is about 35. The rating scale is discrete, taking values from 0 (no star) to 5 (five stars),² with 5 being the highest rating and 0 being the lowest rating. The average rating over all observed votes is ≈ 3.03 and the overall rating variance is ≈ 1.48 .

The EachMovie data set is to our knowledge the largest publicly available data set for collaborative filtering and possesses the advantage of offering explicit user ratings. The latter fact allows us to study both, item selection and rating prediction.

5.2 Evaluation Metric

A thorough empirical analysis of collaborative filtering algorithms has been presented in Breese et al. [1998] and we have adapted most of the proposed evaluation metrics. The effectiveness of collaborative filtering techniques can be measured in various ways dependent on how the recommender system is used and how results are presented to the user.

The first setting we have investigated assumes that the goal of the system is to *predict* user ratings. Hence, we assume that an item y is presented to a user u and the goal is to predict the rating $\hat{v} = g(u, y)$. We have used two loss functions to measure the deviation between the predicted rating \hat{v} and the observed rating v : the absolute deviation $|\hat{v} - v|$ and the squared error $(\hat{v} - v)^2$. Empirical risks based on these loss functions are summarized as the mean absolute error (MAE) and the rooted mean square (RMS) error. In addition we have also measured the zero-one loss, in which case the predictions have been quantized by rounding \hat{v} to the closest integer.

In the second setting, the goal is to predict both, the selected item and the corresponding rating. Here we have used the score for ranked lists proposed

²The original ratings have been multiplied by a factor of 5.

in Breese et al. [1998]. Let us denote a permutation of the items by τ and the rank of an item y with respect to τ by $\tau(y)$. The top ranked item y will have $\tau(y) = 1$, the second item $\tau(y) = 2$, and so forth. Items whose ratings have been used for training are not included in the ranking. We then use the following rank score for τ ,

$$R(u, \tau) = \sum_{\langle u', v, y \rangle: u=u'} 2^{-\frac{\tau(y)-1}{\alpha-1}} \max(v - \bar{v}, 0), \quad (33)$$

with \bar{v} denoting the overall mean vote. The rationale behind this score is that when presented with a ranked list of items, users will sift through the list starting at the top, until they find a relevant item or simply give up. The probability that a user will ever take notice of an item at rank r is modeled as an exponential distribution with a half-life constant α (set to 4 in our experiments). The total score for a population of users is then measured by (cf. Breese et al. [1998])

$$R = 100 \frac{\sum_u R(u, \tau_u)}{\sum_u \max_{\tau'} R(u, \tau')}. \quad (34)$$

This normalizes the sum of the achieved score with what could have optimally achieved, if for every user all relevant items would appear at the very top of the ranked list.

5.3 Evaluation Protocols

We have used the leave-one-out protocol to evaluate the obtained prediction accuracies. This means we randomly leave out exactly one rating for every user possessing at least a minimal number $M \geq 2$ of observed ratings and then average the loss function over this set of users to obtain an estimate of the risk. This protocol has been called *AllBut1* in Breese et al. [1998]. More precisely, we have eliminated one vote for every user from the training set and trained models on this reduced set. Notice that this uses somewhat less data than required, but allows us to use a single model to evaluate the leave-one-out performance averaged over all users. We have varied M to investigate the prediction accuracy for users for which a minimal number of M ratings are available. In order to establish statistical significance of the findings, we have repeated the leave-one-out procedure 20 times with different random seeds. The reported numbers are the mean performance averaged over these 20 runs.

5.4 Results: Prediction Scenario

Table I summarizes experimental results obtained by different variants of the proposed method (multinomial, Gaussian, Gaussian with normalized votes), a memory-based method using the Pearson correlation coefficient, and results published in Breese et al. [1998] for various methods (Bayesian clustering = BC, Bayesian networks = BN, correlation = CR). The baseline is simply defined by the overall mean vote for each item. The test votes have been selected by leave-one-out with $M = 2$.

Table I. Prediction Accuracy of Various Methods in *forced* Prediction Mode Averaged Over 20 Runs

Method	Error			Relative improvement		
	MAE	RMS	0/1 loss	MAE	RMS	0/1 loss
Baseline	1.089	1.371	71.2	±0	±0	±0
Pearson correlation	0.948	1.237	64.7	12.9%	9.7%	9.1%
Multinomial	0.925	1.209	59.2	15.1%	11.8%	16.8%
Gaussian	0.974	1.251	67.2	10.5%	8.8%	5.6%
Gaussian, normalized	0.895	1.165	63.4	17.8%	15.0%	10.9%
CR [Breese et al. 1998]	0.994	—	—	—	—	—
BC [Breese et al. 1998]	1.103	—	—	—	—	—
BN [Breese et al. 1998]	1.066	—	—	—	—	—

As far as different sampling models for the ratings are concerned, one can make the following observations: First of all, the multinomial sampling model is quite competitive, yielding an improvement over the correlation-based method for all three loss functions and achieving the best overall performance for the zero-one loss. Second, the Gaussian model without user-specific normalization does much worse and is clearly not competitive. Third, performing the user-specific scale transformation in the Gaussian rating model leads to a substantial gain in prediction accuracy, yielding the best achieved results with respect to MAE and RMS error. It is also quite remarkable that this result is obtained with a model involving a much smaller number of communities ($k = 40$) compared to the multinomial model ($k = 200$). We conclude from this that the assumption of user-specific rating scales encodes useful prior knowledge.

As can be seen, the proposed Gaussian pLSA outperforms the memory-based method in terms of MAE and achieves a relative accuracy gain over the baseline of 17.8% as opposed to 12.9% for the Pearson correlation. This corresponds to a relative performance gain of approximately 6% when taking the Pearson correlation method as the baseline. In absolute terms, the MAE difference between the memory-based method based on the Pearson correlation coefficient and the normalized Gaussian pLSA model with $k = 40$ has a mean of 0.053 and a standard deviation of 0.0036. This is statistically highly significant, for example, using a paired t-test on the differences this corresponds to a t-value of ≈ 50 , which means that Gaussian pLSA outperforms the Pearson correlation method with confidence approaching certainty.

Notice that the results are overall better than the results published in Breese et al. [1998]. With respect to the latter results one has to acknowledge a somewhat different setup though, which has led to overall better performance results in our experiments. However, an approximate comparison seems to be possible by identifying our implementation of a correlation-based filtering method with the one implemented in Breese et al. [1998].

We have further investigated the effect of M on the prediction accuracy obtained by the correlation-based method and the Gaussian pLSA approach. It has to be expected that the prediction accuracy improves with growing M for all methods, since predictions should be more reliable for users for which a larger number of ratings is available. Figure 2 shows the result in terms of MAE for $M = 2, 5, 10, 20, 50, 100, 200$. It shows that the relative advantage of pLSA over

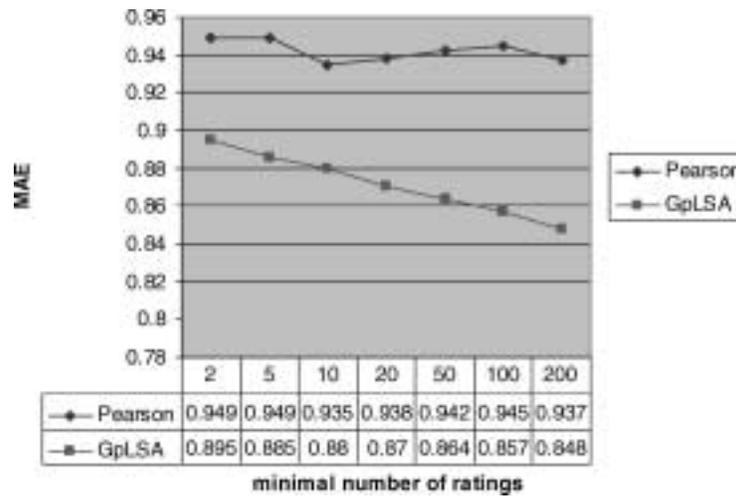


Fig. 2. MAE for the Pearson correlation method and Gaussian pLSA for different values of M (corresponding to the minimal number of ratings required for test users).

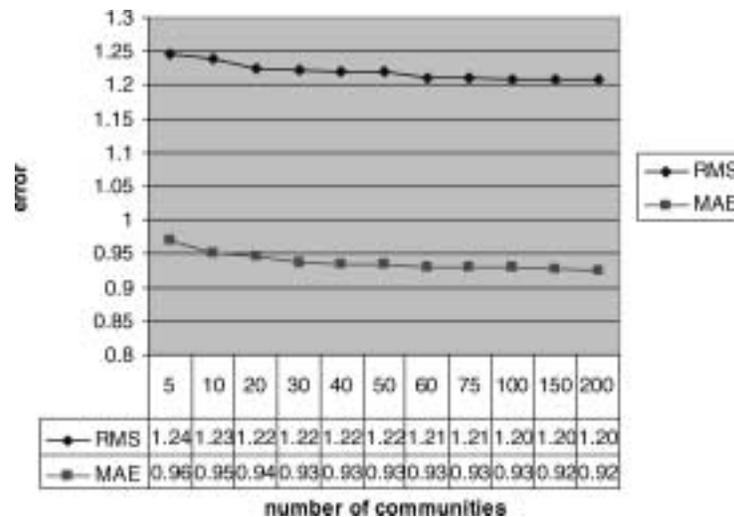


Fig. 3. Predictive performance in terms of MAE and RMS for the multinomial pLSA model as a function of the number of user communities k .

the correlation-based method increases for larger M . Gaussian pLSA seems to be capable of using additional ratings more effectively in order to improve the average prediction accuracy, whereas the correlation-based methods shows only a small improvement for larger M compared to the $M = 2$ case. At $M = 200$ the relative improvement of Gaussian pLSA over the correlation-based method is more than 10% and the relative improvement over the baseline popularity prediction approach is 23%.

We have also investigated the prediction accuracy obtained by models with varying number of user communities. Figure 3 shows the results obtained for

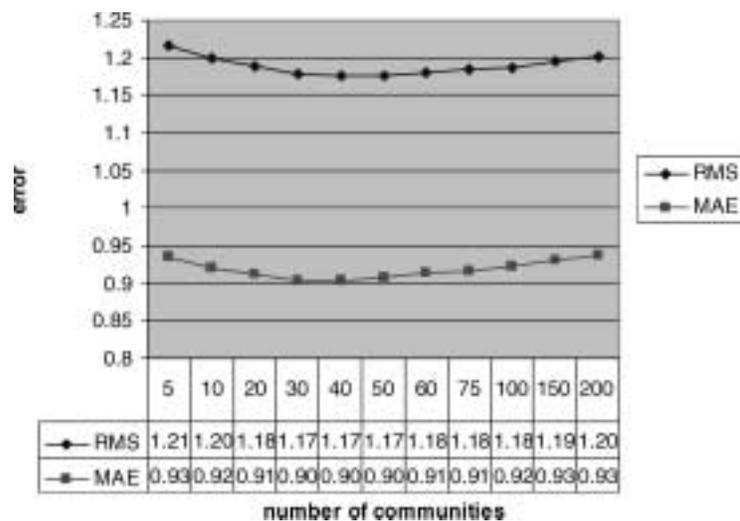


Fig. 4. Predictive performance in terms of MAE and RMS for the Gaussian pLSA model as a function of the number of user communities k .

the multinomial model. It can be seen that the performance improves steadily with the number of communities, but levels off towards the end. This is not true however in the case of models with Gaussian distributions for community ratings. Figure 4 shows a clear optimum around $k = 40$ communities, after which the performance slowly degrades with model size. This seems to indicate that the multinomial rating model needs to introduce a larger number of communities to account for the user-specific shift in the rating scale, which is incorporated a priori in the Gaussian model. The latter therefore requires fewer communities to model the correlations between the normalized user ratings.

Our next comparison (Figure 5) evaluates the effectiveness and importance of the tempered regularization with early stopping. We have used 10% hold-out data to determine the optimal stopping point and β -value, respectively. In the case of early stopping, one more EM training iterations using all data (training plus hold-out) is performed after stopping. Since the results are comparable for different sampling models, we only report and discuss the results for the multinomial case using the MAE criterion. As can be seen from the graph, the models obtained via tempered EM consistently outperform the ones trained by plain EM with early stopping. It is important to notice though that the number of EM iterations performed in early stopping EM is much smaller, typically between 30 and 40, compared to approximately 100–120 in the tempered version. On the other hand, for the same performance level, the tempered models need fewer parameters and are hence more compact. Tempering also requires to determine the optimal inverse temperature β which further increases the computational burden. In practice, the scalability-accuracy tradeoff may be decided with regard to the specific application and the available computational resources.

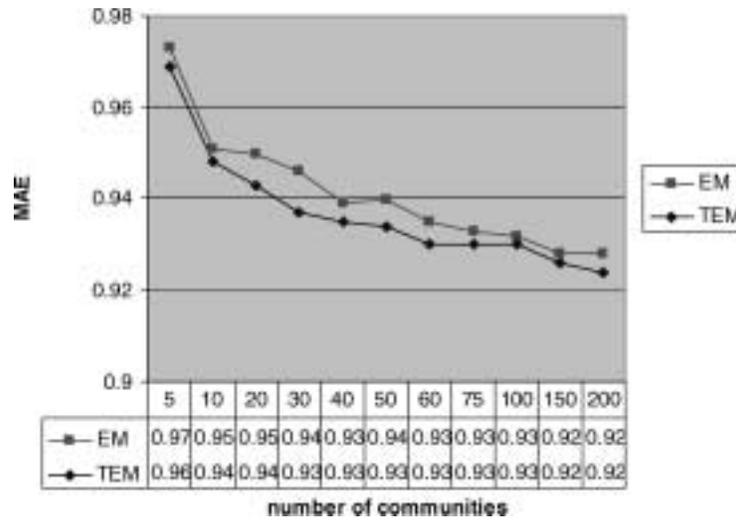


Fig. 5. Predictive performance in terms of MAE for tempered EM (TEM) vs. early stopping EM (EM).

Table II. Performance of Different Methods in Free Prediction Mode According to Ranking Criterion

Method	rank gain	rel. improv.	abs	rms
Baseline	16.76	± 0	1.091	1.371
Pearson correlation	21.14	26.1	0.951	1.231
Multinomial	24.41	45.6	0.981	1.245
Gaussian	21.80	30.0	1.020	1.290
Gaussian, normalized	24.28	44.8	0.955	1.211

5.5 Results: Ranking Scenario

The second scenario we have experimentally investigated is the *free* prediction mode. Since the prediction accuracy in predicting votes is no longer adequate, we have used the ranking loss in Eq. (34) to benchmark different algorithms. Again, we have used the leave-one-out protocol. The results are summarized in Table II.

The best ranking scores are obtained by the multinomial and the normalized Gaussian pLSA model. The difference between these two models is not statistically significant, while the performance gain relative to the Pearson correlation method is significant. The relative performance gain with respect to the popularity baseline is overall higher than in the forced prediction mode—more than 40% relative improvement are achieved. Notice however that the actual prediction error of these models is higher than for the models that have been trained in forced mode. In fact the absolute error of the Gaussian pLSA model is slightly higher than with the correlation-based approach, although the difference is not statistically significant.

5.6 Runtime

We have implemented the Gaussian and multinomial pLSA algorithms in C++ and ran our experiments on a standard PC with a 2GHZ CPU. The computer time needed to perform a single EM step using all 61,265 users for a Gaussian model with $k = 40$ is about 30 seconds. The number of iterations required when using early stopping is around 30–40 while up to 100–120 iterations are required when using tempered EM. In the latter case, the off-line training of the model takes thus slightly less than 1 hour, while off-line training using early-stopping takes less than 20 minutes. For models with larger k , the training time grows proportionally in k .

5.7 Miscellaneous

In the above experiments, we have always used the community variant of the latent class models, that is, models involving $P(v|y, z)$ instead of $P(v|u, z)$. We have also run experiments with the latter though, which has consistently led to worse results. For example, the best MAE obtained in forced prediction was 0.971 compared to an absolute error of 0.927 for the multinomial community model.

We have also not been able to obtain competitive results using the normalized risk function of Eq. (8) for training. In fact, we have even experimented with various interpolated versions of the risk functions in Eq. (7) and Eq. (8) without much success. It thus seems that a uniform weighting of each rating is the best choice.

5.8 Mining User Communities

Finally, we would like to illustrate that the decomposition of user ratings may lead to the discovery of interesting patterns and regularities that describe user interests as well as disinterest. To that extent, we have to find a mapping of a quantitative pLSA model into a more qualitative description suitable for visualization. We propose to summarize and visualize each user community, corresponding to one of the k possible states of the latent variable Z , in the following way. We sort items within each community or interest group according to their popularity within the community as measured by the probability $P(y|z)$. The most popular items are used to characterize a community and we expect these items to be descriptive for the types of items that are relevant to the user community. Figures 6 and 7 display the interest groups extracted by a multinomial pLSA model with $k = 40$, ordered according to the average “positiveness” of each group, computed as $g(y, z) = \sum_{y,v} vP(v|y, z)P(y|z)$. For example, in Figure 6, interest group 6 has romantic movies like “The Remains of the Day”, “The Piano”, “Like Water for Chocolate” and “Much Ado About Nothing” top ranked. Interest group 18 seems to be formed around musicals like “Mary Poppins”, “Cinderella”, “The Sound of Music” and “Dumbo”. With each top-ranked item, we also suggest to display the average rating the item receives in the community, computed as $\bar{v} = \sum_v vP(v|y, z)$. These numbers

Interest Group 1, *4.8*	Interest Group 2, *4.6*	Interest Group 3, *4.5*	Interest Group 4, *4.4*	Interest Group 5, *4.8*
<i>Twister</i> [*4.6*] [0.064]	<i>Batman (1989)</i> [*4.1*] [0.066]	<i>Twinspotting</i> [*5*] [0.038]	<i>Dead Man Walking</i> [*5*] [0.052]	<i>The Santa Clause</i> [*4.5*] [0.014]
<i>Independence Day</i> [*4.9*] [0.061]	<i>Apollo 13</i> [*5*] [0.065]	<i>Fargo</i> [*5*] [0.033]	<i>The Truth about Ca...</i> [*4.3*] [0.039]	<i>Casper</i> [*4.5*] [0.014]
<i>Toy Story</i> [*4.9*] [0.057]	<i>True Lies</i> [*4.7*] [0.039]	<i>Pulp Fiction</i> [*5*] [0.028]	<i>Get Shorty</i> [*4.6*] [0.036]	<i>Robin Hood: Men in...</i> [*4.3*] [0.013]
<i>Broken Arrow</i> [*4.4*] [0.054]	<i>Batman Forever</i> [*4.1*] [0.054]	<i>Clerks</i> [*4.7*] [0.023]	<i>Series and Sensible...</i> [*5*] [0.035]	<i>Tommy Boy</i> [*4.5*] [0.013]
Interest Group 6, *4.3*	Interest Group 7, *4.3*	Interest Group 8, *4.2*	Interest Group 9, *4*	Interest Group 10, *3.9*
<i>The Remains of the...</i> [*4.5*] [0.047]	<i>The Empire Strikes...</i> [*4.7*] [0.032]	<i>Pretty Women</i> [*4.3*] [0.039]	<i>Slipstix</i> [*4.2*] [0.015]	<i>A Clockwork Orange...</i> [*4.2*] [0.01]
<i>The Piano</i> [*4.7*] [0.043]	<i>Raiders of the Los...</i> [*4.7*] [0.03]	<i>Mr. Doubtfire</i> [*4.3*] [0.039]	<i>Larry Maguire</i> [*4.6*] [0.013]	<i>Amadeus (1984)</i> [*4.2*] [0.0086]
<i>Like Water For Cha...</i> [*4.7*] [0.043]	<i>Star Wars</i> [*4.9*] [0.026]	<i>Ghost</i> [*4.4*] [0.057]	<i>The First Wives Cl...</i> [*3.8*] [0.013]	<i>Psycho (1960)</i> [*4.3*] [0.0086]
<i>Much Ado About Not...</i> [*4.6*] [0.041]	<i>Indiana Jones and...</i> [*4.5*] [0.025]	<i>Slipstix in South...</i> [*4.4*] [0.055]	<i>William Shakespeare...</i> [*4.5*] [0.011]	<i>One Flew Over the...</i> [*4.5*] [0.0095]
Interest Group 11, *3.8*	Interest Group 12, *3.8*	Interest Group 13, *3.7*	Interest Group 14, *3.7*	Interest Group 15, *3.7*
<i>Independence Day (...)</i> [*4.1*] [0.05]	<i>The Professional</i> [*4.3*] [0.034]	<i>Boys on the Side</i> [*3.8*] [0.022]	<i>The Usual Suspects</i> [*4.4*] [0.063]	<i>Searching For Robb...</i> [*4.2*] [0.031]
<i>Twister</i> [*3.8*] [0.043]	<i>The Cider</i> [*4.1*] [0.03]	<i>Only You</i> [*3.5*] [0.021]	<i>Quint Shoer</i> [*3.9*] [0.052]	<i>Snow White and the...</i> [*3.9*] [0.03]
<i>Mission: Impossible...</i> [*3.7*] [0.041]	<i>Demolition Man</i> [*3.4*] [0.028]	<i>Something to Talk...</i> [*3.3*] [0.02]	<i>Get Shorty</i> [*3.8*] [0.052]	<i>Pinocchio (1940)</i> [*3.9*] [0.025]
<i>Toy Story</i> [*4.1*] [0.038]	<i>Species</i> [*3.1*] [0.028]	<i>Comma, Comma</i> [*3.4*] [0.018]	<i>Pulp Fiction</i> [*4.6*] [0.044]	<i>Selena</i> [*3.7*] [0.023]
Interest Group 16, *3.6*	Interest Group 17, *3.6*	Interest Group 18, *3.5*	Interest Group 19, *3.5*	Interest Group 20, *3.5*
<i>Outbreak</i> [*3.8*] [0.041]	<i>Three Colors: Red</i> [*4.2*] [0.033]	<i>Mary Poppins (1964...</i> [*4.1*] [0.028]	<i>Leaving Las Vegas</i> [*3.8*] [0.0095]	<i>Casper</i> [*3.5*] [0.025]
<i>Waterworld</i> [*3.1*] [0.04]	<i>Three Colors: Blue</i> [*4*] [0.032]	<i>Cinderella (1950)</i> [*4*] [0.024]	<i>Dead Man Walking</i> [*4.1*] [0.078]	<i>First Knight</i> [*4*] [0.023]
<i>Broncheart</i> [*4.9*] [0.038]	<i>Three Colors: Whit...</i> [*3.9*] [0.028]	<i>The Sound of Music...</i> [*4.3*] [0.022]	<i>The Birdcage</i> [*3.3*] [0.076]	<i>Junior</i> [*3.5*] [0.021]
<i>The Net</i> [*3.4*] [0.038]	<i>Barcelona</i> [*3.4*] [0.02]	<i>Quinto (1961)</i> [*3.9*] [0.022]	<i>Series and Sensible...</i> [*3.9*] [0.074]	<i>The Santa Clause</i> [*3.5*] [0.019]

Fig. 6. User communities 1–20 (of 40) extracted from the EachMovie data set with a multinomial pLSA model.

are displayed in rectangular brackets and enclosed by stars in Figures 6 and 7. Looking at the average ratings obtained by the top movies in each of the interest groups extracted from the EachMovie data set, it is interesting to see that communities seem to constitute themselves around items of either common interest or disinterest. This is indicated by the fact that movies with highest selection probabilities $P(y|z)$ within a community z seem to have similar ratings. Notice that the latter fact is nowhere enforced by any means in the model and is a property that emerges from the data.

While some of the dis-interest groups like the one formed around the movies “Mighty Morphin Power Rangers” and “The Brady Bunch Movie” are certainly not useful to derive recommendations, they are however important to model and predict negative ratings and to prevent that certain items end up as recommendations where they should not. Some of the extracted “communities” may thus not correspond to interest groups in the usual sense, which are formed around a common interest. Rather, communities are characterized by a commonality that can also be a shared depreciation for certain items.

Interest Group 21, *3.5*	Interest Group 22, *3.4*	Interest Group 23, *3.1*	Interest Group 24, *3.1*	Interest Group 25, *3*
Snake [*3.7*] [0.004]	Money Train [*3.2*] [0.017]	Jumanji [*3.5*] [0.006]	Jurassic Park [*3.2*] [0.08]	Larry Maguire [*3.5*] [0.021]
Mighty Aphrodite [*3.3*] [0.018]	Terminal Velocity [*3.2*] [0.016]	Over and Company... [*3.1*] [0.023]	Forest Gump [*3.5*] [0.079]	The People vs. Larr... [*3.5*] [0.016]
Fargo [*4.1*] [0.018]	The Shadow [*3.3*] [0.015]	Muppet Treasure Is... [*3.1*] [0.022]	The Fugitive [*3.6*] [0.066]	Marx Attacks! [*2.6*] [0.015]
Four Weddings and... [*3.3*] [0.018]	Hard Target [*3.4*] [0.015]	Prozac [*3.5*] [0.021]	Terminator 2: Judg... [*3.6*] [0.056]	The Last World Ju... [*2.8*] [0.015]
Interest Group 26, *2.8*	Interest Group 27, *2.7*	Interest Group 28, *2.7*	Interest Group 29, *2.6*	Interest Group 30, *2.2*
Dances With Wolves [*3.4*] [0.11]	Naked Gun 33 1/3... [*2.4*] [0.02]	L.O. [*2.8*] [0.022]	Mrs. Doubtfire [*3*] [0.040]	Midland Falls [*2.1*] [0.018]
Estimote (1989) [*2.4*] [0.11]	The Hudsoner Proa... [*3.3*] [0.019]	French Kiss [*2.9*] [0.019]	Fredly Woman [*2.9*] [0.047]	The Arrival (Shock... [*2.5*] [0.017]
Pop Fiction [*3.4*] [0.1]	Hot Shots! Part De... [*2.3*] [0.019]	Nine Months [*2.5*] [0.019]	Ghost [*2.9*] [0.042]	Primal Fear [*2.9*] [0.017]
Apollo 13 [*3.6*] [0.004]	So I Married an Ac... [*2.6*] [0.016]	Junior [*2.4*] [0.018]	The Mask [*2.5*] [0.042]	City Hall [*2.4*] [0.016]
Interest Group 31, *2.2*	Interest Group 32, *2*	Interest Group 33, *1.8*	Interest Group 34, *1.8*	Interest Group 35, *1.7*
E.T. The Extrater... [*2.6*] [0.01]	Lord of Musions [*1.6*] [0.011]	Sleepless in Seat... [*1.8*] [0.017]	Toy Story [*2.4*] [0.05]	Strip tease [*0.025*] [0.033]
The Sound of Music... [*2.3*] [0.0066]	Tales From the Hoo... [*1.6*] [0.0037]	The Firm [*1.8*] [0.015]	Mission: Impossib... [*1.8*] [0.040]	Independence Day (... [*0.87*] [0.029]
Top Gun (1986) [*2.3*] [0.0066]	Mahrta [*2.4*] [0.0033]	Freddy Woman [*1.5*] [0.015]	Independence Day (... [*2.1*] [0.048]	The Cable Guy [*0.16*] [0.028]
Mary Poppins (1964... [*2.3*] [0.0033]	[*2*] [0.0032]	Days [*2*] [0.015]	Twister [*1.8*] [0.043]	Barb Wire [*4.9+0.015*] [0.025]
Interest Group 36, *1.1*	Interest Group 37, *0.60*	Interest Group 38, *0.39*	Interest Group 39, *0.16*	Interest Group 40, *0.16*
Super Mario Bros... [*0.11*] [0.017]	Mighty Morphin Pow... [*0.017*] [0.033]	Dumb and Dumber [*0.0025*] [0.038]	Korasson [*0.028*] [0.014]	Tales From the Hoo... [*0.022*] [0.0075]
The Beverly Hills... [*0.34*] [0.016]	The Brady Bunch Ma... [*0.28*] [0.024]	Ace Ventura, Pet D... [*0.016*] [0.034]	Children of the Co... [*0.021*] [0.014]	Yankee in Brooklyn... [*1.3+0.05*] [0.007]
Richie Rich [*0.22*] [0.015]	Mortal Kombat [*0.21*] [0.018]	Ace Ventura, When... [*0.00067*] [0.033]	A Very Brady Seque... [*0.083*] [0.012]	The Baby-Sitters C... [*0.0063*] [0.007]
The Next Karate Ki... [*0.21*] [0.014]	The Bridges of Mad... [*0.015*] [0.018]	Waterworld [*0.034*] [0.028]	Halloween: The Cur... [*0.035*] [0.012]	Caddyman, Parents! [*0.0039*] [0.0065]

Fig. 7. User communities 21–40 (of 40) extracted from the EachMovie data set with a multinomial pLSA model.

Overall, we believe that patterns and regularities extracted with pLSA models can be helpful in understanding shared interests of users and correlations among ratings for different items. The ability to automatically discover communities as part of the collaborative filtering process is a trait which pLSA shares with only few other methods such as clustering approaches, but which is absent in all memory-based techniques.

6. CONCLUSION

We have presented a powerful method for collaborative filtering and mining of user data based on a statistical latent class model. The method achieves competitive recommendation and prediction accuracies, is highly scalable, and extremely flexible. Conceptionally, the decomposition of user preferences using overlapping user communities is a novel idea that clearly distinguishes this approach from traditional memory-based approaches as well as previous model-based methods.

APPENDIX

A.1 Derivation of the Generalized E-step

We derive the solution of minimizing the following objective function

$$\mathcal{F}_\beta(Q) = \bar{\mathcal{R}}(\theta, Q) - \frac{1}{\beta} \sum_{(u,v,y)} H(Q(\cdot; u, v, y)) \quad (35)$$

with respect to the variational distribution Q .

Notice first that \mathcal{F}_β can be rewritten as a sum over contributions from all (u, v, y) pairs,

$$\mathcal{F}_\beta(Q) = \sum_{(u,v,y)} \mathcal{F}_\beta(u, v, y, Q), \quad (36)$$

where

$$\mathcal{F}_\beta(u, v, y, Q) = - \sum_z Q(z; u, v, y) [\log S(u, v, y, z) - \log Q(z; u, v, y)]. \quad (37)$$

Here $S(u, v, y, z) = P(y|z)P(z|u)$ in the co-occurrence model, whereas $S(u, v, y, z) = P(v|y, z)P(y|z)P(z|u)$ in the free prediction case and $S(u, v, y, z) = P(v|y, z)P(z|u)$ for forced prediction.

Hence, one can minimize every $\mathcal{F}_\beta(u, v, y, Q)$ separately. Introducing a Lagrange multiplier λ to enforce the normalization constraint $\sum_z Q(z; u, v, y) = 1$, one forms the Lagrangian function

$$\mathcal{L}_\beta(Q, \lambda) = \mathcal{F}_\beta(u, v, y, Q) + \lambda \left(\sum_z Q(z; u, v, y) - 1 \right). \quad (38)$$

Computing the partial derivative of \mathcal{L}_β with respect to $Q(z; u, v, y)$ and setting to zero results in the necessary conditions

$$\frac{1}{\beta} \log Q^*(z; u, v, y) = \log S(u, v, y, z) - \lambda + \frac{1}{\beta}. \quad (39)$$

Exponentiating both sides of the equality yields

$$Q^*(z; u, v, y) = \frac{S(u, v, y, z)^\beta}{\exp[\beta\lambda - 1]}. \quad (40)$$

Apparently, λ needs to be chosen such that

$$\exp[\beta\lambda - 1] = \sum_{z'} S(u, v, y, z')^\beta \iff \lambda = \frac{1}{\beta} \left[1 + \log \sum_{z'} S(u, v, y, z')^\beta \right]. \quad (41)$$

Plugging this value for the Lagrange multiplier λ back into the Lagrangian function results in the general optimality condition for Q which leads to the special cases of Eq. (15) and the tempered version in Eq. (19) for the co-occurrence model and Eq. (29) and Eq. (30) for the model including rating variables.

A.2 Derivation of the M-step

The most general case is the free prediction case, where one has to minimize

$$\bar{\mathcal{R}}(\theta, \hat{\theta}) = -\frac{1}{N} \sum_{(u,v,y)} \sum_z Q^*(z; u, v, y, \hat{\theta}) [\log P(v|y, z) + \log P(y|z) + \log P(z|u)] \quad (42)$$

with respect to the parameters $P(y|z)$, $P(z|u)$ and the parameters representing $P(v|y, z)$, respectively. In order to ensure the normalization $\sum_y P(y|z) = 1$ for all z and $\sum_z P(z|u) = 1$ for all u , we introduce Lagrange multipliers λ_z and λ_u and form the Lagrangian

$$\mathcal{L}(\theta) = \bar{\mathcal{R}}(\theta, \hat{\theta}) + \sum_z \lambda_z \left(\sum_y P(y|z) - 1 \right) + \sum_u \lambda_u \left(\sum_z P(z|u) - 1 \right) \quad (43)$$

Taking derivatives with respect to $P(y|z)$ and setting to zero results in

$$\frac{1}{P(y|z)} \sum_u Q^*(z; u, v, y, \hat{\theta}) - \lambda_z = 0 \iff P(y|z) = \frac{1}{\lambda_z} \sum_u Q^*(z; u, v, y, \hat{\theta}) \quad (44)$$

Similarly one obtains for $P(z|u)$

$$\frac{1}{P(z|u)} \sum_y Q^*(z; u, v, y, \hat{\theta}) - \lambda_u = 0 \iff P(z|u) = \frac{1}{\lambda_u} \sum_y Q^*(z; u, v, y, \hat{\theta}) \quad (45)$$

Plugging these results back into Eq. (43) yields the following expressions for the Lagrange multipliers

$$\lambda_z = \sum_y \sum_u Q^*(z; u, v, y, \hat{\theta}) \quad (46a)$$

$$\lambda_u = \sum_z \sum_y Q^*(z; u, v, y, \hat{\theta}) = \sum_y 1 = |\mathcal{Y}| \quad (46b)$$

which in turn lead to the M-step equations in Eq. (17).

In the multinomial pLSA model, one uses the same approach for $P(v|y, z)$, introducing additional Lagrange multipliers $\lambda_{y,z}$ to ensure that $\sum_v P(v|y, z) = 1$. Augmenting the Lagrangian function by a term $\sum_{y,z} \lambda_{y,z} [\sum_v P(v|y, z) - 1]$ and solving as before leads to equation Eq. (31). Notice that the M-step equations for the co-occurrence model and the forced prediction case can be obtained by dropping the equations for $P(v|y, z)$ and $P(y|z)$, respectively.

In the Gaussian pLSA model, one has to compute derivatives

$$\frac{\partial \bar{\mathcal{R}}(\theta, \hat{\theta})}{\partial \mu_{y,z}} = -\frac{1}{N} \sum_{(u,v,y'), y=y'} Q^*(z; u, v, y, \hat{\theta}) \frac{v - \mu_{y,z}}{\sigma_{y,z}^2} = 0 \iff \quad (47)$$

$$\mu_{y,z} = \frac{\sum_{(u,v,y'), y=y'} Q^*(z; u, v, y, \hat{\theta}) v}{\sum_{(u,v,y'), y=y'} Q^*(z; u, v, y, \hat{\theta})} \quad (48)$$

and a similar equation for $\sigma_{y,z}^2$.

ACKNOWLEDGMENTS

The EachMovie data set is by courtesy of Digital Equipment Corporation and was generously provided by Paul McJones.

REFERENCES

- BASU, C., HIRSH, H., AND COHEN, W. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Recommender System Workshop*. 11–15.
- BLEI, D. M., NG, A. Y., AND JORDAN, M. I. 2002. Latent dirichlet allocation. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, Mass.
- BREESE, J. S., HECKERMAN, D., AND KARDIE, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. 43–52.
- CANNY, J. 2002. Collaborative filtering with privacy. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Los Alamitos, Calif., pp. 45–57.
- CHIEN, Y.-H. AND GEORGE, E. 1999. A Bayesian model for collaborative filtering. In *Online Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*.
- COVER, T. M. AND THOMAS, J. A. 1991. *Information Theory*. Wiley, New York.
- DEERWESTER, S. C., DUMAIS, S. T., LANDAUER, T. K., FURNAS, G. W., AND HARSHMAN, R. A. 1990. Indexing by latent semantic analysis. *J. ASIS* 41, 6, 391–407.
- DEMPSTER, A., LAIRD, N., AND RUBIN, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. B* 39, 1–38.
- EACHMOVIE. www.research.digital.com/src/eachmovie/.
- FREUND, Y., IYER, R., SCHAPIRE, R. E., AND SINGER, Y. 1998. An efficient boosting algorithm for combining preferences. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*. J. W. Shavlik, Ed. Morgan-Kaufmann, San Francisco, Calif., 170–178.
- GOLDBERG, D., NICHOLS, D., OKI, B., AND TERRY, D. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12, 61–70.
- GOLDBERG, K., ROEDER, T., GUPTA, D., AND PERKINS, C. 2001. Eigentaste: A constant-time collaborative filtering algorithm. *Inf. Retr.* 4, 2, 133–151.
- HECKERMAN, D., CHICKERING, D. M., MEEK, C., ROUNTHWAITE, R., AND KADIE, C. M. 2000. Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.* 1, 49–75.
- HERLOCKER, J., KONSTAN, J., BORCHERS, A., AND RIEDL, J. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd ACM-SIGIR International Conference on Research and Development in Information Retrieval* (Berkeley, Calif.). ACM, New York.
- HOFMANN, T. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd ACM-SIGIR International Conference on Research and Development in Information Retrieval* (Berkeley, Calif.), ACM, New York, 50–57.
- HOFMANN, T. 2001a. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn. J.* 42, 1, 177–196.
- HOFMANN, T. 2001b. What people (don't) want. In *Proceedings of the European Conference on Machine Learning (ECML)*.
- HOFMANN, T. AND PUZICHA, J. 1999. Latent class models for collaborative filtering. In *Proceedings of the International Joint Conference in Artificial Intelligence*.
- KONSTAN, J., MILLER, B., MALIZ, D., HERLOCKER, J., GORDON, L., AND RIEDL, J. 1997. GroupLens: Applying collaborative filtering to usenet news. *Commun. ACM* 40, 3, 77–87.
- MINKA, T. AND LAFFERTY, J. 2002. Expectation-propagation for the generative aspect model. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*.
- NEAL, R. AND HINTON, G. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, M. I. Jordan, Ed. Kluwer.
- PEREIRA, F. C. N., TISHBY, N., AND LEE, L. 1993. Distributional clustering of English words. In *Meeting of the Association for Computational Linguistics*. 183–190.

- RESNIK, P., IACOVOU, N., SUCHAK, M., BERGSTROM, P., AND RIEDL, J. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM, Conference on Computer Supported Cooperative Work*. 175–186.
- SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., AND RIEDL, J. 2000. Application of dimensionality reduction in recommender system—A case study. In *Proceedings of the ACM WebKDD 2000 Web Mining for E-Commerce Workshop*. ACM, New York.
- SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., AND RIEDL, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the World Wide Web Conference (WWW10)*. 285–295.
- SHARDANAND, U. AND MAES, P. 1995. Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*. Vol. 1. ACM, New York, 210–217.
- UNGAR, L. AND FOSTER, D. 1998. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, Menlo Park, Calif.

Received January 2003; revised July 2003; accepted September 2003