Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis

Thomas Hofmann
Department of Computer Science
Brown University, Providence, RI, USA
th@cs.brown.edu

ABSTRACT

Collaborative filtering aims at learning predictive models of user preferences, interests or behavior from community data, i.e. a database of available user preferences. In this paper, we describe a new model-based algorithm designed for this task, which is based on a generalization of probabilistic latent semantic analysis to continuous-valued response variables. More specifically, we assume that the observed user ratings can be modeled as a mixture of user communities or interest groups, where users may participate probabilistically in one or more groups. Each community is characterized by a Gaussian distribution on the normalized ratings for each item. The normalization of ratings is performed in a user-specific manner to account for variations in absolute shift and variance of ratings. Experiments on the EachMovie data set show that the proposed approach compares favorably with other collaborative filtering techniques.

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

Algorithms, Experimentation, Theory

Keywords

collaborative filtering, recommender systems, machine learning, data mining

1. INTRODUCTION

Content-based filtering and retrieval build 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 simply be unaware or at least inattentive of their interests. In both cases one would like to predict user preferences and recommend items of interest 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, interests or behavior from community data, i.e. a database of available user preferences. Up to now, the dominant paradigm for performing collaborative filtering in recommender systems has been based on nearest neighbor regression or so-called memory-based techniques. These systems use a general two-step approach. First users are identified that are similar to some active user for which a recommendation has to be made. Then predictions and recommendations are computed based on the preferences and judgments of these similar or like-minded users. Recommender systems using memory-based technology include [1], the GroupLens (and MovieLens) project [2, 3], Ringo [4] 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 sufficiently accurate for many real-world problems. Yet there are also a number of severe shortcomings, four of which we would like to point out here: (i) The accuracy obtained by memory-based methods may be suboptimal. Indeed our experiments indicate that significant accuracy gains are achievable. (ii) Since no explicit statistical model is constructed, nothing is really learned form the available user profiles and very little general insight is gained. Hence, memory-based methods are only of very limited use, for example, as data mining tools. (iii) Memory-based methods do not scale well in terms of their resource requirements (memory and computing 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. (v) Actual user profiles have to be kept for prediction, potentially raising privacy issues.

This paper deals with a model-based approach that addresses the above shortcomings and (i) achieves higher prediction accuracies, (ii) compresses the data into a compact

SIGIR'03 July 28–August 1, 2003, Toronto, Canada.

model that automatically identifies user communities and interest groups, (iii) enables computing preference predictions in constant time, (iv) gives the system designer more flexibility in specifying the objectives of the application and (v) does not require to keep the original user profiles.

Model-based techniques have been investigated before, most notably Bayesian and non-Bayesian clustering techniques [5, 6, 7, 8], Bayesian networks [5], and dependency networks [9]. The approach proposed in this paper is a generalization of a statistical technique called probabilistic Latent Semantic Analysis (pLSA) [10] which was originally developed in the context of information retrieval [11]. It bears some similarity with clustering methods in that latent variables for user communities are introduced, yet the communities can be overlapping and users are not partitioned (not even probabilistically). In fact, the probabilistic latent semantic models are in many ways more closely related to dimension reduction methods and matrix decomposition techniques such as Singular Value Decomposition, which have also been applied in the context of recommender systems [12, 13]. The main difference with respect to Bayesian networks or dependency networks is the fact that the latter learn dependency structures 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 advantage over PCA and SVD-based dimension reduction methods is that our approach offers a probabilistic semantics and can build on statistical techniques for inference and model selection.

2. MODEL-BASED COLLABORATIVE FILTERING

2.1 Implicit and Explicit Ratings

The domains we consider consist of a set of persons or *users* $\mathcal{U} = \{u_1, \ldots, u_n\}$, a set of *items* $\mathcal{Y} = \{y_1, \ldots, y_m\}$ 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. In many cases however, users may also provide an *explicit* rating $v \in \mathcal{V}$ as part of an observation. The main focus of this paper is on numerical ratings, which are commonly quantized to a small number of response levels. For example, a five star rating scale is commonly used in movie recommendation systems such as EachMovie or MovieLens.

2.2 Prediction Problems

We will consider two type of prediction problems. The first setting which we call *forced prediction* involves predicting a preference value for a particular item given the identity of the user, i.e. one would like to learn a mapping $g : \mathcal{U} \times \mathcal{Y} \to \mathcal{V}$. More generally, one may be interested in the conditional probability distribution $p(v|u, y)$ that user u will rate item y with v. Based on the latter one may also define a deterministic prediction function by $g(u, y) = \int_{\mathcal{V}} v\, p(v|u, y)\, dv$, where $p(v|u, y)$ denotes a conditional probability density function.

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 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.

In the forced prediction case, the user is presented with a particular item and provides an *explicit* rating for it. Here the selection of the item on which a user vote or response is solicited becomes part of the experimental design. A single item is presented to the user at a time and the user's response is recorded. This is in contrast to the free prediction case where 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 $\mathcal{H}$. We propose to utilize the (negative) logarithmic loss function which forms the basis of the popular maximum likelihood estimation approach in statistics,

$$
\begin{aligned}
L((u,v,y),\theta) &= -\log p(v|u,y;\theta), \quad \text{or} &(1)\\
L((u,v,y),\theta) &= -\log p(v,y|u;\theta). &(2)
\end{aligned}
$$

The logarithmic loss can be used to define an empirical risk function, which is the negative conditional log-likelihood. Maximum likelihood estimation hence amounts to minimizing

$$\mathcal{R}(\theta) = \frac{1}{N} \sum_{\langle u,v,y \rangle} L((u,v,y),\theta), \qquad (3)$$

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.

3. GAUSSIAN PROBABILISTIC LATENT SEMANTIC MODEL

3.1 Co-occurrence Model

We would like to discuss a simple model for co-occurrence data first, which is known as probabilistic latent semantic analysis (pLSA) [10, 11]. This can be thought of as a special case of collaborative filtering with implicit preference data. 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

$$P(u, y; \theta) = \sum_z P(u, y, z) = \sum_z P(y|z)P(z|u)P(u), \qquad (4)$$

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 $P(u,y;\theta'') = \sum_z P(u|z)P(z|y)$. Since the more typical situation in collaborative filtering is to make personalized recommendations, we will mainly work with the conditional model

$$P(y|u;\theta) = \sum_z P(y|z)P(z|u)\,. \tag{5}$$

Here the parameter vector θ summarizes the probabilities $P(z|u)$ which can be described by $n \times (k-1)$ independent parameters as well as $P(y|z)$ which requires $(m-1) \times k$ independent parameters.

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 distributions over user-item pairs becomes less and less constrained until a saturated model is obtained, which can represent any joint probability 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 can be used for that purpose.

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, i.e. 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 grouping users into user communities and items into groups of related items.

3.2 Models with Numerical Ratings

Since many applications of collaborative filtering involve explicit user ratings, the pLSA model needs to be generalized appropriately. As discussed in [14], there are a number of ways to extend the *dependency structure* of the cooccurrence model to include an additional response variable. The extension that has proven most useful in empirical evaluations is to introduce direct dependencies of the response variable on the item in question, but to mediate the dependency on the user through a latent variable. The latter is supposed to capture user communities or interest groups. Formally, this results in the following mixture model

$$p(v|u,y) = \sum_z p(v|y,z)P(z|u)\,. \tag{6}$$

In addition to specifying the dependency structure, one also needs to define a *parametric form* for the conditional probability density $p(v|y,z)$. Whereas previous models have only dealt with multinomial sampling models [14] which are appropriate for categorical ratings (e.g., binary votes), we investigate the use of a Gaussian model for $p(v|y,z)$ here. We hence propose to introduce location parameters $\mu_{y,z} \in \Re$ for the mean rating and scale parameters $\sigma_{y,z} \in \Re^+$ for the spread of the ratings. This effectively defines a Gaussian mixture model with user-specific mixing weights

$$p(v|u,y) = \sum_z P(z|u)\,p(v;\mu_{y,z},\sigma_{y,z}), \quad \text{with} \tag{7}$$

$$p(v;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(v-\mu)^2}{2\sigma^2}\right] \tag{8}$$

Finally, notice that the expected response can be easily computed as

$$E[v|u,y] = \int_{\mathcal{V}} v\,p(v|u,y)dv = \sum_z P(z|u)\mu_{y,z}\,. \tag{9}$$

3.3 User Normalization

The Gaussian model presented so far assumes that all users express their vote on a common scale. However, it is known that different users may associate subjectively different meanings with ratings. For instance, votes on 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 [15] which effectively normalize ratings by a user's mean rating as well as their spread. Since Gaussian pLSA is based on numerical response variables, this can also be accommodated in our model-based approach. To that extent we propose the following normalized model

$$p(v|u,y) = \mu_u + \sigma_u \sum_z P(z|u)p\left(\frac{v-\mu_u}{\sigma_u};\mu_{y,z},\sigma_{y,z}\right)\,, \tag{10}$$

where μ_u denotes a user-specific mean rating and σ_u a user-specific standard variation. Thus the assumption is that the *normalized* ratings within a user community are normally distributed for every item, where the normalization step performs a simple transformation of the observed ratings according to

$$(u,v,y) \mapsto (u,v',y), \quad \text{with} \quad v' = \frac{v-\mu_u}{\sigma_u} \tag{11}$$

For users with a small number of ratings, one has to be careful to perform appropriate smoothing in estimating the user-specific parameters, which is in particular true for the standard deviations σ_u, since the empirical estimates may be very unreliable due to sampling noise. We have used the following scheme to smooth the estimates of the means and variances,

$$\mu_u = \frac{\sum_{\langle u',v,y\rangle:u'=u} v + q\bar{\mu}}{N_u + q}, \tag{12}$$

$$\sigma_u^2 = \frac{\sum_{\langle u',v,y\rangle:u'=u}(v-\mu_u)^2 + q\bar{\sigma}^2}{N_u + q}, \tag{13}$$

where $\bar{\mu}$ denotes the overall mean vote and $\bar{\sigma}^2$ 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). Notice that we shrink the empirical (maximum likelihood) estimate towards the pooled estimates, which is a common scheme employed, for instance, in hierarchical Bayesian modeling.

3.4 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 risk function in Eq. (3). The following derivations are for the free prediction case, analogous equations for the forced prediction case can essentially be obtained by replacing $p(v, y|z)$ by $p(v|y, z)$.

The Expectation Maximization (EM) algorithm is a standard methods for statistical inference that can be used to (approximately) maximize the log-likelihood in mixture models like pLSA [10]. 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 triplet (u, v, y), we would in fact observe a 4-tuple (u, v, y, z). The complete data model is then given by $p(v, y, z|u) = p(v, y|z)P(z|u)$ and the corresponding (negative) log-likelihood function can be written as

$$\mathcal{R}^c(\theta) = -\sum_{\langle u,v,y,z\rangle} \log p(v, y|z) + \log P(z|u)\,. \qquad (14)$$

Since the states of the latent variables are not known, we introduce a so-called *variational probability distribution*

$$Q(z|u, v, y), \quad \text{such that } \sum_z Q(z|u, v, y) = 1 \qquad (15)$$

for all triplets (u, v, y). Using Q one can define a family of risk functions (one risk function for every choice of Q)

$$\begin{aligned}\bar{\mathcal{R}}(\theta, Q) \;=\; -\sum_{\langle u,v,y\rangle}\sum_z Q(z|u, v, y)\,[\log p(v, y|z) \quad (16)\\ + \log P(z|u)]\end{aligned}$$

Exploiting the concavity of the logarithm and using Jensen's inequality, 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),

$$\begin{aligned}\mathcal{R}(\theta) &= -\sum_{\langle u,v,y\rangle} \log \sum_z Q(z|u, v, y)\frac{p(v, y|z)P(z|u)}{Q(z|u, v, y)}\\ &\leq -\sum_{\langle u,v,y\rangle}\sum_z Q(z|u, v, y) \log \frac{p(v, y|z)P(z|u)}{Q(z|u, v, y)}\\ &= \bar{\mathcal{R}}(\theta, Q) - \sum_{\langle u,v,y\rangle} H(Q(\cdot|u, v, y)) \qquad (17)\end{aligned}$$

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 $\hat{\theta}$, i.e. optimizing Eq. (17) with respect to the variational distribution Q. This is called the E-step and amounts to computing the posterior probabilities of the hidden variable,

$$Q^*(z|u, v, y; \hat{\theta}) = P(z|u, v, y; \hat{\theta}) = \frac{\hat{p}(v, y|z)\hat{P}(z|u)}{\sum_{z'} \hat{p}(v, y|z')\hat{P}(z'|u)}\,. \quad (18)$$

A formal derivation using the technique of Lagrange multipliers is straightforward. The hat indicates quantities parameterized by $\hat{\theta}$. Obviously the posterior probabilities need only to be computed for triplets (u, v, y) that have actually been observed. Averaging $\mathcal{R}^c$ with respect to the posterior distribution calculated from Eq. (18) then yields

$$\begin{aligned}\bar{\mathcal{R}}(\theta, \hat{\theta}) \;=\; -\sum_{\langle u,y\rangle}\sum_z \frac{\hat{p}(v, y|z)\hat{P}(z|u)}{\sum_{z'} \hat{p}(v, y|z')\hat{P}(z'|u)}\\ [\log p(v, y|z) + \log P(z|u)] \quad (19)\end{aligned}$$

which needs to be optimized with respect to the parameters θ and which constitutes the Maximization (M) step. The maximization problem can be readily solved by introducing Lagrange multipliers for every normalization constraint and solving the resulting constrained optimization problem, which leads to the following set of equations for the user-specific mixing proportions

$$P(z|u) \;=\; \frac{\sum_{\langle u',v,y\rangle:u'=u} P(z|u, v, y; \hat{\theta})}{\sum_{z'}\sum_{\langle u',v,y\rangle:u'=u} P(z'|u, v, y; \hat{\theta})}\,. \quad (20)$$

Similarly, one obtains the following equations for the parameters of the Gaussian distributions,

$$\mu_{y,z} \;=\; \frac{\sum_{\langle u,v,y'\rangle:y'=y} v\, P(z|u, v, y)}{\sum_{\langle u,v,y'\rangle:y'=y} P(z|u, v, y)} \qquad (21)$$

$$\sigma^2_{y,z} \;=\; \frac{\sum_{\langle u,v,y'\rangle:y'=y} (v - \mu_{y,z})^2\, P(z|u, v, y)}{\sum_{\langle u,v,y'\rangle:y'=y} P(z|u, v, y)}\,. \quad (22)$$

Notice that ratings v in triplets $\langle u, v, y\rangle$ refer to the user-normalized ratings as given by Eq. (11).

Eqs. (21,22) 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. The complete EM algorithm now proceeds by alternating the E-step in Eq. (18) with the M-step in Eqs. (20-22).

3.5 Regularization

Learning statistical models with many parameters from a limited amount of data bears the risk of overfitting. We have hence used a regularization technique to address this problem. While we have investigated a modification of EM, called tempered EM in [10], a conceptually simpler and computationally less expensive approach has proven to be sufficient for regularizing Gaussian pLSA models. Retaining a fraction of the data from training, we have used the held-out data to perform early stopping, i.e. we terminate the EM iterations once the accuracy on the held-out data deteriorates. Finally, we perform one last iteration of EM on the full training data.

4. EXPERIMENTS

4.1 Data Set

The data set we have used in our experiments is the EachMovie data. 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 2.8 million ratings. The rating scale is discrete, taking values from 0 (no star) to 5 (five stars).[1] 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.

[1] The original ratings are between 0 and 1 and have been multiplied by a factor of 5.

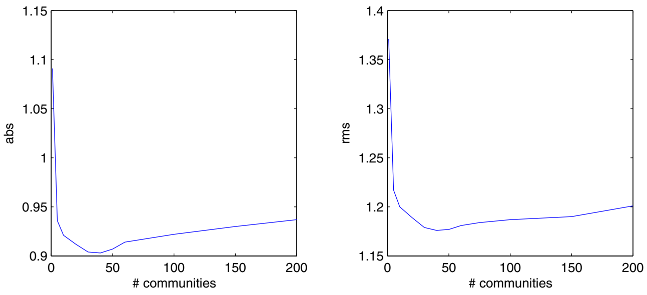


Figure 1: Predictive performance in terms of absolute (left) and rms error (right) for the normalized Gaussian ranking model as a function of the number of user communities.

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.

4.2 Evaluation Metric

A thorough empirical analysis of collaborative filtering algorithms has been presented in [5] 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 the squared loss are summarized as the rooted mean square (RMS) error, which is the square root of the empirical average of the loss. 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 valid 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 in [5]. 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. Ratings for items that have been used for training are not included in the ranking. We then use the following rank score for τ,

$$R(u, \tau) = \sum_{\langle u', v, y \rangle : u = u'} 2^{-\frac{\tau(y)-1}{\alpha - 1}} \max(v - \bar{v}, 0), \qquad (23)$$

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. [5])

$$R = 100 \frac{\sum_u R(u, \tau_u)}{\sum_u \max_{\tau'} R(u, \tau')} . \qquad (24)$$

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.

4.3 Evaluation Protocols

We have used a 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 two 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 [5]. Actually, 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 would like to point out that we have not used the *GivenN* (e.g., $N = 5, 10, 20$) protocols proposed in [5]. The reason is that we cannot afford to perform a complete model retraining for evaluating the system performance on a single user. On the other hand, creating a training data set by keeping only maximally N observations for *every* user would sparsify the data considerably and not provide a useful approximation.

4.4 Baseline Methods

We have implemented two baseline methods to calibrate the achieved results. As a first simple baseline, we have used a largely non-personalized prediction/ranking function which simply computes the average normalized rating for each movie over the user population as a whole. Notice that this does take user specific mean votes and variances into account and that it is equivalent to the normalized Gaussian pLSA model with $k = 1$. This methods is called “Baseline” in Tables 1 and 2.

In addition, we have implemented a standard memory-based method which computes similarities between user profiles based on the Pearson correlation coefficient. This implementation does not include possible improvements such as inverse user frequency or case amplification [15]. The corresponding results in Tables 1 and 2 are labeled as “Pearson”.

4.5 Results: Prediction Scenario

Table 1 summarizes the results, including a comparison with the standard memory based method based on the Pearson correlation coefficient and results published in [5] for various methods (Bayesian clustering = BC, Bayesian networks = BN, correlation = CR). The baseline is simply defined by the overall mean vote for each item.

As can be seen, the proposed Gaussian pLSA model outperforms the memory-based method and also achieves better results than the ones published in [5]. With respect to the latter results one has to acknowledge a somewhat different setup though, which has lead 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 [5].

It can also be seen that the accuracy of the Gaussian

Method	absolute error			relative improvement		
	ABS	RMS	0/1 loss	ABS	RMS	0/1 loss
Baseline	1.091	1.371	71.2	±0	±0	±0
Pearson	0.951	1.231	64.7	12.8%	10.2%	9.1%
Gaussian	0.975	1.252	67.1	5.8%	8.7 %	5.7%
Gauss, normalized	**0.903**	**1.176**	**64.3**	17.2%	14.2%	9.7%
CR [5]	0.994	-	-	-	-	-
BC [5]	1.103	-	-	-	-	-
BN [5]	1.066	-	-	-	-	-

Table 1: Prediction accuracy of various methods in *forced* prediction mode.

Method	rank gain [5]	rel. improv.	abs	rms
Baseline	16.76	±0	1.091	1.371
Pearson	21.14	26.1	**0.951**	1.231
Gaussian	21.80	30.0	1.020	1.290
Gaussian, normalized	**24.23**	44.5	0.955	**1.211**

Table 2: Performance of different methods in *free* prediction mode according to ranking criterion.

model is quite poor without the user-specific normalization, which is a crucial ingredient of the proposed model. It is also quite remarkable that this result is obtained with a model involving a relatively small number of communities ($k = 40$). We conclude from this that the assumption of user-specific rating scales encodes useful prior knowledge.

Figure 1 shows how the prediction accuracy for the Gaussian pLSA model varies with model complexity, i.e. the cardinality of the state space of the latent variable, corresponding to the number of latent communities searched for. The graph shows a clear optimum around $k = 40$ communities, after which the performance slowly degrades with model size. This behavior is strikingly different from results obtained with a multinomial sampling model in [16], where the obtained accuracy constantly improves with the number of communities until it levels off at around $k = 200$.

4.6 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. (24) to benchmark different algorithms. Again we have used the leave-one-out protocol. The results are summarized in Table 2.

The best ranking scores are obtained again by the normalized Gaussian pLSA model. 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 is slightly higher than with the memory-based approach.

4.7 Mining User Communities

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 visualized the items for each latent variable state by sorting them according to the popularity within an interest group as measured by $P(y|z)$ (irrespective of the actual vote). The mean vote $\mu_{y,z}$ is displayed in rectangular brackets under the movie title. Figure 2 and 3 display the interest groups extracted by a normalized Gaussian model with $k = 40$.

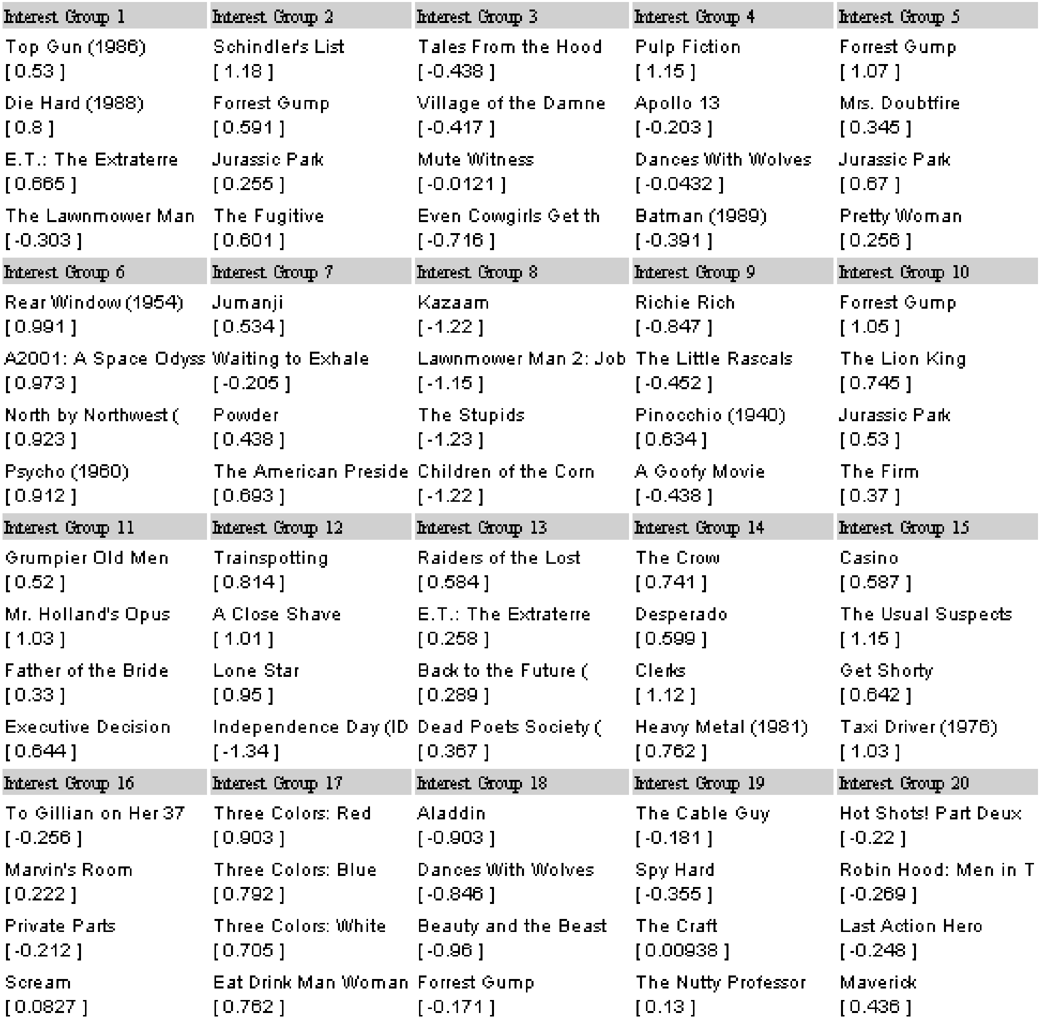
4.8 Computational Complexity

Finally, we would like to emphasize the fact that the computational complexity for computing predictions for an active user scales with the number of communities k, but not with the number of users in the data base. Given that the optimal accuracy was achieved for a model with $k = 40$, this compares very favorably with (naive) memory-based approaches which scale linearly in the total number of users (more than $60,000$ in the case of EachMovie). Notice also that the complexity of adding a new user to the database scales with k as well, as long as the community parameters $P(y|z)$ and the Gaussian parameters $\mu_{y,z}$ and $\sigma_{y,z}$ are held fixed. This can be achieved by performing an operation called fold-in [11]. The model-based representation also achieves a significant compression in terms of space.

The learning phase obviously requires more resources. Notice that the complexity of a single EM iteration scales with k times the number of observed ratings. The typical number of EM iterations performed in the experiments is around $25 - 30$. The overall running time for training the Gaussian pLSA model with $k = 40$ was very reasonable for practical purposes: approximately 15 minutes on a standard computer system powered by a Pentium III 1GHz CPU.

5. CONCLUSIONS

We have presented a powerful method for collaborative filtering and for mining of user data based on a novel statistical latent class model, Gaussian pLSA. The method achieves competitive recommendation and prediction accuracies, is highly scalable, and extremely flexible. Predictions and recommendations can be computed in constant time and without access to the data base of original user profiles. Conceptionally, the decomposition of user preferences is a feature that clearly distinguishes our approach from traditional memory-based approaches.



Interest Group 1	Interest Group 2	Interest Group 3	Interest Group 4	Interest Group 5
Top Gun (1986) [0.53]	Schindler's List [1.18]	Tales From the Hood [-0.438]	Pulp Fiction [1.15]	Forrest Gump [1.07]
Die Hard (1988) [0.8]	Forrest Gump [0.591]	Village of the Damne [-0.417]	Apollo 13 [-0.203]	Mrs. Doubtfire [0.345]
E.T.: The Extraterre [0.665]	Jurassic Park [0.255]	Mute Witness [-0.0121]	Dances With Wolves [-0.0432]	Jurassic Park [0.67]
The Lawnmower Man [-0.303]	The Fugitive [0.601]	Even Cowgirls Get th [-0.716]	Batman (1989) [-0.391]	Pretty Woman [0.256]
Interest Group 6	Interest Group 7	Interest Group 8	Interest Group 9	Interest Group 10
Rear Window (1954) [0.991]	Jumanji [0.534]	Kazaam [-1.22]	Richie Rich [-0.847]	Forrest Gump [1.05]
A2001: A Space Odyss [0.973]	Waiting to Exhale [-0.205]	Lawnmower Man 2: Job [-1.15]	The Little Rascals [-0.452]	The Lion King [0.745]
North by Northwest ([0.923]	Powder [0.438]	The Stupids [-1.23]	Pinocchio (1940) [0.634]	Jurassic Park [0.53]
Psycho (1960) [0.912]	The American Preside [0.693]	Children of the Corn [-1.22]	A Goofy Movie [-0.438]	The Firm [0.37]
Interest Group 11	Interest Group 12	Interest Group 13	Interest Group 14	Interest Group 15
Grumpier Old Men [0.52]	Trainspotting [0.814]	Raiders of the Lost [0.584]	The Crow [0.741]	Casino [0.587]
Mr. Holland's Opus [1.03]	A Close Shave [1.01]	E.T.: The Extraterre [0.258]	Desperado [0.599]	The Usual Suspects [1.15]
Father of the Bride [0.33]	Lone Star [0.95]	Back to the Future ([0.289]	Clerks [1.12]	Get Shorty [0.642]
Executive Decision [0.644]	Independence Day (ID [-1.34]	Dead Poets Society ([0.367]	Heavy Metal (1981) [0.762]	Taxi Driver (1976) [1.03]
Interest Group 16	Interest Group 17	Interest Group 18	Interest Group 19	Interest Group 20
To Gillian on Her 37 [-0.256]	Three Colors: Red [0.903]	Aladdin [-0.903]	The Cable Guy [-0.181]	Hot Shots! Part Deux [-0.22]
Marvin's Room [0.222]	Three Colors: Blue [0.792]	Dances With Wolves [-0.846]	Spy Hard [-0.355]	Robin Hood: Men in T [-0.269]
Private Parts [-0.212]	Three Colors: White [0.705]	Beauty and the Beast [-0.96]	The Craft [0.00938]	Last Action Hero [-0.248]
Scream [0.0827]	Eat Drink Man Woman [0.762]	Forrest Gump [-0.171]	The Nutty Professor [0.13]	Maverick [0.436]

Figure 2: User communities 1-20 (of 40) extracted from the EachMovie data set with a Gaussian pLSA model. Numbers in brackets denote the means $\mu_{y,z}$.

6. ACKNOWLEDGMENTS

This research was sponsored by an NSF-ITR grant, award number IIS-0085836. The EachMovie preference data is by courtesy of Digital Equipment Corporation and was generously provided by Paul McJones.

7. REFERENCES

[1] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[2] P. Resnik, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM, Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[3] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

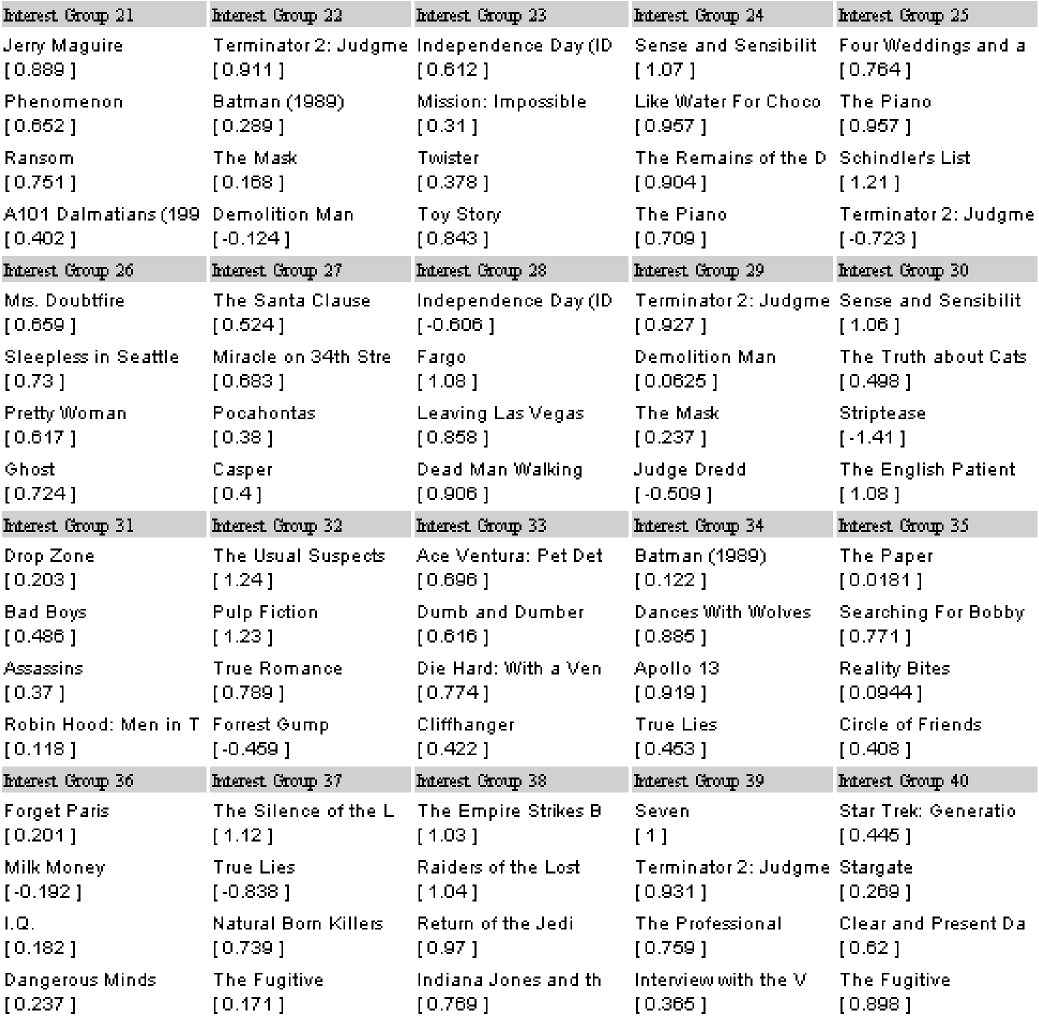
[4] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[5] J. S. Breese, D. Heckerman, and C. Kardie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[6] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems.* AAAI Press, 1998.

[7] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Recommender System Workshop*, pages 11–15, 1998.

[8] Y.-H. Chien and E.I. George. A Bayesian model for



Interest Group 21	Interest Group 22	Interest Group 23	Interest Group 24	Interest Group 25
Jerry Maguire [0.889]	Terminator 2: Judgme [0.911]	Independence Day (ID [0.612]	Sense and Sensibilit [1.07]	Four Weddings and a [0.764]
Phenomenon [0.652]	Batman (1989) [0.289]	Mission: Impossible [0.31]	Like Water For Choco [0.957]	The Piano [0.957]
Ransom [0.751]	The Mask [0.168]	Twister [0.378]	The Remains of the D [0.904]	Schindler's List [1.21]
A101 Dalmatians (199 [0.402]	Demolition Man [-0.124]	Toy Story [0.843]	The Piano [0.709]	Terminator 2: Judgme [-0.723]

Interest Group 26	Interest Group 27	Interest Group 28	Interest Group 29	Interest Group 30
Mrs. Doubtfire [0.659]	The Santa Clause [0.524]	Independence Day (ID [-0.606]	Terminator 2: Judgme [0.927]	Sense and Sensibilit [1.06]
Sleepless in Seattle [0.73]	Miracle on 34th Stre [0.683]	Fargo [1.08]	Demolition Man [0.0625]	The Truth about Cats [0.498]
Pretty Woman [0.617]	Pocahontas [0.38]	Leaving Las Vegas [0.858]	The Mask [0.237]	Striptease [-1.41]
Ghost [0.724]	Casper [0.4]	Dead Man Walking [0.906]	Judge Dredd [-0.509]	The English Patient [1.08]

Interest Group 31	Interest Group 32	Interest Group 33	Interest Group 34	Interest Group 35
Drop Zone [0.203]	The Usual Suspects [1.24]	Ace Ventura: Pet Det [0.696]	Batman (1989) [0.122]	The Paper [0.0181]
Bad Boys [0.486]	Pulp Fiction [1.23]	Dumb and Dumber [0.616]	Dances With Wolves [0.885]	Searching For Bobby [0.771]
Assassins [0.37]	True Romance [0.789]	Die Hard: With a Ven [0.774]	Apollo 13 [0.919]	Reality Bites [0.0944]
Robin Hood: Men in T [0.118]	Forrest Gump [-0.459]	Cliffhanger [0.422]	True Lies [0.453]	Circle of Friends [0.408]

Interest Group 36	Interest Group 37	Interest Group 38	Interest Group 39	Interest Group 40
Forget Paris [0.201]	The Silence of the L [1.12]	The Empire Strikes B [1.03]	Seven [1]	Star Trek: Generatio [0.445]
Milk Money [-0.192]	True Lies [-0.838]	Raiders of the Lost [1.04]	Terminator 2: Judgme [0.931]	Stargate [0.269]
I.Q. [0.182]	Natural Born Killers [0.739]	Return of the Jedi [0.97]	The Professional [0.759]	Clear and Present Da [0.62]
Dangerous Minds [0.237]	The Fugitive [0.171]	Indiana Jones and th [0.769]	Interview with the V [0.365]	The Fugitive [0.898]

Figure 3: User communities 21-40 (of 40) extracted from the EachMovie data set with a Gaussian pLSA model. Numbers in brackets denote the means $\mu_{y,z}$.

collaborative filtering. In *Online Proceedings of The Seventh International Workshop on Artificial Intelligence and Statistics*, 1999.

[9] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. M. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[10] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning Journal*, 42(1):177–196, 2001.

[11] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

[12] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system – a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000.

[13] J. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57, 2002.

[14] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proceedings of the International Joint Conference in Artificial Intelligence*, 1999.

[15] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd ACM-SIGIR International Conference on Research and Development in Information Retrieval*, 1999.

[16] T. Hofmann. What people (don't) want. In *European Conference on Machine Learning (ECML)*, 2001.