A Generalized Probabilistic Framework and its Variants for Training Top-k Recommender Systems

Harald Steck
Bell Labs, Alcatel-Lucent
Murray Hill, NJ
Harald.Steck@alcatel-lucent.com

Yu Xin*
CSAIL MIT
Cambridge, MA
YuXin@mit.edu

ABSTRACT

Accounting for missing ratings in available training data was recently shown [3, 17] to lead to large improvements in the top-k hit rate of recommender systems, compared to state-of-the-art approaches optimizing the popular root-mean-square-error (RMSE) on the observed ratings. In this paper, we take a Bayesian approach, which lends itself naturally to incorporating background knowledge concerning the missing-data mechanism. The resulting log posterior distribution is very similar to the objective function in [17]. We conduct elaborate experiments with real-world data, testing several variants of our approach under different hypothetical scenarios concerning the missing ratings. In the second part of this paper, we provide a generalized probabilistic framework for dealing with possibly *multiple observed rating values* for a user-item pair. Several practical applications are subsumed by this generalization, including aggregate recommendations (e.g., recommending artists based on ratings concerning their songs) as well as collaborative filtering of sequential data (e.g., recommendations based on TV consumption over time). We present promising preliminary experimental results on IP-TV data.

Categories and Subject Descriptors

H.2.8 **[Database Management]:** Database Applications–*Data Mining*

General Terms

Algorithms

Keywords

Recommender Systems

*This work was done while an intern at Bell Labs, Alcatel-Lucent.

1. INTRODUCTION

The idea of recommender systems is to automatically suggest items to each user that s/he may find appealing. The quality of recommender systems can be assessed with respect to various criteria, including accuracy, diversity, surprise / serendipity, and explainability of recommendations.

This paper is concerned with accuracy. The *root mean squared error (RMSE)* has become the most popular accuracy measure in the literature of recommender systems–for training and testing. Its computational efficiency is one of its main advantages. Impressive progress has been made in predicting rating values with small RMSE, and it is impossible to name all approaches, e.g., [4, 6, 7, 11, 13]). There is, however, also some work on optimizing the ranking of items, e.g., measured in terms of normalized Discounted Cumulative Gain (nDCG) [18]. Despite their differences, they have in common that they were trained and tested on *observed ratings only*. Obviously, these measures cannot immediately be evaluated if some items have unobserved ratings.

In this paper, we consider the *top-k hit rate*–based on *all* (unrated) items–as the natural accuracy measure for recommender systems, as only a few out of *all* unrated items can be recommended to a user in practice (see Section 3 for exact definition of top-k hit rate). While this measure is computationally tractable for *testing* the predictions of recommender systems, unfortunately it is computationally very costly for *training* recommender systems. For training, we thus resort to appropriate surrogate objective functions that are computationally efficient.

In recent work [3, 17], it was shown that the top-k hit rate can be significantly improved on large real-world data by accounting for the fact that the observed ratings provide a skewed picture of the (unknown) distribution concerning *all* (unrated) items and users.

Motivated by the results of [17], as the first contribution of this paper, in Section 2 we present a probabilistic approach that allows us to naturally include background knowledge concerning the (unknown) distribution of all items and users into our training objective function, the posterior probability of the model.

In our second contribution, we conduct elaborate experiments on the Netflix Prize data [1] and test several models under different hypothetical scenarios concerning the missing ratings in Section 4. These different scenarios serve as a sensitivity analysis, as the ground truth of the missing data-mechanism is unknown due to lack of data. These experiments are based on our *popularity-stratified* recall measure, which we define in Section 3.

As the third contribution of this paper, we generalize this probabilistic approach as to account for possibly *multiple observed rating values* for a user-item pair in Section 5. This general framework subsumes several applications in addition to the one outlined in Section 2. Two of which are outlined in Section 5: while the training objective function for a recommender system concerning TV programs seems motivated in an ad-hoc manner in [5], we show that it can be understood and improved naturally in a Bayesian framework; apart from that, we also provide a Bayesian approach for making aggregate recommendations, e.g., recommending an artist or concert to a user based on the ratings given to individual songs.

2. MODEL TRAINING

In this section, we outline a probabilistic framework that allows us to incorporate background knowledge when training recommender systems on available data. The use of background knowledge in addition to the available training data can significantly improve the accuracy of recommender systems on performance measures like top-k hit rate, recall, precision, area under the ROC curve, etc. This was demonstrated for implicit feedback data in [5, 10], and for explicit feedback data in [3, 17]. Like in [17], we use background knowledge that missing rating values tend to reflect negative feedback, as experimentally observed in [9, 8]; i.e., negative feedback tends to be missing from the available data with a larger probability than positive feedback does.

The Bayesian approach lends itself naturally to this task. We consider the rating matrix R as a *matrix of random variables*: each element $R_{i,u}$ concerning item $i = 1, ..., i_0$ and user $u = 1, ..., u_0$ is a random variable with normal distribution, where i_0 denotes the number of items and u_0 is the number of users.

2.1 Model

We take a collaborative filtering approach, and use a low-rank matrix-factorization model, which has proven successful in many publications. Like the rating matrix, we consider our model as a *matrix of random variables*, M. Each random variable $M_{i,u}$ corresponds to the rating of item i assigned by user u. In matrix notation, it reads

$$M = r^{\text{offset}} + PQ^\top \tag{1}$$

where $r^{\text{offset}} \in \mathbb{R}$ is an offset value, and P, Q are low-rank matrices of random variables with dimensions $i_0 \times d_0$ and $u_0 \times d_0$, respectively, where rank $d_0 \ll i_0, u_0$. We use upper case symbols to denote random variables (with a Gaussian distribution), and lower case symbols to denote values.

2.2 Prior over Matrix Elements

In our Bayesian approach, we first define the usual prior over model parameters, concerning each entry of the low rank matrices P and Q (see also [12]):

$$\begin{aligned} p(M|\sigma_P^2, \sigma_Q^2) &= \left\{ \prod_i \prod_d \mathcal{N}(P_{i,d}|0, \sigma_{P,i}^2) \right\} \\ &\quad \cdot \left\{ \prod_u \prod_d \mathcal{N}(Q_{u,d}|0, \sigma_{Q,u}^2) \right\} \end{aligned} \tag{2}$$

The vectors of variances $\sigma_Q^2 = (\sigma_{Q,u}^2)_{u=1,...,u_0}$ and $\sigma_P^2 = (\sigma_{P,i}^2)_{i=1,...,i_0}$ for all users $u = 1, ..., u_0$ and items $i = 1, ..., i_0$ are free parameters of the zero-mean normal prior distribution, denoted by $\mathcal{N}$. There are several ways of defining the standard deviations in Eq. 2, eventually resulting in different kinds of regularization. The obvious choice is to assume that $\sigma_{P,i} = \sigma_{Q,u} = 1/\sqrt{2\lambda'} \quad \forall\, i, u$, with $\lambda' \in \mathbb{R}$. This results in the regularization term

$$\log p(M|\sigma_P^2, \sigma_Q^2) = -\lambda' \left(||P||_2^2 + ||Q||_2^2 \right) + c_1,$$

where $||\cdot||_2$ denotes the Frobenius norm of a matrix, and c_1 is an irrelevant constant when training our model.

When optimizing root mean square error on observed data (like in the Netflix Prize competition [1]), however, numerous experimental works reported significant improvements by using a different regularization. This is obtained by choosing the standard deviations σ_P and σ_Q as follows: $\sigma_{P,i} = 1/\sqrt{2\lambda' \cdot u_0(i)}$, $\sigma_{Q,u} = 1/\sqrt{2\lambda' \cdot i_0(u)}$, where $i_0(u)$ denotes the number of items rated by user u, and $u_0(i)$ is the number of users who rated item i. This results in the popular regularization term

$$\begin{aligned} &\log p(M|\sigma_P^2, \sigma_Q^2) \\ &= -\lambda' \left(\sum_i u_0(i) \sum_d P_{i,d}^2 + \sum_u i_0(u) \sum_d Q_{u,d}^2 \right) + c_2 \\ &= -\lambda' \left(\sum_{\text{observed } (i,u)} \left(\sum_d P_{i,d}^2 + Q_{u,d}^2 \right) \right) + c_2, \end{aligned} \tag{3}$$

where c_2 denotes again an irrelevant constant when training our model. Note that this choice increasingly regularizes the model parameters related to the items and users with a larger number of observed ratings. This may seem counter-intuitive at first glance. A theoretical explanation for this empirical finding was recently provided in [14].

2.3 Informative Background Knowledge

We now incorporate the following background knowledge into our sequential Bayesian approach: absent rating values tend to be lower than the observed ratings on average (see [17]). We insert this knowledge into our approach by means of a virtual data point for *each* pair (i, u): a virtual rating value $r_{i,u}^{\text{prior}}$ with small confidence (i.e., large variance $\sigma_{\text{prior},i,u}^2$). Then the likelihood of our model in light of these virtual data points reads (assuming i.i.d. data):

$$p(r^{\text{prior}}|M, \sigma_{\text{prior}}^2) = \prod_{\text{all } i} \prod_{\text{all } u} p(R_{i,u} = r_{i,u}^{\text{prior}}|M_{i,u}, \sigma_{\text{prior},i,u}^2), \tag{4}$$

where r^{prior} denotes the matrix of virtual data points $r_{i,u}^{\text{prior}}$, and σ_{prior}^2 the matrix with elements $\sigma_{\text{prior},i,u}^2$. We assume that the probabilities in this likelihood are determined by normal distributions with mean $M_{i,u}$ and variance $\sigma_{\text{prior},i,u}^2$. The log likelihood then reads

$$\log p(r^{\text{prior}}|M, \sigma_{\text{prior}}^2) = -\sum_{\text{all } i} \sum_{\text{all } u} w_{i,u}^{\text{prior}} \left(r_{i,u}^{\text{prior}} - M_{i,u} \right)^2 + c_3 \tag{5}$$

where we defined the weights of the virtual data points as $w_{i,u}^{\text{prior}} = 1/(2\sigma_{\text{prior},i,u}^2)$; c_3 is again an irrelevant constant when training our model.

With Bayes rule, we obtain the posterior distribution of

the model in light of these virtual data points:

$$p(M|r^{\text{prior}}, w^{\text{prior}}) = \frac{p(r^{\text{prior}}, w^{\text{prior}}|M)p(M)}{p(r^{\text{prior}}, w^{\text{prior}})}. \quad (6)$$

This equation combines our prior concerning the elements in the matrices P and Q (for regularization) with our background knowledge on the expected rating values. This serves as our prior when observing the actual rating values in the training data.

2.4 Training Data

Now we use the rating values actually observed in the training data. The likelihood of the model in light of observed rating values $r^{\text{obs}}_{i,u}$ reads (assuming i.i.d. data):

$$p(r^{\text{obs}}|M, \sigma^2_{\text{obs}}) = \prod_{\text{observed } (i,u)} p(R_{i,u} = r^{\text{obs}}_{i,u}|M_{i,u}, \sigma^2_{\text{obs},i,u})$$

Again assuming a normal distribution, the log likelihood reads:

$$\log p(r^{\text{obs}}|M, \sigma^2_{\text{obs}}) = -\sum_{\text{observed } (i,u)} w^{\text{obs}}_{i,u}(r^{\text{obs}}_{i,u} - M_{i,u})^2 + c_4 \quad (7)$$

where we defined the weights of the observed rating values as $w^{\text{obs}}_{i,u} = 1/(2\sigma^2_{\text{obs},i,u})$; c_4 is again an irrelevant constant when training our model.

2.5 Posterior

The posterior after seeing the observed ratings is again obtained by Bayes rule (we omit the weights $w^{\text{obs}}, w^{\text{prior}}$ for brevity of notation here):

$$\begin{aligned} p(M|r^{\text{obs}}, r^{\text{prior}}) &= \frac{p(r^{\text{obs}}|M, r^{\text{prior}})p(M|r^{\text{prior}})}{p(r^{\text{obs}})} \\ &\propto p(r^{\text{obs}}|M, r^{\text{prior}})p(r^{\text{prior}}|M)p(M) \end{aligned} \quad (8)$$

where we assumed in the denominator that the observed ratings are independent of the chosen prior ratings, i.e., $p(r^{\text{obs}}|r^{\text{prior}}) = p(r^{\text{obs}})$. Substituting Eqs. 3, 5, 6 and 7 into Eq. 8, we obtain the following log posterior:

$$\begin{aligned} &\log p(M|r^{\text{obs}}, r^{\text{prior}}, w^{\text{obs}}, w^{\text{prior}}, \lambda) = \\ &-\sum_{\text{obs.}(i,u)} w^{\text{obs}}_{i,u} \left\{ (r^{\text{obs}}_{i,u} - M_{i,u})^2 + \lambda \sum_d \left[P^2_{i,d} + Q^2_{u,d}\right] \right\} \\ &-\sum_{\text{all}(i,u)} w^{\text{prior}}_{i,u} \left\{ (r^{\text{prior}}_{i,u} - M_{i,u})^2 + \lambda \sum_d \left[P^2_{i,d} + Q^2_{u,d}\right] \right\} \\ &+c_5 \end{aligned} \quad (9)$$

We found that using a prior that involves also the regularization term of P and Q in the third line in Eq. 9 leads to a slight improvements in our experimental results. The weights $w^{\text{obs}}_{i,u}$ and $w^{\text{prior}}_{i,u}$ as well as λ' are absorbed in λ; this is a slight but straight-forward generalization of the prior in Section 2.2; c_5 is again an irrelevant constant for training.

Eq. 9 serves as our training objective function. For simplicity, we choose the same value for all virtual rating values r^{prior}. For computational efficiency, we choose our model offset to equal the prior rating values: $r^{\text{offset}} = r^{\text{prior}}$. Its main effect is that this retains the sparsity of the observed rating matrix. Apart from that, it also leads to the simplification: $(r^{\text{prior}}_{i,u} - M_{i,u})^2 = ((PQ^\top)_{i,u})^2$. For simplicity, we set $w^{\text{obs}}_{i,u} = 1$ for all observed pairs (i, u), and also choose all prior weights to be identical: $w^{\text{prior}} = w^{\text{prior}}_{i,u}$ for all (i, u). In summary, the three tuning parameters in Eq. 9 are w^{prior}, r^{prior} and λ, which can be chosen as to optimize the performance measure on cross-validation data.

2.6 MAP Estimate of Model

For computational efficiency, our training aims to find the *maximum-a-posteriori (MAP)* parameter estimate of our model, i.e., the MAP estimates $\hat{P}$ and $\hat{Q}$ of the matrices P and Q. We use the *alternating least squares* approach. The idea is that one matrix can be optimized exactly while the other one is assumed fixed. A local maximum of the log posterior can be found by alternating between the matrices $\hat{P}$ and $\hat{Q}$. While local optima exist [16], we did not find this to cause major computational problems in our experiments. The update equation for each row i of $\hat{P}$ is (for fixed $\hat{Q}$):

$$\begin{aligned} \hat{P}_{i,\cdot} = (\bar{r}_{i,\cdot} - r^{\text{prior}})(\tilde{W}^{(i)} + w^{\text{prior}} I)\hat{Q} \cdot \\ \left[\hat{Q}^\top(\tilde{W}^{(i)} + w^{\text{prior}} I)\hat{Q} + \lambda(\text{tr}(\tilde{W}^{(i)}) + w^{\text{prior}} u_0)I\right]^{-1} \end{aligned} \quad (10)$$

where $\bar{r}_{i,u} = (r^{\text{obs}}_{i,u} w^{\text{obs}}_{i,u} + r^{\text{prior}} w^{\text{prior}}_{i,u})/(w^{\text{obs}}_{i,u} + w^{\text{prior}}_{i,u})$ denotes the average rating; we defined $w^{\text{obs}}_{i,u} = 0$ if rating at (i, u) is missing; note that $\bar{r}_{i,\cdot} - r^{\text{prior}} = 0$ if rating is missing for (i, u); $\tilde{W}^{(i)}$ is a diagonal matrix containing the i^{th} column of the weight matrix w^{obs}; the trace is $\text{tr}(\tilde{W}^{(i)}) = \sum_{u \in S_i} w^{\text{obs}}_{i,u}$, where S_i is the set of users who rated item i; I denotes the identity matrix; and u_0 is the number of users. This equation can be re-written for efficient computations, e.g., see [17]. The update equation for $\hat{Q}$ is analogous.

3. MODEL TESTING

A key challenge in testing recommender systems is that the observed ratings in the available data typically provide a skewed picture of the (unknown) true distribution concerning *all* ratings [9, 8]. This may be caused by the fact that users are free to choose what items to rate, and they tend to not rate items that would otherwise receive a low rating. If the ratings are missing not at random (MNAR), it is not guaranteed that correct or meaningful results are obtained from testing a recommender system on the *observed ratings only*. The latter is, however, common practice in the literature or recommender systems, using measures like root mean square error or nDCG [4, 6, 7, 11, 13, 18].

The *top-k hit-rate / recall* of *relevant* items is a particularly useful performance measure for assessing the accuracy of recommender systems [17]. An item i is *relevant* to user u if s/he finds this item interesting or appealing [17]. For instance, in the Netflix data [1] we consider items i with a 5-star rating, $r^{\text{obs}}_{i,u} = 5$, as relevant to user u.

Recall can be calculated for a user by ranking the items according to their scores predicted by the recommender system, and determining the fraction of relevant items that are among the top-k items, i.e., the k items with the highest scores. The value of $k \in \mathbb{N}$ has to be chosen, e.g., as the number of items that can be recommended to a user. Only small values of k are important in practice. The goal is to maximize recall for the chosen value of k.

Recall has two interesting properties in this context [17]: it is proportional to precision on fixed data and fixed k when comparing different recommender systems with each other. In other words, the recommender system with the larger

recall also has the larger precision. More interestingly, however, recall can be calculated from the available MNAR data and provides an *unbiased* estimate for the recall concerning the (unknown) complete data (which comprises all rating values of all users) under the following assumption: the *relevant* ratings are *missing at random*, while an arbitrary missing-data mechanism may apply to all other rating values (as long as they are missing with a larger probability than the relevant ones) [17]. This assumption is much milder than the one underlying the popular approach of ignoring missing ratings, i.e., assuming that *all* ratings are missing at random.

The expected performance on the (unknown) *complete data* is important because it is directly related to *user experience*: the recommender system has to pick a few items from among all items the user has not rated yet (and which may hence be new to the user); one can expect the distribution on all unrated items to be well-approximated by the distribution on all (rated and unrated) items under the (mild) assumption that only a small fraction of the relevant ratings has been observed.

Given that ground truth (i.e., the complete data) is typically not available (at low cost), the validity of the assumption in [17] cannot be verified in practice. For this reason, we carry out a sensitivity analysis in the following. We relax this assumption even further and determine its effect on the recall test-results for different models. Note that the assumption in [17] allows for an arbitrary missing-data mechanism concerning all ratings, except for the relevant ratings; only the latter are assumed to be missing at random. For this reason, the following is concerned with the *relevant ratings only.*

We consider the case that the probability of observing a relevant rating depends on the popularity of items. We define the popularity of an item by the number $N^+_{\text{complete},i}$ of *relevant* ratings it obtained in the (unknown) complete data. Let $N^+_{\text{obs},i}$ be the number of relevant ratings observed in the available data; then the probability of observing a relevant rating regarding item i is

$$p_{\text{obs}}(i) = \frac{N^+_{\text{obs},i}}{N^+_{\text{complete},i}}. \tag{11}$$

Assuming that there are no additional (possibly hidden) factors underlying the missing data mechanism concerning the relevant ratings, we obviously obtain an unbiased estimate for recall on the (unknown) complete data by calculating the *popularity-stratified recall* (for user u),

$$\text{recall}_u(k) = \frac{\sum_{i \in S_u^{+,k}} s_i}{\sum_{i \in S_u^+} s_i} \tag{12}$$

on the available MNAR data; S_u^+ denotes the set of *relevant* items of user u; $S_u^{+,k}$ is the subset of relevant items ranked in the top k items based on the predictions of the recommender system; the popularity-stratification weight for each item i is

$$s_i = \frac{1}{p_{\text{obs}}(i)}.$$

If we choose the stratification weights $s_i = 1$ for all items i, we obtain the usual recall measure. Given that complete data is unavailable (at low cost), p_{obs} in Eq. 11 cannot be calculated in practice. For this reason, we now examine different choices for p_{obs} and their effects on recall in Eq. 12.

In the first scenario, p_{obs} may take only two values: it is small for unpopular items, and large for popular items. If the ratio of these two values approaches infinity, this results in $s_i \to 0$ for popular items. Removing the relevant ratings of the most popular items from the test set is indeed common practice, e.g., in [3]. In the second scenario, we consider the case where p_{obs} is a smooth function of the items' popularities. We assume the polynomial relationship

$$p_{\text{obs}}(i) \propto \left(N^+_{\text{complete},i}\right)^\gamma \tag{13}$$

with $\gamma \in \mathbb{R}$. This is consistent with the power-law behavior of the observed relevant ratings (see Figure 1 a) in the sense that also the (unknown) complete data then follows a power-low distribution concerning the relevant ratings. This results in the stratification weights

$$s_i \propto 1/(N^+_{\text{obs},i})^{\gamma/(\gamma+1)}.$$

Note that the unknown proportionality factor cancels out in Eq. 12, so that it provides an unbiased estimate of the recall concerning the complete data for the correct polynomial degree γ (and assuming that there are no additional factors underlying the missing data mechanism). If $\gamma = 0$, the relevant ratings are missing at random; if $\gamma = 1$, the probability of observing relevant ratings increases *linearly* with item popularity ($N^+_{\text{complete},i}$). The extreme case when $\gamma \to \infty$ has several interesting properties: first, one observes in the available data only relevant ratings of the item with the largest popularity in the complete data, which does not agree with empirical evidence. Second, as $\gamma/(\gamma+1) \to 1$, the stratification weights s_i are inversely proportional to the number of observed relevant ratings $N^+_{\text{obs},i}$; this means that every item has the same weight in the recall-measure in Eq. 12, independent of its number of observed relevant ratings. This means that, once an item obtains its first relevant rating, it is weighted the same as all other items that may have obtained thousands of relevant ratings. This obviously entails low robustness against statistical noise as well as against manipulation and attacks. As this is the limiting case of γ, we nevertheless provide experimental results for this extreme scenario in Section 4. The (unknown) realistic case can be expected to be less extreme.

If the test set is a random subset of the observed data, then $N^+_{\text{obs},i}$ can be determined from the test set. Given that the training set is typically much larger than the test set, it might be more robust to determine $N^+_{\text{obs},i}$ based on all the available data, i.e., the training and test set combined. We use the latter in our experiments reported in the next section, but the former choice of $N^+_{\text{obs},i}$ leads to very similar results.

Finally, we define $\text{recall}(k) = \sum_u w^u \text{recall}_u(k)$ as the average recall over all users, with normalized weights, $\sum_u w^u = 1$, like in [17]. In our experiments, we choose $w^u \propto \sum_{i \in S_u^+} s_i$, as a generalization of the definition in [7, 17].

Obviously, stratification like in Eq. 12 carries over analogously to other measures, like ATOP [17] or the area under the ROC curve.

4. EXPERIMENTS

This section summarizes our results on the Netflix Prize data [1]. These data contain 17,770 movies and almost half a million users. About 100 million ratings are available. Ratings are observed for about 1% of all possible movie-

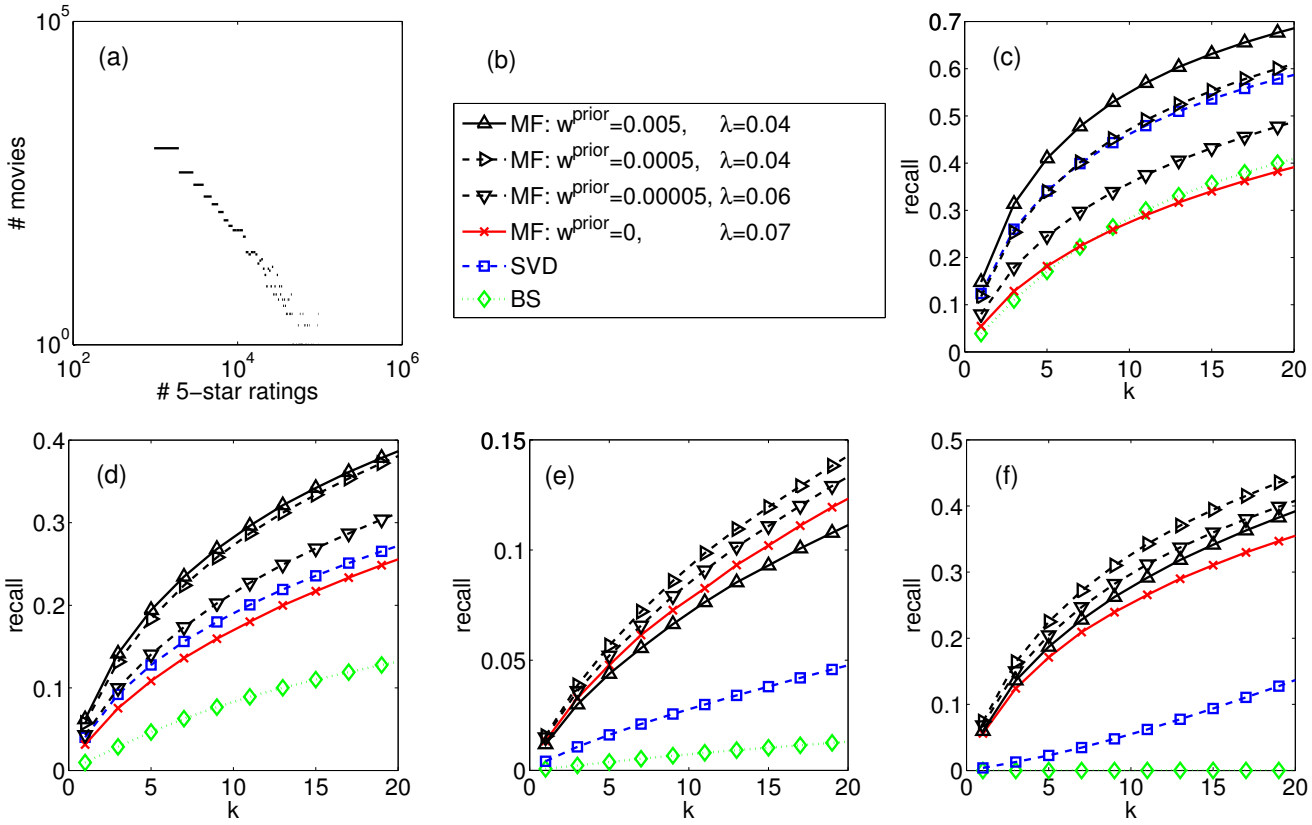

Figure 1: Netflix data [1]: (a) the number of relevant (i.e., 5-star) ratings per movie in the training data shows a close to power-law distribution (i.e., straight line in the log-log plot); (b) legend of models (see text for details); (c)-(f) show recall on probe set for different hypothetical missing-data mechanisms concerning the relvant (i.e., 5-star) ratings (while an arbitrary missing-data mechanism is allowed for the other ratings): (c) relevant ratings are missing at random ($\gamma = 0$ in Eq. 13); (d) relevant ratings observed with probability increasing linearly with item popularity ($\gamma = 1$ in Eq. 13); (e) unrealistic extreme case where $\gamma \to \infty$ in Eq. 13; (f) relevant ratings of the 10% most popular items removed. As a result, for all these missing-data mechanisms, recall test-results are improved by using an appropriate *small* prior weight $w^{\text{prior}} > 0$ during training, compared to the popular approach of ignoring the missing-data mechanism ($w^{\text{prior}} = 0$) during training.

user-pairs. The ratings take integer values from 1 (worst) to 5 (best). The provided data are already split into a training and a probe set. We removed the probe set from the provided data as to obtain our training set.

We consider 5-star ratings as *relevant* to a user (as defined above), and use the popularity-stratified recall, as outlined in Section 3, as our performance measures on the Netflix probe set. For all experiments, we chose rank $d = 50$ of our low-rank matrix factorization (MF) model (Eq. 1). In the following, we compared our MF model, trained with different prior weights $w^{\text{prior}} > 0$, against the popular MF approach that ignores the missing-data mechanism (i.e., $w^{\text{prior}} = 0$ in our notation). The latter achieved a root mean square error on the observed ratings in the probe set of 0.922. Additionally, we compared to singular value decomposition (SVD), where we used the *svds* function of Matlab, which implicitly imputes a zero value (with unit weight) for all missing ratings; and to the bestseller list (BS), which ranks the items according to the number of ratings in the training set. The values of tuning parameters in our training objective function (log posterior) in Eq. 9 are summarized in Figure 1 (b). Like in [17], we chose the prior rating value $r^{\text{prior}} = 2$ in Eq. 9.

Figures 1 (c)-(f) shows the performance of these models under different test scenarios, concerning our popularity-stratified recall measure for the practically important range of small k values. For computational efficiency, we computed recall by randomly sampling, for a user, 1,000 unrated items for each relevant rating in the test set, like in [3]. The only difference to the test procedure used in [7, 17] is that we sample from unrated items only, rather than from all items. This is more realistic. It also results in slightly higher recall values compared to the procedure used in [7, 17].

When comparing the different graphs, it is obvious that the performance of all the models depends on the (unknown) missing-data mechanism concerning the relevant ratings. In particular, when p_{obs} of relevant ratings is assumed to increase more rapidly with growing popularity ($N^{+}_{\text{complete},i}$),

the expected recall on the (unknown) complete data *decreases* for all models, cf. Figures 1 (c)→(d)→(e), and (c)→(f).

As p_{obs} of relevant ratings increases more rapidly with item popularity (compare Figures 1 (c)→(d)→(e)), the difference in recall among the various MF models decreases. Training with smaller but positive weights $w^{\text{prior}} > 0$ results in the best recall on the test set, even in the unrealistic extreme limit in Figure 1 (e). This suggests that, compared to the popular approach of ignoring the missing-data mechanism when training MF models, recall can be improved by using a small prior weight $w^{\text{prior}} > 0$; its value is upper bounded by the value that optimizes the (usual) recall measure on the test set, i.e., under the assumption that the relevant ratings are missing at random, like in [17].

The bestseller list (BS) and SVD perform surprisingly well if relevant ratings are missing at random, see Figure 1 (c), while the popular MF model with $w^{\text{prior}} = 0$ has low recall in comparison. This was also found in [17, 3]. BS and SVD perform rather poorly, however, if p_{obs} increases rapidly with item popularity, as shown in the extreme scenarios in Figure 1 (e) and (f). This suggests that not only BS, but also SVD tend to recommend items that are popular in the available training data. Their recommendations may hence result in a relatively low degree of serendipity or surprise, relative to our MF models trained with a small positive prior weight.

5. GENERALIZED APPROACH

This section outlines a generalization of the Bayesian approach given above. In our probabilistic approach, we consider the rating matrix R as a *matrix of random variables*. As each entry $R_{i,u}$ is a random variable (rather than a value), this naturally allows for possibly *multiple values* concerning each pair (i, u) in the data. This has several advantages over a *matrix of values*, which has typically been considered in the literature of recommender systems. After developing our generalized probabilistic framework, we outline three special cases / applications in Section 5.2.

Let the given data set be $D = \{r_{i,u,j}\}_{i,u,j}$, where $i = 1, ..., i_0$ is the index concerning items, $u = 1, ..., u_0$ is the index regarding users, and $j = 1, ...$ is the index over possibly multiple observed ratings for the same pair (i, u). The likelihood of the model in light of i.i.d. data reads

$$p(D|M) = \prod_{i,u,j} p(R_{iu} = r_{i,u,j}|M). \tag{14}$$

Assuming again a normal distribution of the ratings (with standard deviations $\sigma_{i,u,j}$, or equivalently weights $w_{i,u,j} = 1/(2\sigma^2_{i,u,j})$), the log likelihood of the model is

$$\begin{aligned} \log p(D|M) &= -\sum_{i,u}\sum_{j} w_{i,u,j}(r_{i,u,j} - M_{i,u})^2 + c_5 \\ &= -\sum_{i,u}\sum_{v} w_{i,u,v}(v - M_{i,u})^2 + c_5 \end{aligned} \tag{15}$$

where the second line is obtained by switching–for each pair (i, u)–from index j (over multiple ratings in the data) to the actual rating values v; the cumulative weight is $w_{i,u,v} = \sum_j w_{i,u,j} I_{r_{i,u,j}=v}$, where indicator function $I_{r_{i,u,j}=v} = 1$ if $r_{i,u,j} = v$ and 0 otherwise.

Combining this likelihood with the same kind of prior over the model parameters as in Eq. 2, we obtain the log posterior of our model:

$$\begin{aligned} \log p(M|D) = &-\sum_{i,u}\sum_{v} w_{i,u,v}(v - M_{i,u})^2 \\ &-\sum_{i}\frac{1}{2\sigma^2_{P,i}}\sum_{d} P^2_{i,d} - \sum_{u}\frac{1}{2\sigma^2_{Q,i}}\sum_{d} Q^2_{u,d} + c_6 \end{aligned} \tag{16}$$

The standard deviations $\sigma_{Q,i}$ and $\sigma_{P,i}$ may be chosen as to achieve the desired variant of regularization, as discussed in Section 2.2.

5.1 MAP Estimate of Model

For computational efficiency, we focus on optimizing the log posterior in Eq. 16. The *maximum-a-posteriori (MAP)* parameter estimate of our model, i.e., the MAP estimates $\hat{P}$ and $\hat{Q}$ of the matrices P and Q, can be determined by *alternating least squares*, which alternately optimizes one matrix while the other one is assumed fixed. Using the usual necessary condition for the optimum of Eq. 16, we equate its partial derivative to zero, and obtain the following update equation for each row i of $\hat{P}$ (for fixed $\hat{Q}$):

$$\hat{P}_{i,\cdot} = (\bar{v}_{i,\cdot} - r^{\text{offset}})\tilde{W}^{(i)}\hat{Q} \cdot \left[\hat{Q}^\top \tilde{W}^{(i)}\hat{Q} + \frac{1}{2\sigma^2_{P,i}} I\right]^{-1}, \tag{17}$$

where I denotes the identity matrix, and $\tilde{W}^{(i)}$ is a diagonal matrix containing the i^{th} row of the aggregate weight matrix with elements

$$w_{i,u} = \sum_{v} w_{i,u,v},$$

and $\bar{v}_{i,u}$ is the average rating value

$$\bar{v}_{i,u} = \left(\sum_{v} v w_{i,u,v}\right) / w_{i,u}.$$

Analogously, the update equation for each row u of $\hat{Q}$ is:

$$\hat{Q}_{u,\cdot} = (\bar{v}_{\cdot,u} - r^{\text{offset}})\tilde{W}^{(u)}\hat{P} \left[\hat{P}^\top \tilde{W}^{(u)}\hat{P} + \frac{1}{2\sigma^2_{Q,u}} I\right]^{-1}, \tag{18}$$

where $\tilde{W_u}$ is the diagonal matrix containing the u^{th} column of the aggregate weight matrix.

This derivation shows that optimizing Eq. 16 is equivalent to optimizing

$$\begin{aligned} \log p(M|D) = &-\sum_{i,u} w_{i,u}(\bar{v}_{i,u} - M_{i,u})^2 \\ &-\sum_{i}\frac{1}{2\sigma^2_{P,i}}\sum_{d} P^2_{i,d} - \sum_{u}\frac{1}{2\sigma^2_{Q,i}}\sum_{d} Q^2_{u,d} + c_6 \end{aligned} \tag{19}$$

where multiple rating values for an item-user pair are replaced by their mean value $\bar{v}_{i,u}$ and their aggregate weight $w_{i,u}$.

5.2 Applications

This generalized probabilistic approach subsumes several applications as special cases. In addition to the use of virtual ratings in the prior, as outlined in Section 2, we present two additional applications in the following.

5.2.1 Aggregate Recommendations

The universe of all items available for recommendation may have a structure that goes beyond a flat list. Items can

often be arranged in a hierarchical manner. For instance, songs may be grouped by artist, album, genre, etc. Possibly, there are several layers of hierarchy. Now let us consider the problem that data are available where users have rated individual songs, but the task is to recommend an artist to a user. This problem arises in several situations. For instance, the recommender system may want to suggest a concert to the user, based on the data on individual songs. Another scenario is the release of a new song by an artist: this cold start problem may be overcome by recommending the new song to users who like the artist.

The ratings matrix concerning songs and users can be used to construct a rating matrix regarding artists and users by aggregating the songs of each artist. As a user may have rated several songs of an artist, we now have possibly multiple rating values for an entry in the artist-user matrix. This is exactly the problem solved by our general approach in Section 5. Our framework also shows that, for each user, the rating of each artist can be determined as the weighted average of the ratings of his/her songs, and the weight is the sum of the weights of the songs, as one may have intuitively expected.

5.2.2 Recommendation of TV Shows

IP-TV is much more interactive than traditional TV. Concerning recommender systems, it allows one to collect information on the users' TV consumption. This can be used to learn preferences of users to TV programs, as to make accurate recommendations of TV shows. Unlike the previous applications described in this paper, we now use implicit feedback data (time spent watching a TV show) in place of the ratings. Our approach carries over immediately. This problem can be cast in our general probabilistic framework as follows: we divide the length of each show into $n_{\max}$ time intervals (of equal length), where $n_{\max}$ is the same large integer for all shows. We consider each time interval associated with a random experiment; a show hence comprises $n_{\max}$ repetition of the experiment. The random variable $R_{i,u}$ takes the value 1 if user u watched show i for a time-interval, and 0 otherwise. So, if user u watches $n_{i,u} \in \{1, ..., n_{\max}\}$ out of $n_{\max}$ intervals of show i, then we have $n_{i,u}$ observations of value 1 for the random variable $R_{i,u}$, and $n_{\max} - n_{i,u}$ observations of value 0; as shown in Section 5, these multiple observations can be substituted equivalently in our least-squares objective function in Eq. 16 by the aggregate weight $n_{\max}$ and the averaged value of the observations: $\bar{r}_{i,u} = n_{i,u}/n_{\max}$, i.e., the fraction of the show watched. In addition, if a show comprises several (e.g., weekly) episodes, then we assume that the above applies to each episode; the total weight of the show is then $t_{i,u} n_{\max}$, where $t_{i,u} \in \mathbb{N}$ is the number of episodes of show i watched by user u. Then the averaged observed value is $\bar{r}_{i,u} = \sum_{j=1}^{t_{i,u}} \bar{r}_{i,u,j}/t_{i,u}$, where $\bar{r}_{i,u,j}$ is the fraction of episode j watched (analogous to above). This results in the log likelihood (with an irrelevant constant c_7 in our optimization problem):

$$\log p(D|M) \quad = \quad -n_{\max} \sum_{(i,u)\in S} t_{i,u}(\bar{r}_{i,u} - M_{i,u})^2 + c_7,$$

where the set S contains all pairs (i, u) with shows i that have been *watched at least partially* by user u. Concerning all shows, we additionally incorporate background knowledge that users tend to not like shows with some small confidence / weight. This weight is small, as it can also be interpreted as the variance of our prior, which is large as there are many reasons for not watching a show. Analogously to Section 2, we use virtual observations of value 0, with weight w^{prior}. This results in the log likelihood in light of the virtual data points D^{prior}:

$$\log p(D^{\text{prior}}|M) \quad = \quad -n_{\max} w^{\text{prior}} \sum_{\text{all } (i,u)} (0 - M_{i,u})^2 + c_8$$

Combining these two likelihoods, together with the prior over the model parameters in Eq. 2 (first variant), we obtain the log posterior

$$\begin{aligned} &\log p(M|D, D^{\text{prior}}) \propto \\ &- \sum_{(i,u)\in S} t_{i,u}(\bar{r}_{i,u} - M_{i,u})^2 - w^{\text{prior}} \sum_{\text{all } (i,u)} (0 - M_{i,u})^2 \\ &-\lambda \sum_d \left[\sum_i P_{i,d}^2 + \sum_u Q_{u,d}^2 \right] + c_9, \end{aligned} \tag{20}$$

where we replaced the standard deviations in the prior by $\lambda \in \mathbb{R}$; the proportionality is due to omitting $n_{\max}$, which is an irrelevant constant when optimizing the posterior; c_9 is an irrelevant additive constant in our optimization problem.

In [5], the following objective function was experimentally found to result in the best recommendation accuracy from among several variants (re-written, but equivalent to Eq. 3 in [5]):

$$\begin{aligned} &\sum_{(i,u)\in S} (1 + \alpha n_{i,u}^{\text{t}})(1 - M_{i,u})^2 + \sum_{(i,u)\notin S} (0 - M_{i,u})^2 \\ &+\lambda \sum_d \left[\sum_i P_{i,d}^2 + \sum_u Q_{u,d}^2 \right], \end{aligned} \tag{21}$$

where $n_{i,u}^{\text{t}} = \sum_j n_{i,u,j}$ is the total time spent by user u watching show i (including all episodes j); S denotes again the set of pairs (i, u) where show i is at least partially watched by user u; α takes essentially the role of w^{prior}.

Interestingly, their objective function is similar to ours. The main difference, however, is in the least squares term, where we fit our model to the fraction $\bar{r}_{i,u}$ of the show watched, while the indicator value 1 is used in [5]. We attribute to this difference the fact that our model performs slightly better in our preliminary experiments on our (proprietary) IP-TV data [2], see below. Besides the experimental improvement, our theoretical framework also provides a clear understanding of the assumptions underlying our approach, while the approach in [5] appears to be found experimentally in a somewhat ad-hoc manner.

Preliminary Experiments: We used a (proprietary) IP-TV data set [2] concerning TV consumption of N =25,777 different shows by 14,731 users (living in 6,423 households) in the UK over a 6 month period in 2008/2009 (see also [15] for a more detailed description). In our collaborative filtering approach, we used only implicit feedback data concerning TV consumption (user ID, program ID, the length of the program and the time the user spent on this program), and ignored the available explicit user profiles and content information for simplicity. We randomly split these data into a training and a disjoint test set, with 10 shows per user assigned to the test set. In the test set, we considered shows interesting or relevant to users if they watched at least

model	$k' = 1\%$	$k' = 2\%$	$k' = 3\%$	$k' = 4\%$	$k' = 5\%$
ours	0.671	0.763	0.819	0.856	0.882
[5]	0.624	0.723	0.785	0.828	0.857
Nbr	0.547	0.642	0.704	0.760	0.804

Table 1: Recall(k') test results on IP-TV data.

80 % of them. We used these shows to evaluate the recommender systems w.r.t. the performance measures *recall* (as defined in Section 3 with $\gamma = 0$). Table 1 summarizes our preliminary results in terms of recall(k'), where $k' = k/N$ is normalized regarding the number N of available shows. Again, we used rank $d = 50$ for the matrix factorization models. We find that our new approach (with $\lambda = 0.005$) and the approach in [5] (with $\lambda = 0.02$) give similar results, compared to the neighborhood model (Nbr), $s_{ij} = \frac{r_i r'_j}{\|r_i\| \|r_j\|}$ and $\hat{r}_{iu} = \sum s_{ij} r_{uj}$, which was also used in [5] for comparison. Concerning recall, small values of k' are particularly important in practice, as only a small number of shows can be recommended to a user; in this regime, our new approach seems to perform better than the approach of [5]. We are currently running more refined experiments to confirm this finding.

6. CONCLUSIONS

This paper provides three contributions. First, we outlined a Bayesian framework that naturally allowed us to insert background knowledge concerning the missing-data mechanism underlying the observed rating data. The obtained log posterior probability of the model is very similar to the training objective function outlined in [17].

In our second contribution, we conducted experiments where we considered several hypothetical missing-data mechanisms underlying the observed real-world data. Given that the true missing-data mechanism is unknown in the absence of ground truth data, this sensitivity analysis provided valuable information. Our key insight based on these experiments is that the top-k hit-rate or recall can be improved considerably by training recommender systems with an appropriately chosen *small* positive prior weight concerning background knowledge on the missing-data mechanism. This is in contrast to the popular approach in the literature, which only considers observed ratings.

As third contribution, we provided a generalized probabilistic framework for factorizing a user-item-matrix that possibly has *multiple observed rating values* associated with each user-item pair. We discussed three important special cases / applications: besides training a top-k recommender system by using virtual data points, we outlined how ratings can be aggregated when items are grouped in a hierarchical manner rather than in a flat list, and how recommendations can be made using this hierarchical structure. Additionally, this framework enabled us to derive the training objective function for a recommender system on sequential data concerning TV consumption. This derivation not only provides a clear understanding of the assumptions underlying the training objective function, but also led to improvements in the top-k hit rate over state-of-the-art approaches in our preliminary experiments.

Acknowledgements

We are greatly indebted to Tin Ho for her encouragement and support of this work. We are also very grateful to the anonymous reviewers for their valuable feedback.

7. REFERENCES

[1] J. Bennet and S. Lanning. The Netflix Prize. In *Workshop at SIGKDD-07, ACM Conference on Knowledge Discovery and Data Mining*, 2007.

[2] BARB: Broadcaster Audience Research Board. http://www.barb.co.uk.

[3] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-N recommendation tasks. In *ACM Conference on Recommender Systems*, 2010.

[4] S. Funk. Netflix update: Try this at home, 2006. `http://sifter.org/ simon/journal/20061211.html`.

[5] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, 2008.

[6] R. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries, 2009. arXiv:0906.2027.

[7] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, 2008.

[8] B. Marlin and R. Zemel. Collaborative prediction and ranking with non-random missing data. In *ACM Conference on Recommender Systems (RecSys)*, 2009.

[9] B. Marlin, R. Zemel, S. Roweis, and M. Slaney. Collaborative filtering and the missing at random assumption. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

[10] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, 2008.

[11] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDDCup*, 2007.

[12] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2008.

[13] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *ICML*, 2007.

[14] R. Salakhutdinov and N. Srebro. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm, 2010. arXiv:1002.2780.

[15] C. Senot, D. Kostadinov, M. Bouzid, J. Picault, A. Aghasaryan, and C. Bernier. Analysis of strategies for building group profiles. In *Conference on User Modeling, Adaption and Personalization (UMAP)*, 2010.

[16] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *ICML*, pages 720–7, 2003.

[17] H. Steck. Training and testing of recommender systems on data missing not at random. In *KDD*, 2010.

[18] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.