# Training and Testing of Recommender Systems on Data Missing Not at Random

Harald Steck
Bell Labs, Alcatel-Lucent
600 Mountain Ave
Murray Hill, NJ 07974, USA
Harald.Steck@alcatel-lucent.com

## ABSTRACT

Users typically rate only a small fraction of all available items. We show that the absence of ratings carries useful information for improving the top-$k$ hit rate concerning *all* items, a natural accuracy measure for recommendations. As to *test* recommender systems, we present two performance measures that can be estimated, under mild assumptions, without bias from data even when ratings are *missing not at random (MNAR)*. As to achieve optimal test results, we present appropriate surrogate objective functions for efficient *training* on MNAR data. Their main property is to account for *all* ratings–whether observed or missing in the data. Concerning the top-$k$ hit rate on test data, our experiments indicate dramatic improvements over even sophisticated methods that are optimized on observed ratings only.

## Categories and Subject Descriptors

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

## General Terms

Algorithms

## Keywords

Recommender Systems

## 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, and we consider the *top-k hit rate*–based on *all* items–as the natural accuracy measure for recommender systems, as only a few out of *all* items can be recommended to a user in practice (see Section 2 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.

The *root mean square 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, 7, 8, 13, 15]). 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. This is our motivation for using the top-$k$ hit rate as to assess the ranking of *all* items, as it can deal with observed and missing ratings (see Section 2 for details).

In support of the validity to use observed ratings only, the experiments on the Netflix data in [8] show that models with better RMSE on the *observed* ratings achieve a better top-$k$ hit rate on *all* items.

When using observed ratings only, the implicit underlying assumption is, however, that the ratings in the available data are *missing at random. Missing (completely) at random* and *missing-data mechanisms* in general are outlined in [10, 14]. *Missing at random* means in the context of recommender systems that the probability of a rating to be missing does not depend on its *value.* If data are *missing at random*, using only the observed data for statistical analysis yields 'correct' results (e.g., unbiased maximum-likelihood parameter estimates, and predictions) [10, 14]. In contrast, if the data are *missing not at random (MNAR)*, the missing data mechanism cannot be ignored in general, and has to be modeled precisely as to obtain correct results [10, 14]. In other words, if the missing data mechanism is ignored, the resulting recommendation accuracy may be degraded significantly.

Recent work by Marlin et al. [11, 12] provided evidence that the data typically used for training and testing recommender systems are *MNAR*: their histograms of the distribu-

*KDD'10,* July 25–28, 2010, Washington, DC, USA.
Copyright 2010 ACM 978-1-4503-0055-1/10/07 ...$10.00.

**Table 1: Simplistic Example for ratings missing not at random (MNAR): test data where users rated only what they liked or knew.**

| | | users | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | horror fans | | | | romance lovers | | | |
| | h | 5 | | 5 | 5 | | | | |
| m | o | 5 | 5 | | | | | | |
| o | r | | 5 | | 5 | | | | |
| v | . | 5 | | 5 | 5 | | | | |
| i | r | | | | | 5 | 5 | | 5 |
| e | o | | | | | | 5 | 5 | 5 |
| s | m | | | | | 5 | | 5 | |
| | . | | | | | | 5 | 5 | 5 |

tion of ratings in the Yahoo!LaunchCast data show that *low* ratings are much more likely to be missing from the observed data than *high* ratings (see Fig. 2 in [11]). This may be a consequence of the fact that users are *free to choose which items to rate*. This kind of data is inexpensive to collect, and typically used for training and testing recommender systems (prior to conducting more costly field studies).

The effect of ratings *missing not at random (MNAR)* is illustrated by the following simplistic example, see Table 1: assume that the items comprise horror and romance movies; the romance-lovers assign the highest rating of 5 stars to romance movies and choose not to rate any of the horror movies, while the horror-fans do the opposite. The missing data mechanism is to rate only movies the users like or know. Let us further assume a (useless) recommender system that predicts a 5-star rating for all movies and users. When selecting a movie with the highest predicted rating (ties are broken at random), a horror or romance movie is recommended with equal probability to each user. While this is obviously a useless recommender system, it achieves perfect results on the *observed* ratings in our test data in Table 1–for any performance measure, including RMSE, mean absolute error (MAE), nDCG, expected reciprocal rank (ERR), mean average precision (MAP), etc. The reason is that the test data are *missing not at random (MNAR)*, as they do not contain any test cases for the combinations (horror movie, romance lover) and (romance movie, horror fan). The solution is to consider, for each user, the predictions / rankings of *all* items–*whether their ratings are observed or missing.* This is the scenario of a real-world recommender system, which has to pick a few items from among all available ones.

Many popular performance measures cannot readily deal with missing ratings. In this paper, we adopt the *top-k hit rate* as a natural measure for assessing the accuracy of recommender systems. In Section 2, we first define the top-$k$ hit rate, TOPK, and present a second, related performance measures, named ATOP; both can be estimated without bias on *MNAR data* under mild assumptions. While these measures are computationally tractable for testing recommender systems, they are too costly for training. We thus resort to surrogate objective functions that are computationally efficient (Section 3). In Section 4, we take a collaborative filtering approach, and present the update equations for training a matrix factorization model w.r.t. our new objective functions. Our experiments on publically available data in Section 6 show that it is essential to account for the fact that the data are MNAR, and we obtain a significantly improved top-$k$ hit rate. Section 7 summarizes our conclusions and future work.

## 2. TEST PERFORMANCE MEASURES

We consider the top-$k$ hit rate as a natural performance measure for recommender systems in practice, as only a small number of items can be recommended to a user. Ideally, each of the recommended items is *relevant* to a user.

**Definition ('relevant'):** *An item is relevant to a user if s/he finds this item 'appealing' or 'interesting'. Different items may be relevant to different users (personalization). In our notation, a quantity with the superscript '+' refers to relevant items. As to make this general definition more concrete, concerning the data in our experiments, we consider items with the highest (5 star) rating as* relevant *to a user. In case of continuous-valued ratings, one may regard the values above an appropriate threshold (possibly different for each user) as relevant.*

### 2.1 (Unknown) Complete Data

In this section, we assume that *complete* data are available, i.e., the ratings of *all* items by *all* users. This allows us to define the performance measures we ideally would like to compute. For a chosen cut-off value $k' \in \mathbb{N}$, the top-k hit rate may be defined in two alternative ways:

- precision: $N_u^{+,k'}/k'$,
- recall: $N_u^{+,k'}/N_u^+$,

where $N_u^{+,k'}$ is the number of items that are *relevant* to user $u$, and which made it into the top $k'$ items (out of all items) based on the recommender system's predictions; and $N_u^+$ denotes the number of items relevant to user $u$. We make the following three observations: (1) Precision and recall can both be calculated from *complete* data, i.e., if the ratings of all $N$ items are known. (2) Typically, however, most ratings are unobserved: in contrast to precision, recall can be estimated from MNAR data without bias under mild assumptions (see next section). (3) When comparing different recommender systems to each other on *fixed data and fixed k'*, then precision and recall are proportional to each other, with a (user-dependent) proportionality-factor $k'/N_u^+$. In other words, the recommender system with the larger recall also has the larger precision.

In place of the integer-valued $k'$ as threshold of the top-$k$ hit rate, we will use a normalized $k \in [0, 1]$ in the remainder of this paper: $k = (k' - 1)/(N - 1)$ determines the *fraction* of items relative to the total number $N$ of items.

For the reasons outlined above, we define the top-$k$ hit rate as recall ($\propto$ precision, for fixed data and $k$):

$$\text{TOPK}_u(k) = \frac{N_u^{+,k}}{N_u^+}, \tag{1}$$

where $N_u^+$ denotes all items relevant to user $u$, and $N_u^{+,k}$ counts the relevant ones in the top $k$, as above. Its maximum value (achieved for perfect recommendations) is $k'/N_u^+ < 1$ if $k' < N_u^+$, and equal to 1 otherwise. The average over all users $u$ is given by

$$\text{TOPK}(k) = \sum_u w^u \text{TOPK}_u(k), \tag{2}$$

with user-weights $w^u$, which are normalized as follows: $\sum_u w^u = 1$.

The following two properties of Eq. 1 may be considered a disadvantage: (1) it depends on the chosen value of $k$, and (2) it has a hard cut-off for the top $k$ items, and ignores the ordering (position bias) of relevant items that are ranked either below or above the cut-off. Both can be overcome by the *Area under the* $\text{TOPK}_u$*-curve (*$\text{ATOP}_u$*)* for user $u$. We define it as follows :

$$\text{ATOP}_u = \int_0^1 \text{TOPK}_u(k)\, dk. \tag{3}$$

Given that $\text{TOPK}_u(k)$ is a monotone increasing function of $k \in [0, 1]$, we can alternatively consider $k$ as a function of $\text{TOPK}_u$, and integrate along $\text{TOPK}_u$ instead: $\text{ATOP}_u = \int_0^1 (1-k)\, d\text{TOPK}_u$. From complete data, the latter can be calculated (as a non-parametric estimate) as follows: for fixed user $u$, *all* items $i$ are ranked according to their *predicted* rating $\hat{R}_{i,u}$: item $i$ with the highest predicted rating is assigned the largest rank $\text{Rank}_{u,i} = N$, and the item with the lowest predicted rating obtains $\text{Rank}_{u,i} = 1$.[1] We define the normalized rank $\text{Nrank}_{u,i} = (\text{Rank}_{u,i} - 1)/(N - 1) \in [0, 1]$. Let $S_u^+$ denote the set of relevant items for user $u$. Then the non-parametric estimate of the $\text{TOPK}_u$-curve, based on these ranked items, increases by $1/N_u^+$ where $\text{Nrank}^+_{u,i} = 1-k$ for each $i \in S_u^+$, and is constant everywhere else. For this reason, we have:

$$\text{ATOP}_u = \frac{1}{N_u^+} \sum_{i \in S_u^+} \text{Nrank}^+_{u,i} = \langle \text{Nrank}^+_{u,i} \rangle_{i \in S_u^+}, \tag{4}$$

where $\langle \cdot \rangle$ denotes the average. The area under the $\text{TOPK}_u$-curve equals the *average normalized rank of the relevant items.* This also shows that the maximum value of $\text{ATOP}_u$ is $1 - (N_u^+ - 1)/(N-1)/2$, where $N_u^+$ in the *complete* data is typically unknown. Rather than the ordering of all items, often the ordering above a small cut-off value is of interest; it is straightforward to define a truncated version of ATOP that captures the area under the initial part of the *TOPK* curve only. With the same weights for the users as above, the average area under the $\text{TOPK}_u$-curves is defined as

$$\text{ATOP} = \sum_u w^u \text{ATOP}_u. \tag{5}$$

## 2.2 Observed MNAR Data

In this section, we show that the performance measures outlined above, TOPK and ATOP, can be estimated from available *MNAR* data under mild assumptions. The motivation for our approach is as follows. In the one extreme, the ratings are indeed missing at random, and the missing data mechanism can be ignored. For a given data set, one cannot generally expect this (convenient) assumption to hold. In the other extreme, there is a (non-trivial) missing data mechanism, and one is able to model it exactly. Unfortunately, the (exact) missing data mechanism is typically unknown. Even if it were known for one particular data set, one could not expect it to be valid for other data sets in general. But there is also some middle ground between these two extremes, which is the focus of this paper. Inspired by *general* properties of *various* publically available data sets for recommender systems, we make the following

**Assumptions (User Model):**

1. *We assume that the* relevant[2] *rating values are missing at random in the observed data.*
2. *Concerning the other rating values, we allow for an arbitrary missing data mechanism, as long as they are missing with a higher probability than the* relevant *rating values do.*

Several comments are in order. Typically, the number of *relevant* ratings is only a tiny fraction (a few percent) of *all* ratings in the complete data, e.g., see Fig. 2 in [11]. Our first assumption is hence a major relaxation compared to the assumption that *all* ratings are missing at random. Note, however, as every assumption is a simplification of the real world, it may not hold exactly for a given data set in reality; but it may serve as a reasonable approximation across a large number of data sets. This is supported by the large improvements we obtained in our experiments. Our second assumption can be expected to hold for many data sets, including the Yahoo!LaunchCast data, see Fig. 2 in [11]. Note that this also implies that the average unobserved rating is lower than the average observed rating.

Let $S_u^{+,\text{obs}}$ denote the set of observed relevant items for user $u$, and $N_u^{+,\text{obs}} = |S_u^{+,\text{obs}}|$ their number; let $N_u^{+,\text{obs},k}$ be the number of relevant items in the top $k$. $\text{Nrank}^{+,\text{obs}}_{u,i}$ is calculated for the observed relevant items w.r.t. to *all* items.

**Theorem:** *Under the above two assumptions, the measures*

$$\text{TOPK}_u^{\text{obs}}(k) = \frac{N_u^{+,\text{obs},k}}{N_u^{+,\text{obs}}}, \tag{6}$$

$$\text{ATOP}_u^{\text{obs}} = \langle \text{Nrank}^{+,\text{obs}}{}_{u,i} \rangle_{i \in S_u^{+,\text{obs}}}, \tag{7}$$

*computed from the* observed MNAR *data, provide unbiased estimates of the measures* $\text{TOPK}_u(k)$ *and* $\text{ATOP}_u$*, respectively. The averages* $\text{TOPK}^{\text{obs}}(k) = \sum_u w^u \text{TOPK}_u^{\text{obs}}(k)$ *and* $\text{ATOP}^{\text{obs}} = \sum_u w^u \text{ATOP}_u^{\text{obs}}$ *are unbiased estimators of the corresponding measures evaluated on complete data.*

**Proof:** The ranks of the observed relevant items are determined by using *all* the $N$ items, whether their ratings are observed or missing. The ranks of the observed relevant items are identical for complete and MNAR data. With the assumption that the relevant items are missing at random, it is apparent that their average rank can be estimated without bias. The other claims follow analogously from Assumption 1, and the well-known fact that estimates are unbiased from data missing at random. □

In summary, $\text{TOPK}_u^{\text{obs}}(k)$ is (1) the unbiased estimate of recall (under the outlined assumptions); and (2) proportional to precision (with unknown user-dependent proportionality-factor $k'/N_u^+$) when comparing recommender systems on fixed data and fixed $k$. In other words, the recommender system with larger recall also has larger precision on fixed data and fixed $k$. Analogously, the weighted average over all users, $\text{TOPK}^{\text{obs}}(k) = \sum_u w^u \text{TOPK}_u^{\text{obs}}(k)$ is (1) an unbiased estimate of the average recall (weighted by $w^u$), and (2) proportional to the precision averaged over the users.

[1] Note that ranks are often defined the other way around in the literature: the smallest rank-value is best. Our definition has the advantage that it can also be viewed from the perspective of the Borda count.

[2] See Definition of 'relevant' above.

$\text{ATOP}_u^{\text{obs}}$ can be viewed (1) as the unbiased estimate of the area under the $\text{TOPK}_u(k)$-curve; and (2) as the average normalized rank of the relevant items. Both views also apply to the weighted average over the users, $\text{ATOP}^{\text{obs}} = \sum_u w^u \text{ATOP}_u^{\text{obs}}$. Note that ATOP captures position bias (within the entire list of items), but in a quantitatively different way than nDCG, ERR, or MAP. Note that the latter measures cannot be readily evaluated in the presence of missing ratings.

## 2.3 Practical Computation of ATOP Measure

First, our definition of the normalized rank, $\text{Nrank}_{u,i} = (\text{Rank}_{u,i} - 1)/(N-1) \in [0,1]$ (above Eq. 4), can also be viewed as follows: for a given user $u$ and item $i$, it is the fraction of *remaining* items (i.e., all items except for item $i$) that are ranked *lower* than item $i$. One can thus calculate $\text{Nrank}_{u,i}^+$ for each user-item pair $(i,u)$ with an observed relevant rating in the test set *individually*; in other words, one can iterate through the list of relevant pairs $(i,u)$ in the test set in any order and calculate the normalized rank $\text{Nrank}_{u,i}^+$ for *one* pair $(i,u)$ at a time. This may be more convenient than calculating the ranks of all relevant items simultaneously for each user $u$. The ATOP measure can then be calculated as the average of all these normalized ranks; and the $\text{TOPK}(k)$ measure as the fraction of those normalized ranks with $\text{Nrank}_{i,u} \geq 1-k$; note that this implicitly determines the weights $w^u$ over the users: $w^u \propto N_u^{+,\text{obs}}$, i.e., the number of observed relevant ratings of user $u$. We use these weights in our experiments, so that our measure becomes identical to the one used in [8], and thus allows for a fair comparison of the experimental results.

Second, one may rank a relevant pair $(i,u)$ w.r.t. to a *random subsample* of size $\tilde{N}-1$ of the remaining items $(N-1)$, as suggested in [8]. This results in computationally efficient estimation of the $\text{TOPK}(k)$ and ATOP measures. Note that this may result in a small positive bias, especially for $\text{TOPK}(k)$ at *small* values of $k$; it is hence important to use the same value of $\tilde{N}$ for fair comparisons (of course, bias-correction is an alternative).

## 2.4 Area under the ROC Curve

As an aside, we like to note that the $\text{TOPK}(k)$ and ATOP measures are closely related to the *Receiver Operating Characteristics (ROC) Curve*, which originates from signal processing and has since been adopted by the machine learning community for assessing classification performance and the quality of rankings. The latter is due to the equivalence of the *Area Under the ROC Curve (AUC)* and the *Wilcoxon-Mann-Whitney (WMW)* statistic. While the definition of WMW is concerned with the correct ranking of all possible pairs of items, it was shown in [5, 19] that WMW or AUC can be evaluated in time linear in $N$ given the ranks of the relevant items. It reads for user $u$:

$$\text{AUC}_u = \frac{1}{N_u^+ N_u^-}\left[\left(\sum_{i=1}^{N_u^+} \text{Rank}_{i,u}^+\right) + \binom{N_u^+ + 1}{2}\right], \quad (8)$$

where $N_u^- = N - N_u^+$ is the number of irrelevant items for user $u$. Note that we have to assume complete data here, as $N_u^+$ and $N_u^-$ would be unknown otherwise. It follows immediately:

$$\begin{aligned} \text{AUC}_u &= \left(1 + \frac{N_u^+ - 1}{N_u^-}\right)\text{ATOP}_u + \frac{N_u^+ + 3}{2N_u^-} \\ &= \text{ATOP}_u + \mathcal{O}\left(\frac{N_u^+}{N_u^-}\right). \end{aligned} \quad (9)$$

This shows that the difference between AUC and ATOP becomes negligible if $N_u^+ \ll N$. This is the case in the Yahoo!LaunchCast data set, where $N_u^+/N \approx 3\%$ in the survey data, see Figure 2 in [11]. A similarly small fraction may be expected for the other commonly used data sets. Under this condition, AUC can be computed from *MNAR* data in good approximation; the TOPK curve can be expected to be close to the ROC curve.

# 3. TRAINING OBJECTIVE-FUNCTION

Good performance of recommender systems w.r.t. our TOPK and ATOP measures on a held-out test set can only be expected when they have been *trained on an appropriate objective function*. Ideally, one would use the same objective function (plus some regularization) for training as is later used as performance measure in testing. While our TOPK and ATOP measures are computationally tractable for testing, they are unfortunately computationally very expensive to optimize during training, like other rank-based measures. For this reason, we resort to an appropriate surrogate measure for computationally efficient training. As motivated above, *key is to consider the ranking of* all *items–whether their ratings are observed or missing in the data.* This key property is captured by our `AllRank` objective function, of which we present three versions: they are based on (logistic) regression for computational efficiency. They may also be viewed as the minimal modification necessary to extend the well-studied approach based on RMSE from *observed* to *all* ratings. Given the large improvement achieved w.r.t. TOPK and ATOP in our experiments (compared to training on the observed ratings only), further improvements from using more sophisticated versions of `AllRank` might be small in comparison.

First, note that the $\text{ATOP}^{\text{obs}}$-test-measure is the average rank of the observed relevant items w.r.t. *all* items. This suggests to cast the learning task as a binary classification problem: the observed relevant items are assigned to class 1, and the other items (with lower or missing ratings) are assigned to class 0. In place of the rating matrix $R$, we use the corresponding binary matrix $Y$.

Given that $\text{ATOP}^{\text{obs}} \approx \text{AUC}^{\text{obs}}$ for $N^+ \ll N$ (see above), we can motivate the likelihood of logistic regression as appropriate objective function for training: in [17], AUC was related to the so-called hinge loss, which is typically used for training support vector machines (SVM). The key property of the hinge loss is that the loss increases *linearly* with the distance from the decision boundary for misclassified data points. The likelihood of logistic regression shows the same linear behavior in good approximation for reasonably large distances from the decision boundary. Note that the only difference between logistic regression and Fisher's linear discriminant analysis (LDA) is that the latter additionally assumes that the input variables follow a Gaussian distribution. Both approaches can thus be expected to optimize AUC approximately if the data fulfill their underlying assumptions. Given the large data set size, training an SVM is computationally very costly compared to logistic regression. For this reason, our first variant, `AllRank`-Binary-Logistic, uses the penalized log likelihood of logistic regression,

$$\sum_{\text{all } u}\sum_{\text{all } i} W_{i,u}\left\{Y_{i,u}\hat{Y}_{i,u} - \log\left(1+e^{\hat{Y}_{i,u}}\right) - \frac{1}{2}||\theta_{i,u}||_2^2\right\}, \tag{10}$$

where $\hat{Y}_{i,u} \in [0,1]$ denotes the prediction. The sum extends over *all* item-user pairs $(i,u)$ (with observed and unobserved ratings). The weights $W_{i,u}$ take only two values for simplicity: $W_{i,u} = 1$ for observed relevant ratings (class 1), and $W_{i,u} = w_m$ if the rating of pair $(i,u)$ does not have a relevant value or if the rating is missing (class 0); $w_m$ is a tuning parameter in this objective function, besides the regularization factor $\lambda$ in the ridge penalty term involving the L2 norm of the model parameters $\theta_{i,u}$. These tuning parameters are determined by cross-validation as to optimize the ATOP$^{\text{obs}}$-measure on the test set. The objective function in Eq. 10 has two major drawbacks: (1) when cast as a classification problem, all the observed ratings, except for the relevant ones, are ignored, which may result in a considerable loss of information; (2) maximizing Eq. 10 is computationally inefficient for large data sets, e.g., using the Newton-Raphson algorithm.

Given the large training set size, computational efficiency of optimizing Eq. 10 can be increased by using quadratic approximations as outlined in [16]. A computationally efficient approach can be obtained by replacing Eq. 10 with least squares for the binary matrix $Y$, which results in our second objective function, `AllRank`-Binary-Regression:

$$\sum_{\text{all } u}\sum_{\text{all } i} W_{i,u}\left\{\left(Y_{i,u} - \hat{Y}_{i,u}\right)^2 + \lambda||\theta_{i,u}||_2^2\right\}. \tag{11}$$

Finally, in our third objective function, `AllRank`-Regression, we use all observed rating values (i.e., not binarized), as this allows one to learn from gradual differences in the rating values. For all missing rating values, we impute the value $r_m$. This is the simplest approach to dealing with missing data; more importantly, though, this allows us to retain the sparsity of the original rating matrix, which is essential for efficient computations. We allow for a different weight-value for each rating value $R^{\text{obs}}_{i,u}$, and an additional weight $w_m$ for the missing ratings:

$$W_{i,u} = \begin{cases} w(R^{\text{obs}}_{i,u}) & \text{if } R^{\text{obs}}_{i,u} \text{ observed} \\ w_m & \text{otherwise} \end{cases}, \tag{12}$$

$$\sum_{\text{all } u}\sum_{\text{all } i} W_{i,u} \cdot \left\{\left(R^{\text{o\&i}}_{i,u} - \hat{R}_{i,u}\right)^2 + \lambda||\theta_{i,u}||_2^2\right\}, \tag{13}$$

where $R^{\text{o\&i}}_{i,u}$ are the observed and imputed ratings, while $\hat{R}_{i,u}$ are the predicted ratings.

## 4. MODEL TRAINING

We now take a collaborative filtering approach, and use a (basic) low-rank matrix-factorization model: the matrix of predicted ratings $\hat{R} \in \mathbb{R}^{i_0 \times u_0}$, where $i_0 = N$ denotes the number of items, and $u_0$ the number of users, is modeled as

$$\hat{R} = r_m + PQ^\top, \tag{14}$$

with matrices $P \in \mathbb{R}^{i_0 \times j_0}$ and $Q \in \mathbb{R}^{u_0 \times j_0}$, where $j_0 \ll i_0, u_0$ is a free parameter of the model and determines the rank. Note that we choose the offset $r_m \in \mathbb{R}$ in Eq. 14 to be equal to the imputed rating $r_m$. All entries in these matrices are considered independent parameters, with L2-norm for pair $(i,u)$:

$$||\theta_{i,u}||_2^2 = \sum_{j=1}^{j_0} P_{i,j}^2 + Q_{u,j}^2.$$

In the remainder of this section, we outline how our third objective function (see Eq. 13) can be optimized efficiently for this model.

### 4.1 Alternating Least Squares

This section outlines the update equations for learning $P$ and $Q$ for fixed tuning parameters $W_{i,u}, r_m$, and $\lambda$. Additionally, the tuning parameters have to be optimized via cross validation as to maximize ATOP$^{\text{obs}}$ on the test data.

Determining the values in the two matrices $P$ and $Q$ by minimizing Eq. 13 is a non-convex minimization problem [16]. Alternating least squares is an elegant and efficient method for finding a (close to) minimum solution of Eq. 13 by gradient descent. It can also be used for incremental updates as new rating values arrive over time. At each step of this procedure, one of the two matrices $P$ and $Q$ is assumed fixed, which turns the update of the other one into a quadratic optimization problem that can be solved exactly.

For fixed $Q$, the matrix $P$ that minimizes Eq. 13 can be calculated using the usual necessary condition (equate gradient to zero), and solving for $P_{i,\cdot}$ for each item $i$. This results in the following update equation for each row $i$ of $P$:

$$P_{i,\cdot} = (R^{\text{o\&i}}_{i,\cdot} - r_m)\tilde{W}^{(i)}Q\left(Q^\top\tilde{W}^{(i)}Q + \lambda\text{tr}(\tilde{W}^{(i)})I\right)^{-1}, \tag{15}$$

where the dot in the index of a matrix refers to the vector of all entries; $\tilde{W}^{(i)} = \text{diag}(W_{i,\cdot}) \in \mathbb{R}^{u_0 \times u_0}$ is the diagonal matrix containing the $i^{\text{th}}$ row of weight matrix $W$; $I \in \mathbb{R}^{j_0 \times j_0}$ is the identity matrix. While the diagonal matrix $\tilde{W}^{(i)}$ may appear to be of computationally prohibitive size, Eq. 15 can be computed efficiently, as outlined in the next section.

Analogously, the update equation for each row $u$ of $Q$ is:

$$Q_{u,\cdot} = (R^{\text{o\&i}\,\top}_{\cdot,u} - r_m)\tilde{W}^{(u)}P\left(P^\top\tilde{W}^{(u)}P + \lambda\text{tr}(\tilde{W}^{(u)}) \cdot I\right)^{-1}, \tag{16}$$

where $\tilde{W}^{(u)} = \text{diag}(W_{\cdot,u}) \in \mathbb{R}^{i_0 \times i_0}$ is the diagonal matrix containing the $u^{\text{th}}$ column of the weight matrix $W$. The common random initialization is used to generate the matrices $P, Q$ of expected maximal rank $j_0$ at the beginning.

Apart from that, note that this approach lends itself to parallelization (each row can be computed independently of the other rows in either matrix). Moreover, this provides also–to a good approximation–an efficient incremental update mechanism if a new user or a new item is added, or if a few ratings concerning a user or an item changes: assuming that the addition of the new user-ratings does not affect the item-matrix $P$ notably, the corresponding row in the user-matrix $Q$ can be updated in one step according to Eq. 16; similarly for new user ratings.

## 4.2 Practical Computations

Update equation Eq. 15 (and Eq. 16 analogously) can be easily rewritten in a way that is computationally efficient. In practice, computations are only slightly more costly compared to the popular objective of optimizing RMSE on the *observed* ratings only.

First, because $R^{\text{o\&i}}_{i,u} - r_m = 0$ if the rating of pair $(i, u)$ is unobserved, we have

$$\begin{aligned} &(R^{\text{o\&i}}_{i,\cdot} - r_m)\tilde{W}^{(i)}Q \qquad (17)\\ &= (R^{\text{o\&i}}_{i,u\in S_i} - r_m)\left(\left(W^{\top}_{i,u\in S_i}\mathbb{1}(1,j_0)\right)\otimes Q_{u\in S_i,\cdot}\right), \end{aligned}$$

where $\mathbb{1}(1, j_0) \in \mathbb{R}^{1\times j_0}$ is a vector of ones, and $\otimes$ denotes the elementwise product of matrices. Note that this expression involves only the submatrices for the set of users $u$ who have rated item $i$, denoted by $S_i$. This is typically a very small subset of all users.

Second, because $W_{i,u} - w_m = 0$ if the rating of pair $(i, u)$ is missing, one can decompose:

$$\begin{aligned} &Q^{\top}\tilde{W}^{(i)}Q \qquad (18)\\ = \;\; & w_m Q^{\top}Q - w_m Q^{\top}_{u\in S_i,\cdot}Q_{u\in S_i,\cdot}\\ &+\left(Q^{\top}_{u\in S_i,\cdot}\otimes\left(\mathbb{1}(j_0,1)W_{i,u\in S_i}\right)\right)Q_{u\in S_i,\cdot} \end{aligned}$$

Note that the first and computationally most expensive term can be pre-computed at each step for all $u$, while the other two terms require only a submatrix of $Q$ concerning the users $u$ who have rated item $i$.

Third, the trace simplifies into a sum over the users $u$ who have rated item $i$ (rather than summing over all users):

$$\operatorname{tr}(\tilde{W}^{(i)}) = w_m u_0 + \sum_{u\in S_i}(w_{i,u} - w_m). \qquad (19)$$

# 5. RELATED WORK

Nearly all work on recommender systems is concerned with *explicit* feedback data (e.g., ratings assigned by users). Impressive progress has been made in recent years–and it is impossible to name all. While almost all work has focused on predicting the *rating values* accurately (e.g., [1, 4, 7, 15, 13, 8]), only very few papers (e.g., [18]) were concerned with optimizing the *ranking* of items. Despite their differences, they have in common that they were tested on the *observed ratings only*, and hence trained accordingly: optimizing the ranking or the distribution over rating values *conditioned on the fact whether the items had been rated by the user or not.* Such a conditional ranking or distribution by itself is only of limited value for real-world recommender systems, as the marginal distribution or ranking (i.e., without conditioning) is needed. To this end, the conditional distribution may be combined with the additional prediction on which items will be rated by the user, which seems a similarly challenging problem [9].

Even when *testing on observed ratings* only, prediction accuracy could be improved by *training* not only on the observed rating values but also by using the indicator matrix, which captures whether an item was rated by a user or not [13, 8, 15]. The conditional Restricted Boltzmann Machine learns the distribution conditioned on this indicator matrix, which was found to be beneficial for improving RMSE on observed ratings [15]. Its applicability to ranking *all* items is not immediately clear. While the matrix factorization models NSVD1, NSVD2 and SVD++ [13, 8] used the indicator matrix, the training objective function summed over the observed ratings only. The original motivation for using the indicator matrix was to reduce the effective number of free parameters in the matrix factorization model [13]. Moreover, the indicator matrix allows one to incorporate into the model the items that were rated in the test set (and hence have to be predicted), which was known in the Netflix competition beforehand, but does not seem applicable to a real-world recommendation scenario. Whether these models lead to further improvements when trained with our `AllRank` objective function, is subject to future work. Note that our paper illustrates the importance of accounting for missing ratings in the *training objective function* (rather than in our simple matrix-factorization *model*).

It makes sense to ignore missing ratings if one assumes that the ratings are *missing at random.* Recently, however, the validity of the *missing at random* assumption was questioned [11, 12]. This work outlines empirical evidence that the 'typical' data (i.e., where users are free to choose what to rate) provide a distorted distribution over the rating-values compared to the (usually unavailable) complete data: low ratings tend to be missing with increased probability. Their work developed multinomial mixture models to account for the missing data mechanism underlying the training data. While valuable, computational efficiency of this approach may need to be improved as to scale up to the Netflix data and other real-world data sets, which are typically much larger. Their models achieved significant improvements w.r.t. their testing procedure, which relied on having an approximation to the distribution of the complete data available (by means of the survey data). Such survey data, unfortunately, is typically not available (at low cost). Their work was based on the Yahoo!LaunchCast data set, which is rather small and not available to the general public. It is not obvious how their testing procedure could be modified for use on MNAR test data. Being able to *test* on (inexpensive) *MNAR data* is one of the main motivations of our work.

In the context of *implicit* feedback data (e.g., log files of TV consumption for each user), missing feedback was taken into account when training a matrix factorization model in [6]. This work points out that accounting for missing feedback is important only for *implicit* feedback data, while this is unnecessary for *explicit* feedback data. The reason is that implicit feedback provides an imbalanced picture, containing exclusively positive feedback (e.g., TV show watched by user), while lacking negative feedback (e.g., there are many reasons for not watching a TV show). In contrast, explicit feedback provides a balanced picture of a user's preferences, as it contains positive and negative feedback, according to [6]. A main point of our paper is that, even though explicit feedback data contain negative feedback, it is not sufficient; missing feedback tends to be mainly negative, and hence carries additional negative information that can be used. Apart from that, while [6] obtained best results on implicit feedback data by fitting the model to *binarized* target values (called preference in their paper), we found that it is best to use the *full range* of rating values when training on explicit feedback data in our experiments.

**Table 2: Test results on MovieLens data for different training objective functions.**

| objective function for training | $w_m$ | $r_m$ | $\lambda$ | RMSE (test) | ATOP (XV) | ATOP (test) |
|---|---|---|---|---|---|---|
| bestseller list 1: mean of observed ratings | - | - | - | 0.990 | 0.804 | 0.803 |
| bestseller list 2: count of observed ratings (any value) | - | - | - | - | 0.874 | 0.873 |
| bestseller list 3: count of observed 5-star ratings | - | - | - | - | 0.883 | 0.880 |
| observed ratings only, optimize $\lambda$ for RMSE (Eq. 13) | 0 | - | 0.06 | 0.901 | 0.863 | 0.861 |
| observed ratings only, optimize $\lambda$ for ATOP (Eq. 13) | 0 | - | 0.05 | 0.914 | 0.866 | 0.864 |
| `AllRank`-Regression (Eq. 13) | 0.05 | 2 | 0.05 | 1.218 | 0.933 | 0.933 |
| 'dense' SVD (Eq. 13) | 1 | 2 | 0.03 | 1.766 | 0.915 | 0.915 |
| `AllRank`-Binary-Logistic (Eq. 10) | 0.02 | 0 | 0.02 | - | 0.923 | 0.928 |
| `AllRank`-Binary-Regression (Eq. 11) | 0.01 | 0 | 0.03 | - | 0.919 | 0.924 |

# 6. EXPERIMENTAL RESULTS

This section summarizes our results on the MovieLens and Netflix data sets. Unfortunately, we do not have the Yahoo!LaunchCast data available for experiments. For all experiments, we chose rank $j_0 = 50$ of our low-rank matrix factorization model (Eq. 14). We consider 5-star ratings as *relevant*[3] to a user (see Definition above), and use the top-$k$ hit rate, TOPK, and the area under the top-$k$ hit rate curve, ATOP, as outlined above, as our performance measures on the test data. The standard deviation (std) of all reported numbers in the tables is less than 0.005.

Our results confirm the assumption that these data are MNAR. Training a matrix factorization model such that it optimizes our new objective functions, we obtain significantly better results w.r.t. the top-$k$ hit rate than have been reported in the literature for even sophisticated models that were trained to optimize the popular RMSE on the *observed* data only.

## 6.1 MovieLens Data

The MovieLens data [2] involves 3,900 movies and 6,040 users, and ratings are available for about 1 million movie-user-pairs. About 4% of all possible ratings are observed in this data set. The ratings take integer values from 1 (worst) to 5 (best). As we train a collaborative filtering model in this paper, we do not use the provided demographic and content information. We split this data set into a training set and a disjoint test set. As the time-stamps of the ratings are available, we assigned the last 5 ratings of each user to the test set, and the earlier ratings to the training set. We further split the test set randomly into two disjoint sets of equal size. The first test set ('XV') is used for cross-validation during training as to determine the tuning parameters in our objective function. The second test set is used as a truly held-out data set ('test') for final evaluation of the trained model w.r.t. the ATOP measure.

Table 2 shows the results concerning our ATOP measure on both test sets–XV and test. The small differences concerning these two test sets are due to the random split of the data and provide an estimate for the confidence intervals (std$\leq$ 0.005). They also show that there is no significant overfitting toward the XV set (used for tuning). Some approaches provide a *ranking* of the movies, so that RMSE does not have any meaning in this case.

Let us now discuss the results obtained for each approach.

First, we set the weights of all observed ratings to 1, while we set $w_m = 0$. This ignores missing ratings. Eq. 13 simplifies to the extremely popular RMSE measure used for training recommender systems. $\lambda$ remains the only free tuning parameter for training. We optimized it by cross validation on the first test set (XV) in two different ways: (1) to minimize RMSE, and (2) to maximize ATOP. In both cases, we obtained relatively similar results on the test set: relatively good results w.r.t. RMSE, but relatively poor ones w.r.t. ATOP, see Table 2.

We obtained a significant improvement regarding ATOP, however, when we took into account the missing ratings by allowing for $w_m > 0$. *This shows that ignoring the missing ratings in the training data is detrimental w.r.t.* ATOP.

Besides $w_m$, we also optimized the weights $w(r)$ for each of the observed rating values ($r = 1, .., 5$). We found that this does not significantly improve the test results w.r.t. ATOP, and thus used $w(r) = 1$ for all observed rating values.

Finally, for $w_m = 1$, each rating is weighted equally–whether observed or missing in the training data. This is identical to regular / dense *singular value decomposition (SVD)* (plus regularization). This approach had already been used successfully in the past, e.g., in latent semantic analysis / indexing [3]. Interestingly, this 'dense' SVD achieves better ATOP test results in Table 2 than the 'sparse' matrix approach ($w_m = 0$), which recently had became popular for the Netflix prize competition.

Moreover, we also cast the training problem as a binary classification task: the observed 5-star ratings vs. all other (observed and missing) ratings. As outlined in Section 3, we optimized the two training objective functions in Eqs. 10 and 11. The ATOP test results were slightly below our best results (see also Table 2). This may not be surprising, as these binary classification approaches completely ignored the information contained in the observed rating values 1,...,4. Interestingly, however, these ATOP test results were considerably better than the ones obtained by ignoring the missing ratings ($w_m = 0$), see Table 2.

A few comments on the optimal values of the tuning parameters for the MovieLens data are in order: we found the optimal value for the imputed rating value to be $r_m \approx 2$. This makes intuitive sense when the imputed value is interpreted as the average rating in the (unknown) complete data: it is considerably lower than the mean ($\approx$ 3.6) of the observed ratings (see also Assumption 2). The optimal value $w_m \approx 0.05$ of the weight assigned to each missing rating may appear very small. This value has to be seen in perspective to the 4% of observed ratings in this data set. With a weight

[3] Considering both 4 and 5 star ratings as relevant, experiments showed similar differences among the various approaches.

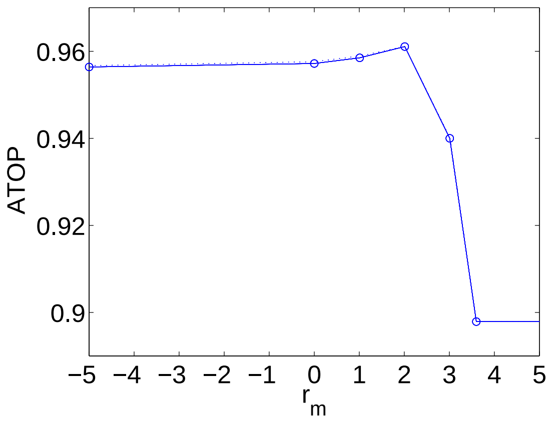

**Figure 1:** ATOP **measure on the Netflix data for different imputed ratings $r_m$; on held-out test set (solid) and on XV test set used for parameter tuning (dashed); the two curves are almost identical.**

of 1 assigned to each observed rating, $w_m \approx 0.05$ essentially means that all the missing ratings together have about the same weight as all the observed ratings together. This makes sense, as it is common practice in machine learning to convert very imbalanced training sets into somewhat balanced ones as to improve the trained model's test performance.

As a baseline model for (non-personalized) recommendations, we considered three variants of the bestseller list: the rank of each item is determined according to (1) the mean of the observed ratings (as in [8]), (2) the number of ratings (of any value), and (3) the number of 5-star ratings in the training data. Note that the 'test' and 'XV' sets are both held-out test sets for the bestseller lists. Table 2 illustrates two interesting results. First, among the three bestseller lists, the one that ignores the missing ratings (mean of observed ratings) is by far worse than the two bestseller lists that account for the missing values (by counting). Second, the latter two bestseller lists are competitive with the matrix factorization model that ignores missing ratings ($w_m = 0$) concerning ATOP.

We also considered optimizing the nDCG measure on the observed ratings in the training data, as is readily done by CofiRank[4] [18]. We used the provided configuration file, changed the hidden dimension to 100 and trained with nDCG@10 as in [18]. We obtained test results concerning ATOP and TOPK that were significantly worse than the bestseller lists; however, we did not have time yet to examine this issue in further detail.

## 6.2 Netflix Prize Data

The Netflix Prize data [1] 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-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 training set as to obtain our training set. A random split of the probe set provided us with two test sets of equal size, one is used for cross-validation during training (XV) and one for final testing (test), as before.

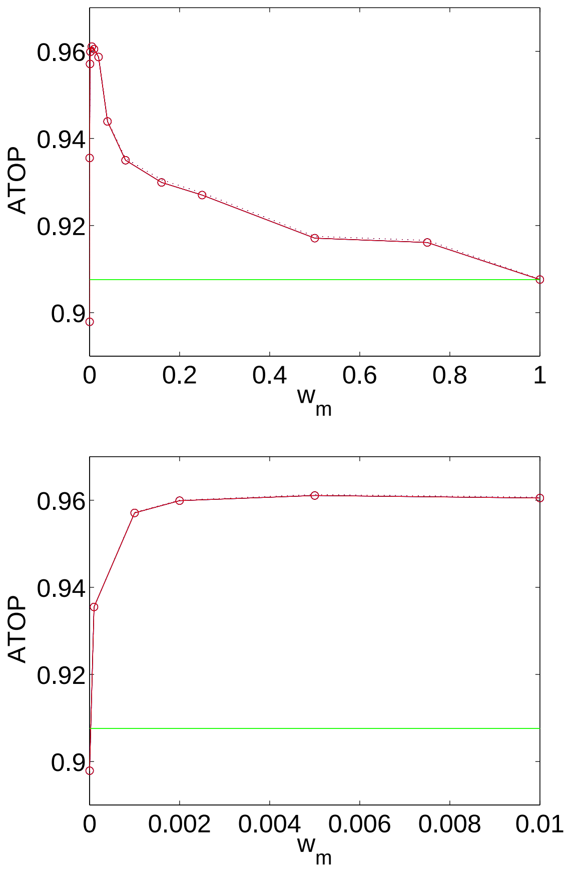

**Figure 2:** ATOP **measure on the Netflix data for different weights $w_m$ of the missing ratings; on held-out test set (solid) and on XV test set used for parameter tuning (dashed); the two curves are almost identical. For $w_m \in [0,1]$ (top), and zoomed in to $w_m \in [0,0.01]$ (bottom). There is a clear maximum at a small positive weight, while the global minimum is at $w_m = 0$ (i.e., ignoring missing data). For comparison,** ATOP **at $w_m = 1$ (horizontal line).**

Among our three objective functions for training (Eqs. 10, 11 and 13), the first one (logistic regression) turned out to be computationally too costly on the much larger Netflix data. This is expected from our experiments on the MovieLens data. Training with least squares on the binary classification data (Eq. 11) resulted in ATOP $\approx 0.94$. Like for the MovieLens data, this is slightly worse than optimizing Eq. 13 (see Table 3), but considerably better than ignoring the missing ratings ($w_m = 0$).

When optimizing Eq. 13, we have to determine the optimal values of the tuning parameters $w_m$, $r_m$ and $\lambda$. Note that we set the weights for all observed rating values to 1, $w(r) = 1$ for $r = 1, .., 5$; this was suggested by the MovieLens experiments above. Figure 1 shows the test results for different values of $r_m$ imputed for the missing ratings. For each $r_m$, the other tuning parameters were optimized.

[4] We used release 0.1 from `http://www.cofirank.org`

Several observations are interesting. First, there exists a maximum. It is located at $r_m \approx 2$, which again is considerably below the average of the observed ratings ($\approx 3.6$). For $r_m < 2$, ATOP changes only slightly, i.e., the test results do not depend very sensitively on the exact imputation value. Also note that, if the imputed value $r_m$ is interpreted as the average rating of the (unknown) complete data, then values below 1 are not permissible. For $r_m > 2$ there is a sudden drop and ATOP reaches a plateau for $r_m > 3.6$, which is the average of the observed ratings. The reason is that the best ATOP on the test data is achieved for $w_m = 0$ in our training objective function Eq. 13; in other words, the imputed values are ignored in this case, and the standard RMSE on the observed ratings is optimized for $r_m > 3.6$.

Figure 2 illustrates how ATOP changes as a function of the weight $w_m$ assigned to the missing ratings in Eq. 13. Again, for each $w_m$ the other tuning parameters were optimized. The optimum is reached at $w_m \approx 0.005$. While small, it is again close to the percentage of observed ratings (1%), resulting in a relatively balanced training set: all the observed ratings together have twice the weight compared to all the unobserved ratings together. To the left and right of the optimum, ATOP drops steadily and significantly. Interestingly, ATOP is again larger at $w_m = 1$ than at $w_m = 0$.

For (close to optimal) tuning parameter values $w_m = 0.01$, $r_m = 2$ and $\lambda = 0.04$, Figure 3 shows the TOPK($k$) curve as a function of $k$. We estimated TOPK($k$) in exactly the same way as in [8], i.e., we used sample size $\tilde{N} = 1001$, see also Section 2.3 above (note that we used $\tilde{N} = 101$ for computing ATOP in all our other experiments). We can thus compare our best model to two models: (1) the model that optimizes Eq. 13 with $w_m = 0$ (i.e., the usual RMSE), but with $\lambda$ such that ATOP is maximized on the cross validation set; this resulted in a significantly lower curve in Figure 3; (2) the results reported for the integrated model in [8], which we summarized in Table 3, alongside our results. Table 3 shows that it is essential to account for the missing ratings in the training data as to achieve high top-$k$ hit rates on the test data. By optimizing our new `AllRank` objective function–even though we used a simple model–, we significantly improved ATOP and TOPK compared to the sophisticated integrated model of [8], which was trained to optimize RMSE on the way to winning the $1 million Netflix prize. Our model with optimal weight $w_m$ achieved ATOP = 0.96, which is significantly closer to the maximal possible value of 1. Moreover, the TOPK($k$) measure shows that 39% more relevant items are ranked at the very top (i.e., none of the 1000 other, randomly chosen items ranks higher), 48% more relevant items are ranked in the top 0.2%, and 50% more in the top 2% by our new approach, compared to the integrated model in [8]; see also Figure 3 in [8] for comparison.

Note that ATOP and AUC refer to the area under the *entire* curve. In real-world top-$k$ recommender systems, one may be interested in maximizing the *initial* area under the curve, e.g., up to a pre-determined threshold $k$. Optimizing the initial area directly is important in several areas of machine learning, and currently under active research. While these two objectives are theoretically different, however, optimizing the entire area under the curve often provides a good optimization of the initial part of the area in practice. In the context of recommender systems, this was also found for the area under the top-$k$ hit rate in [8], Figure 3.

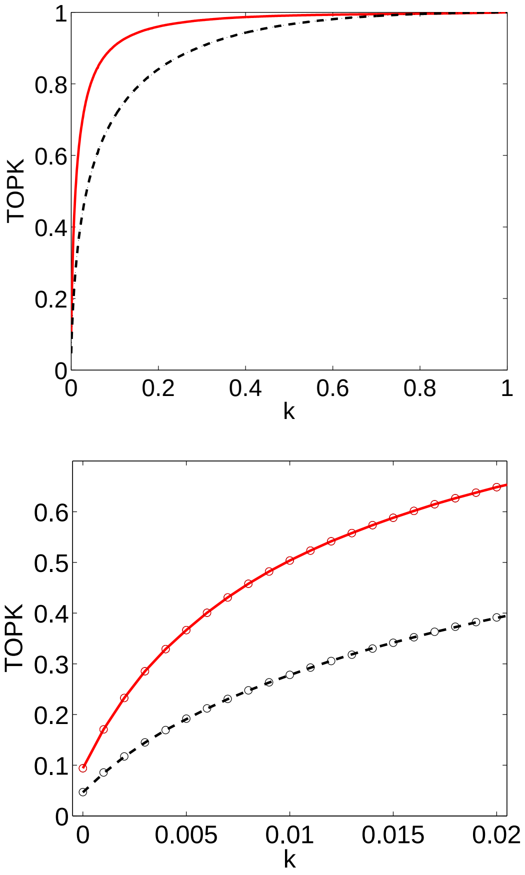

**Figure 3:** TOPK($k$) **curves on the Netflix data for** $k \in [0, 1]$ **(top), and zoomed in to top** 2% **(bottom) on held-out test set (XV test set resulted in practically indistinguishable curves in this plot). For all values of** $k$**, significant improvements are obtained by accounting for the missing ratings during training (solid line), rather than ignoring them (dashed line).**

As a baseline in Table 3, we also calculated the recommendation accuracy of three variants of the bestseller list, like for the MovieLens data above. Concerning the TOPK($k$) measure, the bestseller lists perform considerably worse than the methods that ignore the missing data mechanism in Table 3 if $k$ is *small*, which is the relevant case in practice. For larger $k$, the situation is the other way around, which explains the surprisingly good ATOP values of count-based bestseller lists 2 and 3 in Table 3. Among the bestseller lists, the variant that ignores the missing values (by calculating the mean of the observed values), is again by far the worst, like for the MovieLens data above.

## 7. CONCLUSIONS AND FUTURE WORK

In this paper, we considered the *top-k hit rate* on *all* items as the natural performance measure for recommender systems, as only a few out of all items can typically be recommended to a user at a time. We found that optimizing the popular root mean square error or other (ranking) measures

**Table 3: Netflix Data: Test results for** RMSE, ATOP **and** TOPK **measures.**

| training approach | RMSE | ATOP | TOPK(0%) | TOPK(0.2%) | TOPK(2%) |
|---|---|---|---|---|---|
| bestseller list 1: mean of observed ratings | 0.999 | 0.81 | 1.8% | 4.4% | 19.8% |
| bestseller list 2: count of observed ratings (any value) | - | 0.90 | 2.6% | 7.9% | 37.6% |
| bestseller list 3: count of observed 5-star ratings | - | 0.92 | 3.4% | 10.0% | 41.9% |
| observed ratings only: Eq. 13, $w_m = 0$ | 0.921 | 0.90 | 4.7% | 11.7% | 39.1% |
| integrated model: from [8] | 0.887 | 0.91 | 6.7% | 15.7% | 43% |
| `AllRank`-Regression: Eq. 13, $w_m = 0.01$ | 1.106 | 0.96 | 9.3% | 23.3% | 64.8% |

*on the observed data* can result in dramatically degraded performance w.r.t. the top-$k$ hit rate, in particular when the available data are *missing not at random (MNAR)*, as is typically the case in practice.

As the exact missing-data mechanism is unknown in general, we have developed mild assumptions concerning user behavior, which were inspired by general properties across various publically available data sets for recommender systems. We outlined *test* performance measures that can be evaluated in the presence of missing ratings, and which are unbiased on MNAR data under these assumptions. As these measures are computationally inefficient for *training*, we have developed appropriate surrogate objective functions. Our new family of objective functions, `AllRank`, shares the crucial property of accounting for *all* ratings–whether observed or missing in MNAR training data.

When optimizing our new objective functions (in a collaborative filtering setting with a simple matrix factorization model), we obtained considerably better results w.r.t. the top-$k$ hit rate than has been achieved with even sophisticated models that were trained to optimize RMSE on the *observed* data. Depending on the value of $k$, our approach resulted in a $39 - 50\%$ higher top-$k$ hit rate compared to state-of-the-art recommender systems in our experiments on the Netflix data.

We consider this work as only the first step toward improving recommendation accuracy concerning *all* items when the available training and test data are MNAR. The next steps include the development of more sophisticated surrogate objective functions for training, as well as more powerful models.

One cannot take for granted that the numerous results obtained for RMSE are guaranteed to carry over to the top-$k$ hit rate. Re-evaluation of these results with respect to accuracy measures concerning *all* items may be in order. One such observation might be that content-based recommender systems were clearly inferior to collaborative filtering methods w.r.t. RMSE on the *observed* data.

## Acknowledgements

I am greatly indebted to Tin Ho for her encouragement and support of this work, and for fruitful discussions with her. I am very grateful for useful discussions with Aiyou Chen and Jin Cao, and for valuable comments from the anonymous reviewers.

## 8. 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] MovieLens data. homepage: `http://www.grouplens.org/node/73`, 2006.

[3] S. Deerwester, S. Dumais, G. Furnas, R. Harshman, T. Landauer, K. Lochbaum, Lynn Streeter, et al. Latent semantic analysis / indexing. homepage: `http://lsa.colorado.edu/`.

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

[5] D. J. Hand and R. J. Till. A simple generalization of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–86, 2001.

[6] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *International Conference on Data Mining (ICDM)*, 2008.

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

[8] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Conf. on Knowledge Discovery and Data Mining (KDD)*, 2008.

[9] M. Kurucz, A. Benczur, T. Kiss, I. Nagy, A. Szabo, and B. Torma. KDD Cup 2007 task 1 winner report. *ACM SIGKDD Explorations Newsletter*, 9:53–6, 2007.

[10] R. Little and D. B. Rubin. *Statistical Analysis with missing data*. Wiley, 1986.

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

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

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

[14] D. B. Rubin. Inference and missing data. *Biometrika*, 63:581–92, 1976.

[15] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Int. Conf. on Machine Learning (ICML)*, 2007.

[16] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *International Conference on Machine Learning (ICML)*, pages 720–7, 2003.

[17] H. Steck. Hinge rank loss and the area under the ROC curve. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2007.

[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.

[19] S. Wu and P. Flach. A scored AUC metric for classifier evaluation and selection. In *ROCML workshop at ICML*, 2005.