# OrdRec: An Ordinal Model for Predicting Personalized Item Rating Distributions

Yehuda Koren
Yahoo! Research, Haifa
yehuda@yahoo-inc.com

Joe Sill
Analytics Consultant
joe_sill@yahoo.com

## ABSTRACT

We propose a collaborative filtering (CF) recommendation framework, which is based on viewing user feedback on products as ordinal, rather than the more common numerical view. This way, we do not need to interpret each user feedback value as a number, but only rely on the more relaxed assumption of having an order among the different feedback ratings. Such an ordinal view frequently provides a more natural reflection of the user intention when providing qualitative ratings, allowing users to have different internal scoring scales. Moreover, we can address scenarios where assigning numerical scores to different types of user feedback would not be easy. Our approach is based on a pointwise ordinal model, which allows it to linearly scale with data size. The framework can wrap most collaborative filtering algorithms, upgrading those algorithms designed to handle numerical values into being able to handle ordinal values. In particular, we demonstrate our framework with wrapping a leading matrix factorization CF method. A cornerstone of our method is its ability to predict a full probability distribution of the expected item ratings, rather than only a single score for an item. One of the advantages this brings is a novel approach to estimating the confidence level in each individual prediction. Compared to previous approaches to confidence estimation, ours is more principled and empirically superior in its accuracy. We demonstrate the efficacy of the approach on some of the largest publicly available datasets, the Netflix data, and the Yahoo! Music data.

## Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Apps—*Data Mining*

## General Terms

Algorithms

## Keywords

recommender systems, collaborative filtering, matrix factorization

## 1. INTRODUCTION

Collaborative filtering (CF) is a leading approach to building recommender systems which has gained much popularity recently [15]. CF is based on analyzing past interactions between users and items, and hence can be readily applied in a variety of domains, without requiring external information about the traits of the recommended products. Most CF systems view user feedback as numerical scores or as binary scores. Such a view limits the applicability of these systems. While in common star-rating systems (e.g., when user scores are between 1 star and 5 stars) viewing the qualitative user feedback as numerical may be intuitive, this is not always the case. In several common scenarios, there is no direct link between the user feedback and numerical values, even though the feedback is richer than a binary "like-vs-dislike" indication.

For example, the user feedback at Yahoo! Movies assumes the values A+,A,A-,B+,B,...,F, having no direct resemblance to numerical or binary values. In some e-commerce systems, user preferences to products are perceived by tracking the various actions users perform. For example, a user can search and browse a product page, which is a weak indication of interest in the product. A stronger indication would be bookmarking the product or adding it to a "wish list". An even stronger indication would be entering the product to the "shopping cart" or bidding on the product. The strongest indication would be actually purchasing the product. Certainly the richness of possible user actions cannot be captured with a binary indicator. Yet, mapping the user actions into a numerical scale would not be natural or trivial. Any decision to map the actions into a numerical scale, e.g. coding "search and browse" as a 1, bookmarking or wish-listing as 2, etc., would be somewhat arbitrary. An ordinal scale seems to best fit this case.

Another scenario is when users are asked to enter their feedback by a comparative ranking of a set of products. Indeed, some studies argue [2, 6, 17] that humans are more consistent when comparing products than when giving each of them an absolute score. Again, the recorded ordinal feedback in such a scenario will have no direct interpretation in terms of numerical values.

Furthermore, we argue that even when user feedback is related to absolute numbers, taking the scores as numerical may not reflect the user intentions well. Different users tend to have different internal scales. For example, taking star ratings as numeric will put the same distance between "3 stars" and both "4 stars" and "2 stars". However, one user can take "3 stars" as similar to "4 stars", while another user strongly relates "3 stars" to low quality, being similar to "1-2 stars".

In this work we suggest a novel CF framework, which we dub *OrdRec*, motivated by the above discussion and inspired by the ordinal logistic regression model originally described by McCullagh [10]. The model views user feedback as ordinal. Hence it only assumes an order among the observed feedback values, but does not require mapping these values into a numerical scale. Our framework is based on a pointwise (rather than pairwise) ordinal approach, letting it scale linearly with data size. The framework


*RecSys'11,* October 23–27, 2011, Chicago, Illinois, USA.

can wrap existing CF methods, and upgrade them into being able to tackle ordinal feedback. We work with a matrix factorization [9] CF model known as SVD++ [7], which is among the more accurate approaches reported in the literature.

An important property of OrdRec is an ability to output a full probability distribution of the scores rather than a single score, which provides richer expressive power. For example, one can predict mean, mode, account for lowest probability of "1 star", etc. In particular, *confidence levels* can be associated with the given prediction. Confidence estimation can have a significant impact on the end user experience, and allows system designers more flexibility and tools when selecting the items to recommend. Yet, the topic has so far gathered very limited attention in the literature. The approach to confidence estimation enabled by OrdRec is both more principled and empirically more accurate than previous approaches.

In passing, we also suggest a multinomial CF approach based on matrix factorization, which views ratings as categorical, and can also predict a full probability distribution of the ratings.

Our methods were extensively evaluated on three large scale datasets: the Netflix Prize dataset [3], which has became a standard benchmark, and two versions of the Yahoo! Music dataset, which underlie the KDD-Cup'2011 contest. To summarize, the main contributions of this work are:

1. A CF framework treating user ratings as ordinal rather than numerical, thereby being directly applicable to a wider variety of systems.
2. Flexibly associating different semantics to the available scores, depending on the user.
3. Predicting the full probability distribution of the scores rather than a single score.
4. Enhancing and integrating with many known CF methods.
5. New methods and evaluation metrics for assessing confidence in recommendations.

## 2. BASIC NOTIONS

We are given ratings for $m$ users and $n$ items. We reserve special indexing letters to distinguish users from items: for users $u, v$, and for items $i, j$. In addition, we index rating values by $r$. A rating $r_{ui}$ indicates the rating which user $u$ gave item $i$, where high values mean stronger preference. The ratings themselves are ordinal and need not be numbers. Thus, we assume a total order between the possible rating values. For example, values can range from 1 star (indicating no interest) to 5 stars (indicating a strong interest). We denote the number of distinct ratings by $\mathcal{S}$. For notational simplicity, we will refer to the ratings as 1,2,...,$\mathcal{S}$. In practice, however, they could actually be letter grades or any other ordered set of preference levels. Usually the data is sparse and the vast majority of ratings are unknown. We distinguish predicted ratings from known ones, by using the notation $\hat{r}_{ui}$ for the predicted value of $r_{ui}$. The set of items rated by user $u$ (in the train set) is denoted by $\mathrm{R}(u)$. The overall training set, containing all rated user-item pairs $(u, i, r = r_{ui})$ is denoted by $\mathcal{R}$. The test set is denoted by $\hat{\mathcal{R}}$.

## 3. BACKGROUND AND RELATED WORK

A popular approach to building recommender systems is *Collaborative Filtering* (CF), a term coined by the developers of the first recommender system - Tapestry [5]. CF relies only on past user behavior, e.g., their previous transactions or product ratings. It analyzes relationships between users and interdependencies among products, in order to identify new user-item associations. Latent factor CF models explain ratings by characterizing both items and users in terms of factors inferred from the pattern of ratings. One of the most successful realizations of latent factor models is based on *matrix factorization*, e.g., [9]. In particular, we will employ a variant of matrix factorization, known as SVD++, which was demonstrated to yield superior accuracy by also accounting for the more implicit information represented by the set of items which were rated (regardless of their rating value) [7]. The model predicts the rating by user $u$ of item $i$ as follows:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left( p_u + |\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} x_j \right) \quad (1)$$

Here, both users and items are mapped into a joint latent factor space of dimensionality $f$, such that ratings are modeled as inner products in that space. Accordingly, each user $u$ is associated with a vector $p_u \in \mathbb{R}^f$ and each item $i$ is associated with a vector $q_i \in \mathbb{R}^f$. A second set of item factors relates an item $i$ to a factor vector $x_i \in \mathbb{R}^f$. These secondary item factors are used to characterize users based on the set of items that they rated. The scalar $\mu$ is a constant denoting the overall average rating. The scalar parameters $b_u$ and $b_i$ are user and item biases. Model parameters are learned by stochastic gradient descent on the regularized squared error function; see [7] for full details.

The OrdRec model presented in this paper wraps SVD++ in an outer framework, which turns its predictions into a probability distribution over an ordered set of values. This framework is inspired by the ordinal linear regression model first described by McCullagh in [10]. In ordinal linear regression, we want to predict a response variable taking on one of a discrete set of $S$ ordered values (e.g. the letter grades students receive in a class) based on various other indepenent variables (e.g. demographic information about the students). Let $x$ be the vector of independent variables and $\gamma_s$ represent the probability that the response variable is less than or equal to $s$. Then the ordinal regression model is

$$log(\gamma_s/(1-\gamma_s)) = t_s - w^T x \quad (2)$$

where $w$ is a vector of coefficients associated with the independent variables, just as in ordinary least-squares linear regression. The values $t_1 \leqslant t_2 \leqslant \cdots \leqslant t_{S-1}$ can be thought of as thresholds or cutpoints determining when each of the ordered response variable levels becomes probable. The design of the OrdRec model arises out of the observation that the dot product of $w$ and $x$ in the ordinal linear regression equation can be replaced by a CF model that outputs a real-valued quantity, thereby yielding a CF model which predicts the full probability distribution over an ordered set of ratings. In addition, we suggest a new way for representing and learning the non-decreasing threshold variables together with the model parameters.

Two recent papers have successfully combined matrix factorization with ordinal modeling of user ratings, although in each case the implementation was fairly complex. In [13], a hierarchical Bayesian model for ordinal matrix factorization is presented. A collection of prior distributions over various model parameters is required and the model is trained using Gibbs sampling. The authors obtained substantial accuracy improvements over standard matrix factorization techniques on the Netflix dataset. In [18], ordinal matrix factorization is a special case of an elaborate model which combines collaborative filtering and content-based metadata information and which offers the possibility of incremental, online training. Inference is achieved via a combination of variational message passing and expectation propagation. In contrast, the work presented here as the OrdRec model represents a straight-

forward technique based on the widely-used stochastic gradient descent technique for training a matrix factorization model. The familiarity and ease of implementation it offers may make it an attractive option to many practitioners. More importantly, neither of the previous versions of ordinal matrix factorization investigates the accuracy of the confidence estimates implied by the probability distributions predicted by the models. Here, the OrdRec model is shown to produce better indicators of its own accuracy than previously used collaborative filtering confidence estimators, allowing decisions to be made based on the degree of confidence the model has in its prediction.

The recent BPR model [14] has treated ratings as ordinal. That interesting work deals with binary feedback scenario (like tag recommendations, where user feedback is implicit and binary), while we treat a setup where the user can express multiple rating levels, and we emphasize prediction of the probability distribution over these levels for each user-item pair. Another important distinction between the approaches is that we use an efficient point-wise approach to rankings, whereas BPR uses a pairwise approach, which is computationally intensive and hard to scale.

Several other papers have utilized ordinal modeling in the context of recommendation systems, although none of these other models combine a true ordinal treatment of user feedback with matrix factorization. In [12], the authors present their model as capable of modeling ordinal data, but the structure of the model is same as the structure of a multi-class classification model. Thus, the model does not take advantage of the known ordering of the ratings, and the authors acknowledge that the model is overparametrized for ordinal problems. In [19], a Gaussian process-based system for supervised learning of user preferences based on content-based item features is presented where the preference functions to be learned for a set of users are modeled jointly in a hierarchical Bayesian framework. In [4], a small-scale collaborative filtering experiment is presented where collaborative filtering is treated within the framework of standard supervised learning for the purpose of predicting the movie preferences of a set of users. Each movie is represented as an input vector with the ratings of users as the input variables, and various ad-hoc strategies are used to fill in the missing ratings.

In addition to the previous work on ordinal CF models, there are other examples of CF models which produce a probability distribution over the set of ratings. One previous work that reports scaling to large datasets while treating ratings as categorical and predicting the full probability distribution of ratings is based on a neural network model known as Restricted Boltzmann Machines (RBM) [16]. In this work we suggest a faster and more accurate matrix factorization-related alternative to RBM, which also takes ratings as categorical and predicts their full probability distribution.

Quite surprisingly, there has only been a small amount of previous work on computing and using confidence levels of recommendations; two such works are [1, 11]. A shared disadvantage of these approaches is that (1) they are non-personalized; and (2) they are not derived from the properties of the prediction algorithms, but rather rely on independent user and item data for estimating confidence. On the other hand, we will propose a method for estimating recommendation confidence which depends on the actual user and item and on the the specific recommendation method used. A more detailed comparison is presented in Sec. 7.

# 4. THE ORDREC MODEL

The OrdRec framework works together with an internal model for producing user-item scores, which will be converted into a probability distribution over the ordinal set of ratings. Henceforth, we denote such an internal scoring mechanism by $y_{ui}$. In our case, we employ the SVD++ algorithm (1), so that

$$y_{ui} = b_i + b_u + q_i^T \left( p_u + |\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} x_j \right) \qquad (3)$$

In general, any other rating predictor could serve for defining $y_{ui}$, including those whose parameters are fixed and need not be learned. However, when desiring to keep the ordinal nature of the method, we would encourage limiting the choices of $y_{ui}$ to those formulas that do not treat given ratings as numbers. (For instance, if one employs OrdRec with an item-item method, it would be more sensible to take item-item similarities as rank correlations rather than as Pearson correlations.) Our choice (3) certainly obeys this criterion.

We introduce $\mathcal{S} - 1$ ordered thresholds, associated with each of the rating values besides the last one

$$t_1 \leqslant t_2 \leqslant \cdots \leqslant t_{\mathcal{S}-1} \qquad (4)$$

Only the first threshold $t_1$ is actually a parameter in our model. The other thresholds are represented by encoding their non-negative gaps, thereby enforcing the order of the thresholds. To this end, we introduce another set of parameters $\beta_1, \ldots, \beta_{\mathcal{S}-2}$ such that

$$t_{r+1} = t_r + \exp(\beta_r) \qquad r = 1, \ldots, \mathcal{S} - 2 \qquad (5)$$

Let us denote by $\Theta$ all model parameters, that is the biases and factors participating in (3) and $t_1, \beta_1, \ldots, \beta_{\mathcal{S}-2}$. Given these parameters, we model the process of generating an obsereved rating as the following random process. First a *random score* $z_{ui}$ is generated from a normal distribution centered at the *internal score* $y_{ui}$

$$z_{ui} \sim N(y_{ui}, 1) \qquad (6)$$

Then the observed ordinal rating corresponds to the threshold bracket where the random score falls, yielding the probability distribution

$$P(r_{ui} = r|\Theta) = P(t_{r-1} < z_{ui} \leqslant t_r) \qquad (7)$$

with the convention $t_0 = -\infty$ and $t_{\mathcal{S}} = \infty$. Thus

$$P(r_{ui} \leqslant r|\Theta) = P(z_{ui} \leqslant t_r) = \Phi(t_r - y_{ui}) \qquad (8)$$

where $\Phi$ is the cumulative normal distribution function. In practice we replace $\Phi$ with the more convenient logistic function such that

$$P(r_{ui} \leqslant r|\Theta) = 1/\left(1 + \exp(y_{ui} - t_r)\right) \qquad (9)$$

Thus, the probability of observing rating $r_{ui} = r$ is

$$P(r_{ui} = r|\Theta) = P(r_{ui} \leqslant r|\Theta) - P(r_{ui} \leqslant r - 1|\Theta) \qquad (10)$$

The log likelihood of observing the training set given $\Theta$ is

$$L(\mathcal{R}) = \sum_{(u,i,r) \in \mathcal{R}} \log P(r_{ui} = r|\Theta) \qquad (11)$$

Learning proceeds by stochastic gradient ascent on $L(\mathcal{R})$. Given a training example $(u, i, r)$ we update each individual parameter $\theta$ by

$$\Delta\theta = \eta \left( \frac{\partial \log P(r_{ui} = r|\Theta)}{\partial \theta} - \lambda\theta \right) = \frac{\eta}{P(r_{ui} = r|\Theta)} \cdot \left( P(r_{ui} \leqslant r|\Theta)(1 - P(r_{ui} \leqslant r|\Theta)) \frac{\partial(t_r - y_{ui})}{\partial\theta} - P(r_{ui} \leqslant r - 1|\Theta)(1 - P(r_{ui} \leqslant r - 1|\Theta)) \frac{\partial(t_{r-1} - y_{ui})}{\partial\theta} - \lambda\theta \right) \qquad (12)$$

where $\eta$ is the learning rate and $\lambda$ is the regularization rate.

The OrdRec model gives us the flexibility to define a separate set of thresholds per user or per item. In our setting, the thresholds are user-specific. This allows the model to express the unique rating scale of each user. Hence, for each user $u$ we define a set of thresholds $t_1^u \leqslant t_2^u \leqslant \cdots \leqslant t_{\mathcal{S}-1}^u$, which are based on user-specific $\beta_1^u, \ldots, \beta_{\mathcal{S}-2}^u$. Acordingly, we replace (9) with

$$P(r_{ui} \leqslant r|\Theta) = 1/\left(1 + \exp(y_{ui} - t_r^u)\right) \tag{13}$$

The rest of the derivation is left intact. Since our thresholds are user-specific, we no longer need to employ user biases ($b_u$) in the definition of $y_{ui}$ (3).

Similarly, we could work with thresholds depending on both users and items $t_1^{u,i} \leqslant t_2^{u,i} \leqslant \cdots \leqslant t_{\mathcal{S}-1}^{u,i}$ by defining

$$t_{r+1}^{u,i} = t_r^{u,i} + \exp(\beta_r^u + \beta_r^i) \qquad r = 1, \ldots, \mathcal{S}-2 \tag{14}$$

Or alternatively

$$t_{r+1}^{u,i} = t_r^{u,i} + \exp(\beta_r^u) + \exp(\beta_r^i) \qquad r = 1, \ldots, \mathcal{S}-2 \tag{15}$$

We leave exploring such user-item combined threshold schemes to future work.

## 4.1 Ranking items for a user

OrdRec predicts a full probability distribution over ratings, rather than the common prediction of single rating value. Thus, OrdRec provides a richer output, which goes beyond describing only the average rating or the most likely rating. This can have an impact on system design, e.g., by letting certain aspects of the system be more conservative (avoiding high probability of the lowest rating) or more daring (caring mostly about the probability of the highest rating). Another advantage of predicting a full probability distribution is an ability to assess the confidence in predictions, to which we dedicate Sec. 7.

Here, we would like to discuss ways to rank products for a user given their predicted rating distributions. When the ratings are numerical, the predictions can get ranked by their associated mean or median. However, for the more general case, where ratings need not be numbers, computing statistics like mean would no longer be plausible. For example, we may need to decide preference order between a product with probability $3/4$ to be rated "A+" and a probability $1/4$ to be rated "F", and a product with a probability 1 of being rated "C". Hence we devise a general data-driven method for ranking the items considered for a user.

Let us formally cast the problem as a learning-to-rank task. Given an OrdRec model parameterized by $\Theta$, a user $u$ and two items $i$ and $j$, we define the vector $\Delta P_u(i,j) \in \mathbb{R}^{\mathcal{S}}$ such that $\Delta P_u(i,j)[r] = P(r_{ui} = r|\Theta) - P(r_{uj} = r|\Theta)$ for $r = 1, \ldots, \mathcal{S}$.

We include in a training set $\mathcal{T}$ all ordered item pairs together with their predicted rating distribution differential

$$\mathcal{T} = \{(i,j,\Delta)|r_{ui} > r_{uj}, \Delta = \Delta P_u(i,j)\} \tag{16}$$

We would like to learn a linear map $w \in \mathbb{R}^{\mathcal{S}}$ that maximizes the well ordering of items through the function

$$F(\mathcal{T}|w) = \sum_{(i,j,\Delta)\in\mathcal{T}} I(w^T\Delta \geqslant 0) \tag{17}$$

We approximate the step function $I$ with the logistic function $\sigma(x) = 1/(1 + exp(-x))$, and maximize the continuous function

$$\bar{F}(\mathcal{T}|w) = \sum_{(i,j,\Delta)\in\mathcal{T}} \sigma(w^T\Delta) \tag{18}$$

The weight vector $w$ is learned through stochastic gradient ascent. Given $w$, we transform any predicted rating distribution into a number, which allows us to rank all the items for a given user. In our tests, with datasets where ratings are numeric, ranking items by optimizing $\bar{F}$ provides results as good (in term of $F$ or equivalently in terms of the AUC measure) as ranking them by their expected rating score, with the advantage of being able to handle also datasets where ratings are not numeric.

# 5. A MULTINOMIAL FACTOR MODEL

We would like to compare the OrdRec model against an alternative model that can also: (1) deal with non-numerical scores; (2) predict a full probability distribution of scores for each user-item pair. One such alternative model, which is reported to scale to the large datasets we experiment with, is the aforementioned RBM model [16]. However, in our experiments RBM proved to be fairly slow to train and lagged in its accuracy behind the matrix factorization-based methods; see Sec. 6 Hence we would like to suggest a matrix factorization-based method, which like RBM models a multinomial distribution over categorical scores, while offering improved speed and accuracy over RBM. We now turn to describe this method, which we dub *Multinomial Matrix Factorization*, or in short *MultiMF*.

As with OrdRec, we use MultiMF with an SVD++ underlying model, which our experiments showed to be effective. However, one can use the same framework with other models.

For each score $r$ we define:

$$z_{ui}^r = \mu + b_i^r + b_u^r + \left(p_u + |\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j\in\mathrm{R}(u)} x_j\right)^T q_i^r \tag{19}$$

So far, the difference from the SVD++ model (1) is the usage of score-dependent biases ($b_u^r, b_i^r \in \mathbb{R}$), and a score-dependent item factor vector ($q_i^r \in \mathbb{R}^f$). Now, the probability of observing $r_{ui} = r$ follows the multinomial distribution

$$P(r_{ui} = r|\Theta) = \frac{\exp(z_{ui}^r)}{\sum_{r'=1}^{\mathcal{S}} \exp(z_{ui}^{r'})} \tag{20}$$

where $\Theta$ denotes all model parameters (user and item factors and biases). We would like to maximize the log likelihood of the training data given $\Theta$

$$\mathcal{L}(\mathcal{R}) = \sum_{(u,i,r)\in\mathcal{R}} \log P(r_{ui} = r|\Theta) \tag{21}$$

Learning proceeds by stochastic gradient ascent on $\mathcal{L}(\mathcal{R})$. Given a training example $(u,i)$ with $r_{ui} = k$ we update each individual parameter $\theta$ by

$$\begin{aligned}\Delta\theta =& \eta\left(\frac{\partial \log P(r_{ui} = r|\Theta)}{\partial\theta} - \lambda\theta\right) = \\ & \eta\left(\frac{\partial z_{ui}^r}{\partial\theta} - \sum_{r'=1}^{\mathcal{S}} P(r_{ui} = r'|\Theta)\frac{\partial z_{ui}^{r'}}{\partial\theta} - \lambda\theta\right)\end{aligned} \tag{22}$$

where $\eta$ is the learning rate, and $\lambda$ is the regularization rate.

This concludes our description of MultiMF. Before turning to the empirical study, we would like to mention two disadvantages that approaches like MultiMF or RBM have in comparison to OrdRec.

1. MultiMF and RBM model user ratings as categorical, which is often a less realistic assumption than the ordinal assumption by OrdRec. For example, the categorical representation misses the fact that a "5 star" rating is more similar to a "4 star" rating than to a "1 star" rating.

2. Unlike OrdRec, MultiMF and RBM require a separate item factor vector for each score. Hence their space complexity is $\Omega(\mathcal{S}\cdot f\cdot n)$, having a multiplicative dependency in three quantities: number of distinct scores, factorization dimension and number of items. For the Netflix dataset, where number of items (17,770) is much lower than the number of users, and there are just 5 distinct scores, such a complexity might not be restrictive. However, as we move to datasets with a larger number of items, it might get more prohibitive. For example, as we will see in Sec. 6, the Yahoo! Music dataset has 624,961 items, and 11 different scores, where the item factors will start consuming a significant amount of memory, limiting their maximal possible dimensionality.

# 6. EMPIRICAL STUDY

## 6.1 Datasets description

We evaluated OrdRec on some of the largest publicly available user rating datasets. First is the Netflix dataset [3], which was subject to extensive research as part of Netflix Prize contest. The other twin datasets, denoted by Y!Music-I and Y!Music-II are based on user ratings on Yahoo! Music. Ratings were given to musical entities of four different types: tracks, albums, artists, and genres. The larger of the two, Y!Music-II, is the subject of the KDD-Cup'11 contest. The Y!Music-I dataset was constructed similarly to Y!Music-II, but on a disjoint user set of half the size.

All three datasets are split into train, validation and test sets. Some of their basic statistics are given in Table 1. Number of ratings range from 103,297,638 for the Netflix dataset to 262,810,175 for the Y!Music-II dataset. The Y!Music datasets are characterized by having a much larger number of items than the Netflix dataset (624,961 vs. 17,770), reflecting the fact that there are many more musical tracks than movies. In terms of number of distinct ratings, the Netflix dataset has five such values (1-star to 5-stars), while the Y!Music datasets have 11 distinct values (0,1,...,10).[1] We plot the scores distribution of the three datasets in Fig. 1. Interestingly, while the Netflix dataset is unimodal, with most ratings concentrated around high values, the Y!Music datasets are bimodal, with distribution peaking around the very high (9) and very low (0) scores.

All results are reported on the test sets, which were excluded from the training process. For the Netflix dataset we followed the common practice of including the validation set in the training set, when predicting the test set, which is known to significantly improve test results. However, we did not follow the same practice at the Y!Music datasets, where the validation set was always excluded from the train set.

In both datasets the representation of each user in the test and validation sets is bounded to a few ratings at most (hence, no bias towards heavy raters). For example, in the Y!Music datasets each user has exactly 4 ratings in the validation set and 6 ratings in the test set (the latest ratings of each user).

The validation set was used for setting the parameters of the evaluated algorithms, including: learning rate (a.k.a., step-size), regularization rate (a.k.a., weight-decay), and number of iterations over the training set. We have used a different learning rate and regularization rate for each type of parameter. For example, the learning rate used for item factors is not the same as the learning rate used for user factors. In order to set these values, we have performed a grid search on each pair of learning and regularization rates (both belonging to the same type of parameter), while picking the values yielding best results on the validation set.

[1] In the original data about 2.2% of the ratings were lying in-between 0,1,...,10, and we rounded those to their closest integer.

## 6.2 Evaluation metrics

We use two evaluation metrics for comparing the performance of different algorithms. First, we use the root mean squared error (RMSE), which is the standard metric on the Netflix dataset, and is often favorable thanks to its elegance and mathematical tractability

$$RMSE(\hat{\mathcal{R}}) = \frac{1}{|\hat{\mathcal{R}}|} \sum_{(u,i,r)\in\hat{\mathcal{R}}} (\hat{r}_{ui} - r)^2 \tag{23}$$

Despite its merits, RMSE can be quite detached from item ranking experience, which is often the ultimate goal of a recommender system. This is because a perfectly ranked solution can score arbitrarily badly on an RMSE scale by having scores on the wrong scale, e.g., out of bounds, or just very close to each other. Quite surprisingly, the other direction is also partially true. Decent solutions in RMSE terms can contain no personalization power ranking-wise. For example, on the Netflix dataset a predictor explaining only rating biases could get RMSE as low as 0.9278 [8], which is a significant improvement over the RMSE=0.9514 baseline set by Netflix's commercial system [3]. Nonetheless, such a solution, which is based on only explaining biases, is meaningless ranking-wise. All user-dependent biases play no role when ranking items for a single user, while the item-related biases are not personalized. Thus, it will yield the same item ranking for all users.

The RMSE metric has another issue, particularly important in our context: it assumes numerical rating values. Thus, it shares all the discussed disadvantages of such an assumption. First, it cannot express rating scales which vary among different users. Second, it cannot be applied at cases where ratings are ordinal. Thus, besides using RMSE we also employ a ranking-oriented metric, which is free of the aforementioned issues.

Given a test set $\hat{\mathcal{R}}$, we define the number of concordant pairs for user $u$ by counting those ranked correctly by rating predictor $\hat{r}_{u\cdot}$

$$n_c^u = |\{(i,j) \mid \hat{r}_{ui} > \hat{r}_{uj} \text{ and } r_{ui} > r_{uj}\}| \tag{24}$$

Similarly, we count the discordant pairs for user $u$, which are misranked by $\hat{r}_{u\cdot}$

$$n_d^u = |\{(i,j) \mid \hat{r}_{ui} \geqslant \hat{r}_{uj} \text{ and } r_{ui} < r_{uj}\}| \tag{25}$$

Summing over all users we define $n_c = \sum_u n_c^u$ and $n_d = \sum_u n_d^u$. The quality metric we use measures the proportion of well ranked items pairs, denoted by FCP (for Fraction of Concordant Pairs)

$$FCP = \frac{n_c}{n_c + n_d} \tag{26}$$

a measure that generalizes the known AUC metric into non-binary ordered outcomes. The number of ordered pairs in each of the test sets is given in the rightmost column of Table 1.

## 6.3 Results

We compared OrdRec with the followings methods: (1) SVD++ [7], which represents a leading RMSE-oriented method; (2) RBM [16], which is aimed at likelihood maximization and can work with categorical scores; (3) MultiMF (Sec. 5) having properties similar to RBM, but with improved performance. Results on the three datasets are reported in Tables 2–4.

Results on the Netflix Prize place OrdRec as the leader both in terms of RMSE and in terms of FCP (where it is in a virtual tie with SVD++). It is notable that OrdRec outperforms SVD++ in RMSE-terms, despite the fact that only SVD++ is aiming at optimizing the RMSE measure. The strong performance of OrdRec may be attributed to its better ability to model ordinal semantics of user ratings. As for the other two algorithms, MultiMF yields results significantly better than RBM, despite the two sharing related

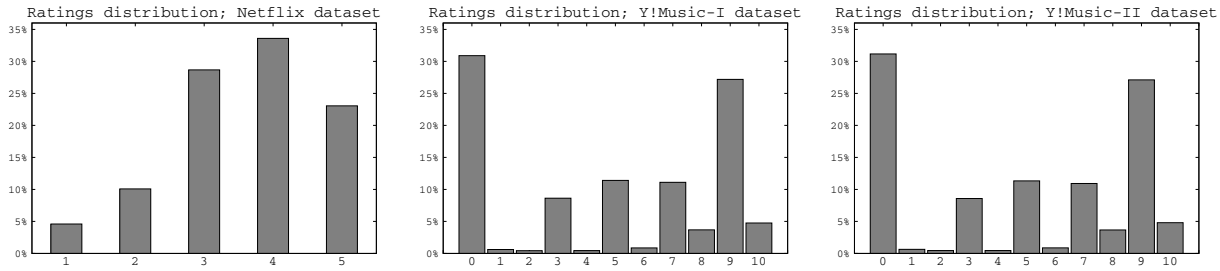


**Figure 1: Histograms of ratings distributions for the three datasets**

| Dataset | number of users | number of items | \|Train\| | \|Validation\| | \|Test\| | #of ordered test pairs ($n_c + n_d$) |
|---|---|---|---|---|---|---|
| **Netflix** | 480,189 | 17,770 | 99,072,112 | 1,408,395 | 2,817,131 | 4,879,515 |
| **Y!Music-I** | 500,269 | 445,440 | 123,318,314 | 2,001,076 | 3,001,614 | 3,357,977 |
| **Y!Music-II** | 1,000,990 | 624,961 | 252,800,275 | 4,003,960 | 6,005,940 | 6,644,664 |

**Table 1: Properties of the three datasets used in our study**

| RMSE | | | |
|---|---|---|---|
| **Method** | **f = 50** | **f = 100** | **f = 200** |
| **SVD++** | .8952 | .8924 | .8911 |
| **RBM** | .9147 | .9063 | .9023 |
| **MultiMF** | .8953 | .8938 | .8930 |
| **OrdRec** | **.8894** | **.8878** | **.8872** |

| FCP | | | |
|---|---|---|---|
| **Method** | **f = 50** | **f = 100** | **f = 200** |
| **SVD++** | 74.26% | 74.46% | **74.54%** |
| **RBM** | 72.98% | 73.57% | 73.86% |
| **MultiMF** | 74.06% | 74.17% | 74.25% |
| **OrdRec** | **74.36%** | **74.50%** | **74.54%** |

**Table 2: Performance on the *Netflix* test set of the different models under different dimensionalities. Results are measured by RMSE (lower is better) and by FCP (higher is better).**

modeling assumptions. For all algorithms, performance improves as dimensionality is increasing.

| RMSE | | | |
|---|---|---|---|
| **Method** | **f = 50** | **f = 100** | **f = 200** |
| **SVD++** | **2.4427** | **2.4430** | **2.4403** |
| **RBM** | 2.8294 | 2.8534 | 2.8373 |
| **MultiMF** | 2.4932 | 2.4925 | 2.4916 |
| **OrdRec** | 2.4808 | 2.4730 | 2.4669 |

| FCP | | | |
|---|---|---|---|
| **Method** | **f = 50** | **f = 100** | **f = 200** |
| **SVD++** | 72.78% | 72.79% | 72.80% |
| **RBM** | 64.21% | 63.07 % | 63.05% |
| **MultiMF** | 72.67% | 72.67% | 72.69% |
| **OrdRec** | **73.65%** | **73.85%** | **73.99%** |

**Table 3: Performance on the *Y!Music-I* test set of the different models under different dimensionalities. Results are measured by RMSE (lower is better) and by FCP (higher is better).**

| RMSE | | | |
|---|---|---|---|
| **Method** | **f = 50** | **f = 100** | **f = 200** |
| **SVD++** | **2.4369** | **2.4347** | **2.4334** |
| **MultiMF** | 2.4823 | 2.4848 | 2.4868 |
| **OrdRec** | 2.4786 | 2.4708 | 2.4660 |

| FCP | | | |
|---|---|---|---|
| **Method** | **f = 50** | **f = 100** | **f = 200** |
| **SVD++** | 72.59% | 72.42% | 72.13% |
| **MultiMF** | 73.01% | 73.03% | 72.99% |
| **OrdRec** | **73.63%** | **73.83%** | **73.98%** |

**Table 4: Performance on the *Y!Music-II* test set of the different models under different dimensionalities. Results are measured by RMSE (lower is better) and by FCP (higher is better).**

Moving on to the two Y!Music datasets, we observe similar results on the two. SVD++ consistently yields the best results RMSE-wise. Although SVD++ did not have the best RMSE on the Netflix dataset, it is not surprising that it achieves the best RMSE on at least one of the datasets, given the fact that SVD++ is the only method directly trained to minimize RMSE. The reader should note that RMSE values on the Y!Music datasets are significantly greater than those reported for the Netflix dataset. This is mostly explained by the more extended rating range at the Y!Music datasets (0–10 vs. 1–5), which is expected to proportionally increase RMSE by 10/4. However, the FCP scores are not as sensitive to rating scale. The OrdRec model consistently outperforms the rest in terms of FCP, indicating that it is capable of better ranking items for a user. This may reflect the benefit of better modeling of the semantics of user feedback. The RBM model produced inferior results on Y!Music-I. Given its slow running time, we did not complete its runs on the Y!Music-II dataset.

Note that only SVD++ directly aims at minimizing RMSE, so measuring accuracy by the same RMSE metric would not be a neutral judging criterion here. Therefore we tend to view performance under the FCP metric (which none of the evaluated methods directly aims at) as more representative of differences in user experience. This places OrdRec as the top performer among evaluated algorithms across all datasets. However, we would like to emphasize that the added predictive performance is not the main motivation behind OrdRec, but rather its ability to apply to cases where ratings are ordinal scores, where SVD++ and the likes cannot be applied. Also, we view the ability of OrdRec to yield probabilistic ratings as a helpful feature going beyond the scope of the evaluation study here, as will be discussed in Sec. 7.

Finally we report training times of the different methods on the largest dataset – Y!Music-II. All methods were implemented in C++, with a similar degree of optimization and ran on an HP DL160 G6, 2xXeon X5650 2.67GHz machine. Running times heavily depend on platform and implementation, so they should be only taken as relative. The times are reported in Table 5. While time per iteration is linear in input size for all methods, times differ significantly across methods. SVD++, which does not require expensive exponentiation operations, is much faster than the rest. On the other hand, RBM is the slowest, and also requires many more iterations to converge.

| Method | #iterations | f=50 | f=100 | f=200 |
|---|---|---|---|---|
| **SVD++** | **10** | **0:02:44** | **0:04:52** | **0:11:34** |
| **RBM** | 150 | 0:31:58 | 00:49:29 | 1:27:34 |
| **MultiMF** | 11 | 0:29:39 | 0:41:33 | 1:11:17 |
| **OrdRec** | 25 | 0:14:31 | 0:17:10 | 0:22:55 |

**Table 5: Training times (h:mm:ss) per iteration on the *Y!Music-II* dataset for the different methods.**

# 7. ESTIMATION OF RECOMMENDATION CONFIDENCE

A recommender system has varying levels of confidence (or, certainty) in the different recommendations it provides. Accordingly, McNee et al. [11] suggested adding a confidence level indicator to the GUI of a recommendation system, as a way of improving user trust in the system and altering user behavior. Even when not directly exposed to end users, confidence measurement bears impact on the internal working of the system. For example, when picking among several items with the same expected rating, the system can favor the item for which the confidence in the prediction is greatest. Additionally, the system can combine different recommendation techniques based on the confidence each has when predicting a particular user-item pair.

Adomavicius et al. [1] propose confidence measures which are based on item or user rating variance, while treating the recommendation algorithm as a "black box". This is based on the observation that recommendations tend to be more accurate for users and items exhibiting lower rating variance. Similarly, recommendation algorithms are expected to be more accurate for items and users associated with many ratings. However, such static metrics are not fully satisfying. User-dependent metrics fail to be applicable to ranking items for the same user, a high-priority goal of any recommendation system. Item-dependent metrics are not personalized, and will generate low confidence assessments for the same items (usually the controversial and less popular) equally for all users. Furthermore, assessing confidence without regards to the inner workings of the prediction algorithm is likely to overlook some valuable information. Certainly, such assessments would not be helpful for combining different recommendation algorithms based on differing confidence values.

Methods like OrdRec, which predict a full distribution of ratings, allow a more principled approach to confidence estimation. For each user-item pair, we associate confidence level with the amount of concentration of the rating probabilities. This way, we can employ metrics like the standard deviation, entropy, or Gini impurity associated with the predicted rating distribution of the actual user-item pair. The resulting confidence metric is personal, being dependent on both user and item, together with a dependency on the inner workings of the algorithm used.

In order to assess whether the predicted rating distribution of the OrdRec technique is helpful in estimating the level of confidence in the predictions, we formulate confidence estimation as a binary classification problem. We wish to predict whether the model's predicted rating is within one rating level of the true rating. For these purposes, the model's predicted rating is taken to be the expected value of the predicted rating distribution. Using logistic regression, we trained "confidence classifers" to predict if the model's error is larger than 1 rating level. For example, in the Netflix dataset, if the model's prediction is 3.5 stars and the true rating is 4 stars, then the model is within 1 rating level, whereas if the true rating is 5 stars, then it is not.

We ran experiments on both the Netflix dataset and the Y!Music-I dataset, in both cases using the Test sets, which the OrdRec model had not been trained on. This out-of-sample data formed the full dataset on which the confidence classifiers were trained and tested. The classifiers were trained on a randomly-chosen two-thirds of the data and tested on the remaining one-third. Overall, 22.7% of the Netflix data points and 46.8% of the Y!Music-I data points had an error magnitude of more than 1 rating.

To measure how much value the OrdRec predicted rating distribution adds to confidence estimation, we first obtained results when using only traditional indicators of confidence used in previous work, such as user and item rating standard deviation or number of user and item ratings. Classifiers were trained using each of a variety of different features, both individually and in conjunction with each other.

Tables 6 and 7 show the results on the Netflix and Y!Music-I datasets, respectively. The user and item support (number of user and item ratings) and the standard deviation of the user and item ratings are referred to collectively in the table as the traditional features. A classifier using all 4 of these traditional features was trained and tested in order to see the best accuracy that could be achieved without any of the novel features derived from OrdRec. All features listed in the table starting with "OrdRec" (e.g. *OrdRec stdev*) refer to features extracted from the OrdRec model's predicted rating distribution. To determine the best possible performance overall, a classifier with all features—both traditional and OrdRec-derived—was also trained.

*OrdRec stdev* is simply the standard deviation of the predicted rating distribution. *OrdRec max rating prob* is the largest probability for any single rating in the predicted distribution. *OrdRec entropy* is the well-known entropy of a probability distribution, $-\sum_s P_s log(P_s)$. Gini impurity is defined as $1 - \sum_s P_s P_s$.

We report the performance of the confidence classifiers based on both *AUC* and the *mean log-likelihood*. The *Area Under the ROC Curve (AUC)* measures the probability that a positive example is scored higher than a negative example. Hence, in our case it measures how well each predictor orders the points from most confident to least confident. Higher AUC values are desired, as they indicate a higher fraction of well-ordered pairs. Note that AUC is independent of the threshold chosen for determining which of the two classes is to be predicted.

Logistic regression classifiers output a probability for each class and therefore can be evaluated in terms of mean log-likelihood on the test set. For perspective, on the Netflix dataset the trivial baseline log-likelihood achieved by always predicting the frequency of the more common class (aka, a constant classifier) is -0.536, whereas for the Y!Music-I dataset it is -0.691.

The results clearly indicate that the information derived from the OrdRec predicted rating distribution is more valuable than the traditional features. On the Netflix dataset, the test log-likelihood using all traditional features combined is -0.514 and the AUC is 0.645, whereas adding the OrdRec-derived features boosts the results to Log-Likelihood=-0.485 and AUC=0.708. In fact, each single OrdRec-derived feature outperforms the combination of the 4 traditional features, with best results for *OrdRec stdev* and then *OrdRec entropy.*

Similarly on the Y!Music-I dataset, the best that can be achieved with traditional features is Log-Likelihood=-0.632 and AUC=0.723, whereas when the OrdRec-derived features are also used, the results improve to Log-Likelihood=-0.463 and AUC=0.862. Again, each individual OrdRec-derived feature already significantly outperforms the combination of the 4 traditional features, now with *OrdRec entropy* yielding best results followed by *OrdRec stdev*.

Confidence experiments conducted using the MultiMF model are omitted for the sake of brevity, but follow a similar pattern, with the features derived from the model's predicted probability distribution being more valuable than the traditional features.

It is interesting to note that the AUC and the percentage improvement over the trivial baseline is significantly larger for the Y!Music-I dataset. This is not surprising in light of Figure 1, which shows that the ratings for the Y!Music-I dataset are much more bimodal than for the Netflix dataset. The Y!Music-I dataset likely provides richer information for the sake of learning which scenarios are associated with high uncertainty.

| Feature(s) | Test Log-Likelihood | AUC |
|---|---|---|
| constant classifier | -0.536 | 0.500 |
| *user support* | -0.534 | 0.556 |
| *stdev user ratings* | -0.521 | 0.619 |
| *item support* | -0.536 | 0.515 |
| *stdev item ratings* | -0.530 | 0.576 |
| All traditional features | -0.514 | 0.645 |
| *OrdRec stdev* | -0.490 | 0.698 |
| *OrdRec max rating prob* | -0.502 | 0.674 |
| *OrdRec entropy* | -0.494 | 0.692 |
| *OrdRec Gini impurity* | -0.498 | 0.691 |
| All features | **-0.485** | **0.708** |

**Table 6: Confidence estimation results on the Netflix dataset (higher values are better)**

| Feature(s) | Test Log-Likelihood | AUC |
|---|---|---|
| constant classifier | -0.691 | 0.500 |
| *user support* | -0.689 | 0.620 |
| *stdev user ratings* | -0.638 | 0.723 |
| *item support* | -0.679 | 0.642 |
| *stdev item ratings* | -0.690 | 0.531 |
| All traditional features | -0.632 | 0.723 |
| *OrdRec stdev* | -0.543 | 0.837 |
| *OrdRec max rating prob* | -0.554 | 0.828 |
| *OrdRec entropy* | -0.537 | 0.842 |
| *OrdRec Gini impurity* | -0.538 | 0.835 |
| All features | **-0.463** | **0.862** |

**Table 7: Confidence estimation results on the Y!Music-I dataset (higher values are better)**

# 8. CONCLUSIONS

The ratings users supply to a recommender system can come in many different forms, including thumbs-up/down, like-votes, stars, numerical scores, or A-to-F grades. In addition, users generate more implicit feedback indicating different levels of interest in a product, depending on the actions they take. An example of such a range of possible actions, with a roughly increasing order of significance is: browsing, tagging, saving, adding to cart, bidding and actually buying.

Most recommender systems treat user input as numeric or binary, which is usually convenient to model and compute with. However, the numeric view might be too strict and would not naturally apply at all cases.

We advocate taking user feedback as ordinal, a view unifying all feedback examples given above. The ordinal view relaxes the numerical view to the proper extent, allowing it to deal with all usual kinds of user feedback, without assuming an over-relaxed approach representing user feedback as categorical, which would discard the internal structure of the feedback. Another merit of the ordinal view, which applies even at cases where feedback can be naturally mapped to numbers, is that it allows expressing the fact that different users have distinct internal scales for their qualitative ratings.

This motivates the OrdRec model, which treats user ratings as ordinal. Our empirical study shows that OrdRec performs favorably on datasets where traditional methods taking numerical ratings can be applied. OrdRec employs a point-wise approach to ordinal modeling, letting its training time scale linearly with dataset size. Indeed, we demonstrated it with some of the largest publicly available datasets containing 100s of millions of ratings. Yet, we should mention that efficient methods considering ratings as numbers will train considerably faster in practice, so we would not dismiss taking the numerical view altogether.

There are various ways to extend the OrdRec method. For example, it is known that integrating additional signals can vastly improve recommendation quality. This way, temporal signals or item attributes for rarely rated items can be introduced into OrdRec in ways we leave to future work.

# 9. REFERENCES


[1] G. Adomavicius, S. Kamireddy and Y. Kwon. Towards More Confident Recommendations: Improving Recommender Systems Using Filtering Approach Based on Rating Variance *Proc. 17th Workshop on Information Technology and Systems (WITS'07)*, 2007.

[2] S. Agarwal, J. Wills, L. Cayton, G. Lanckriet, D. Kriegman, and S. Belongie. Generalized Non-metric Multidimensional Scaling. *AISTATS*, San Juan, Puerto Rico, 2007.

[3] J. Bennett and S. Lanning. The Netflix Prize. *Proc. KDD Cup and Workshop*, 2007.

[4] W. Chu and Z. Gharamani. Gaussian Processes for Ordinal Regression *J. Mach. Learn. Res 6*, 1019-1041, 2005.

[5] D. Goldberg, D. Nichols, B. M. Oki and D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM* 35:12, 61–70, 1992.

[6] M. Kendall and K. D. Gibbons. Rank Correlation Methods. Oxford University Press, 1990.

[7] Y. Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. *Proc. 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'08)*, pp. 426–434, 2008.

[8] Y. Koren. The BellKor Solution to the Netflix Grand Prize. 2009.

[9] Y. Koren, R. Bell and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42:8, 30–37, 2009.

[10] P. McCullagh. Regression Models for Ordinal Data (with Discussion). *Journal of the Royal Statistical Society, Series B* 42:109-142, 1980.

[11] S.M. McNee, S.K. Lam, C. Guetzlaff, J.A. Konstan and J. Riedl. Confidence Displays and Training in Recommender Systems. *Proc. Conference on Human-Computer Interaction (INTERACT '03)*, 176–183, 2003.

[12] A. Menon, C. Elkan. A Log-Linear Model with Latent Features for Dyadic Prediction. *Proc. IEEE International Conference on Data Mining (ICDM'10)*, 2010.

[13] U. Paquet, B. Thomson and O. Winther. Large-Scale Ordinal Collaborative Filtering. *1st Workshop on Mining the Future Internet, Future Internet Symposium*, 2010.

[14] S. Rendle, C. Freudenthaler, Z. Gantner and L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. *Proc.25th Conference on Uncertainty in Artificial Intelligence (UAI'09)*, 2009.

[15] F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor. *Recommender Systems Handbook*. Springer 2010.

[16] R. Salakhutdinov, A. Mnih and G. Hinton. Restricted Boltzmann Machines for collaborative filtering. *Proc. 24th International Conference on Machine Learning (ICML'07)*, pp. 791–798, 2007.

[17] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. *Neural Information Processing Systems (NIPS'04)*, 2004.

[18] D. Stern, R. Herbrich and T. Graepel. Matchbox: Large Scale Bayesian Recommendations. *Proc. 18th International World Wide Web Conference (WWW'09)*, 2009.

[19] S. Yu, K. Yu, V. Tresp and H.P Kriegel. Collaborative Ordinal Regression *Proc. 23rd International Conference on Machine Learning (ICML'06)*, 1089-1096,2006.