Collaborative Filtering on Ordinal User Feedback[*]

Yehuda Koren
Google
yehudako@gmail.com

Joseph 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. 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. The framework can wrap most collaborative filtering algorithms, enabling algorithms previously designed for numerical values to handle ordinal values. We demonstrate our framework by 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 two of the largest publicly available datasets: the Netflix data and the Yahoo! Music data.

1 Introduction

Collaborative filtering (CF) is a leading approach to building recommender systems which has gained much popularity recently [Ricci *et. al.*, 2010]. 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, in some e-commerce systems, user preferences for products are inferred by tracking the various actions users perform. Browsing a product, adding it to a wish list, adding it to a shopping cart, bidding on it or actually buying the product are actions which each indicate a successively larger degree of interest in the product. An ordinal scale fits this case better than a numerical coding of the actions. Furthermore, numerical representation implicitly assumes the same rating scale across all users, whereas an ordinal representation allows different users to have different gaps between rating values.

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 [McCullagh,1980]. 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. The framework can wrap existing CF methods, and upgrade them into being able to tackle ordinal feedback.

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. In particular, *confidence levels* can be associated with the given prediction. Confidence estimation can have a significant impact on the end user experience. The approach to confidence estimation enabled by OrdRec is both more principled and empirically more accurate than previous approaches.

Our methods were extensively evaluated on two large scale datasets: the Netflix Prize dataset [Bennet and Lanning, 2007], which has became a standard benchmark, and the Yahoo! Music dataset.

This work is presented in more detail in [Koren and Sill, 2011].

2 Basic notation

We are given ratings for m users and n items, with u, v indexing users and i, j indexing items. In addition, we index rating values by r. A rating r_{ui} indicates the rating which

[*]The paper on which this extended abstract is based was the recipient of the best paper award of the 2011 ACM Conference on Recommendation Systems (RecSys '11) [Koren and Sill, 2011].

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. 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

Collaborative Filtering (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., [Koren *et. al.*, 2009]. In our experiments, we employ a variant of matrix factorization, known as SVD++. We use SVD++ in two ways: in its standard form as a baseline against which to compare OrdRec and also as the core of the OrdRec model itself. SVD++ has been shown 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) [Koren, 2008]. 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 μ 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 [Koren, 2008] for full details.

A detailed survey of related work on ordinal modeling and recommendation systems is given in [Koren and Sill, 2011].

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

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

In general, any other rating predictor could serve for defining y_{ui}, including those whose parameters are fixed and need not be learned.

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

$$t_1 \leqslant t_2 \leqslant \cdots \leqslant t_{\mathcal{S}-1} \quad (3)$$

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

$$t_{r+1} = t_r + \exp(\beta_r) \qquad r = 1, \ldots, \mathcal{S} - 2 \quad (4)$$

Let us denote by Θ all model parameters, that is the biases and factors participating in (2) and $t_1, \beta_1, \ldots, \beta_{\mathcal{S}-2}$.

The probability of observing rating $r_{ui} = r$ is

$$P(r_{ui} \leqslant r | \Theta) = 1/\left(1 + \exp(y_{ui} - t_r^u)\right) \quad (5)$$

Learning proceeds by stochastic gradient ascent on the log likelihood of observing the training set with regularization. The reader is referred to [Koren and Sill, 2011] for a detailed derivation.

5 Empirical study

5.1 Datasets description

We evaluated OrdRec on some of the largest publicly available user rating datasets. First is the Netflix dataset [Bennett and Lanning, 2007], which was subject to extensive research as part of Netflix Prize contest. The other dataset is denoted by Y!Music-II and is based on user ratings on Yahoo! Music. Ratings were given to musical entities of four different types: tracks, albums, artists, and genres. Y!Music-II was used in the KDD-Cup'11 contest. Results on a third dataset, Y!Music-I, are omitted here but are presented in [Koren and Sill, 2011].

Both datasets are split into train, validation and test sets. The Y!Music-II dataset has 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-II dataset has 11 distinct values (0,1,…,10). All results are reported on the test sets, which were excluded from the training process. The validation set was used for setting the parameters of the evaluated algorithms.

5.2 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 favored thanks to its elegance and mathematical tractability

$$RMSE(\hat{\mathcal{R}}) = \frac{1}{|\hat{\mathcal{R}}|} \sum_{(u,i,r)\in\hat{\mathcal{R}}} (\hat{r}_{ui} - r)^2 \tag{6}$$

Despite its merits, RMSE can be quite detached from the ultimate goal of evaluating item ranking experience, since 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.

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 in 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 $\hat{r}_u$.

$$n_c^u = |\{(i,j) \mid \hat{r}_{ui} > \hat{r}_{uj} \text{ and } r_{ui} > r_{uj}\}| \tag{7}$$

We count the discordant pairs n_d^u for user u in a similar fashion.

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)

$$FCP = \frac{n_c}{n_c + n_d} \tag{8}$$

a measure that generalizes the known AUC metric into non-binary ordered outcomes.

5.3 Results

We compared OrdRec with the followings methods: (1) SVD++ [Koren, 2008], which represents a leading RMSE-oriented method; (2) RBM [Salakhutdinov *et. al.* 2007], which is aimed at likelihood maximization and can work with categorical scores.

Results on the datasets are reported in Tables 1–2.

RMSE			
Method	**f = 50**	**f = 100**	**f = 200**
SVD++	.8952	.8924	.8911
RBM	.9147	.9063	.9023
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%
OrdRec	**74.36%**	**74.50%**	**74.54%**

Table 1: 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).

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

RMSE			
Method	**f = 50**	**f = 100**	**f = 200**
SVD++	**2.4369**	**2.4347**	**2.4334**
OrdRec	2.4786	2.4708	2.4660

FCP			
Method	**f = 50**	**f = 100**	**f = 200**
SVD++	72.59%	72.42%	72.13%
OrdRec	**73.63%**	**73.83%**	**73.98%**

Table 2: 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).

We observe similar results on the Y!Music-II dataset, although SVD++ consistently yields the best results RMSE-wise. While 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 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 had very slow running times on the Y!Music-II data and therefore those experiments were not completed.

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 emphasize that the main motivations behind OrdRec are the handling of ordinal data and the generation of a probability distribution over ratings, rather than improved predictive performance.

6 Estimation of recommendation confidence

A recommender system has varying levels of confidence (or, certainty) in the different recommendations it provides. Accordingly, [McNee *et. al.* 2003] 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 measurements can impact 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. [Adomavicius *et. al.* 2007] 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.

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 various measures of the concentration of the rating probabilities.

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.

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.

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.

The user and item support (number of user and item ratings) and the standard deviation of the user and item ratings will be referred to collectively 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.

To assess the value of the OrdRec predicted rating distribution, four novel features were used. *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 $1 - \sum_s P_s P_s$.

To determine the best possible performance overall, a classifier with all features—both traditional and OrdRec-derived—was also trained.

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	<u>-0.490</u>	<u>0.698</u>
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 3: Confidence estimation results on the Netflix dataset (higher values are better)

In Table 3 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.

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. More detailed Y!Music-I results are available in [Koren and Sill, 2011].

7 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.

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

[3] 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.

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

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

[6] Y. Koren, J. Sill, OrdRec: An Ordinal Model for Predicting Personalized Item Rating Distributions *RecSys '11*, 2011.

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

[8] 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.

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

[10] 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.