JMLR: Workshop and Conference Proceedings 18:3–18, 2012 Proceedings of KDD-Cup 2011 competition

The Yahoo! Music Dataset and KDD-Cup'11

Gideon Dror GIDEONDR@YAHOO-INC.COM
Yahoo! Labs

Noam Koenigstein NOAMK@ENG.TAU.AC.IL
School of Electrical Engineering
Tel Aviv University

Yehuda Koren YEHUDA@YAHOO-INC.COM
Yahoo! Labs

Markus Weimer WEIMER@YAHOO-INC.COM
Yahoo! Labs

Editor: G. Dror, Y. Koren and M. Weimer

Abstract

KDD-Cup 2011 challenged the community to identify user tastes in music by leveraging Yahoo! Music user ratings. The competition hosted two tracks, which were based on two datasets sampled from the raw data, including hundreds of millions of ratings. The underlying ratings were given to four types of musical items: *tracks*, *albums*, *artists*, and *genres*, forming a four level hierarchical taxonomy.

The challenge started on March 15, 2011 and ended on June 30, 2011 attracting 2389 participants, 2100 of which were active by the end of the competition. The popularity of the challenge is related to the fact that learning a large scale recommender systems is a generic problem, highly relevant to the industry. In addition, the contest drew interest by introducing a number of scientific and technical challenges including dataset size, hierarchical structure of items, high resolution timestamps of ratings, and a non-conventional ranking-based task.

This paper provides the organizers' account of the contest, including: a detailed analysis of the datasets, discussion of the contest goals and actual conduct, and lessons learned throughout the contest.

1. Introduction

People have been fascinated by music since the dawn of humanity. A wide variety of music genres and styles has evolved, reflecting diversity in personalities, cultures and age groups. It comes as no surprise that human tastes in music are remarkably diverse, as nicely exhibited by the famous quotation: "We don't like their sound, and guitar music is on the way out" (Decca Recording Co. rejecting the Beatles, 1962).

Yahoo! Music has amassed billions of user ratings for musical pieces. When properly analyzed, the raw ratings encode information on how songs are grouped, which hidden patterns link various albums, which artists complement each other, how the popularity of songs, albums and artists vary over time and above all, which songs users would like to listen to. Such an analysis introduces new scientific challenges. We have created a large scale music dataset and challenged the research world to model it through the KDD Cup

2011 contest[1]. The contest released over 250 million ratings performed by over 1 million anonymized users. The ratings are given to different types of items: tracks, albums, artists, genres, all tied together within a known taxonomy. Thousands of teams participated in the contest, trying to crack the unique properties of the dataset.

2. The Yahoo! Music Dataset

2.1. Yahoo! Music Radio service

Yahoo! Music[2] was one of the first providers of personal(ized) internet music radio stations, with a database of hundreds of thousands of songs. As a pioneer in online music streaming, it influenced many subsequent services. Yahoo! Music used to be the top ranked online music site in terms of audience reach and total time spent. The service was free with some commercial advertising in between songs that could be removed by upgrading to a premium account. Users could rate songs, artists, albums and even genres on a 5 star system, or using a slider interface. These ratings were used by Yahoo! Music to generate recommendations that match the user's taste, based on either the taxonomy of items or on recommendations of other users with similar musical tastes.

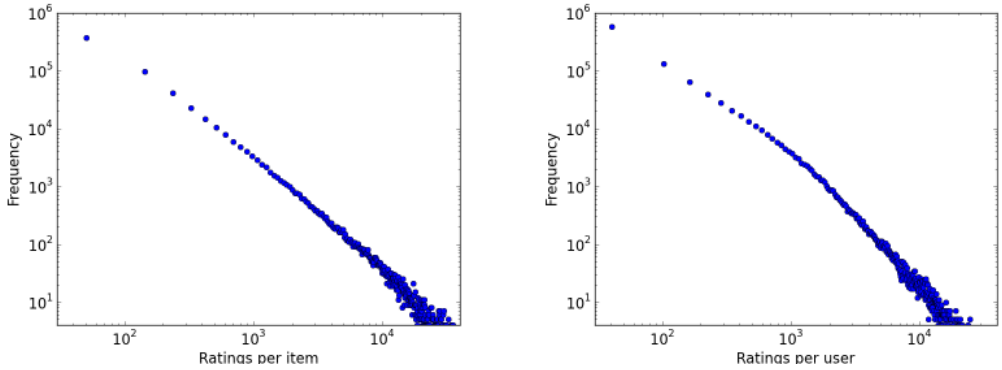
2.2. Ratings dataset

The KDD-Cup contest released two datasets based on Yahoo! Music ratings. The larger dataset was created for Track1 of the contest, and a smaller dataset was created for Track2. The two datasets share similar properties, while the dataset for Track2 omits dates and times and refers to a smaller user population. In addition, the distinctive nature of the tasks dictates using different kinds of test sets. This section presents a statistical analysis of the dataset of Track1, which is the richer and larger of the two. Since both datasets were sampled from the same source, both are characterized by many similar patterns. More details specific to compilation of the Track2 dataset are given in Sec. 3.

The Track1 dataset comprises 262,810,175 ratings of 624,961 music items by 1,000,990 users collected during 1999-2010. The ratings include one-minute resolution timestamps, allowing refined temporal analysis. Each item and each user have at least 20 ratings in the whole dataset. The available ratings were split into train, validation and test sets, such that the last 6 ratings of each user were placed in the test set and the preceding 4 ratings were used in the validation set. The train set consists of all earlier ratings (at least 10). The total sizes of the train, validation and test sets are therefore 252,800,275, 4,003,960, and 6,005,940 ratings, respectively.

Fig. 1 depicts the distribution of the number of ratings per item and number of ratings per user. Both distributions exhibit clear power law characteristics with a long tail of "popular" items and very active users, and a very large set of items and users associated with very few ratings. Indeed the sparsity factor of the dataset - 99.96% is relatively high when compared to other collaborative filtering datasets; the Netflix dataset, for example, has a 98.82% (Bennett and Lanning, 2007) sparsity factor.

1. http://kddcup.yahoo.com
2. http://music.yahoo.com



(*a*) Distribution of items (*b*) Distribution of users

Figure 1: Power-law distribution of the number of ratings per user and per item

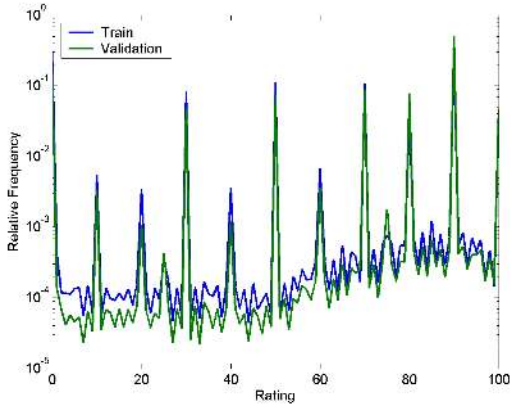
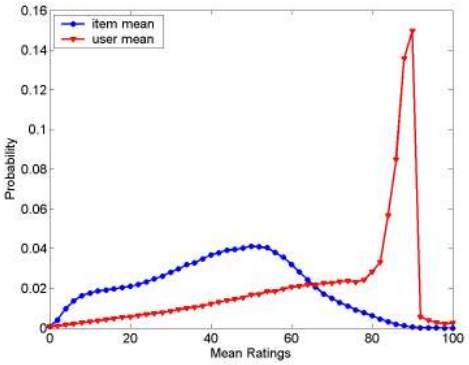
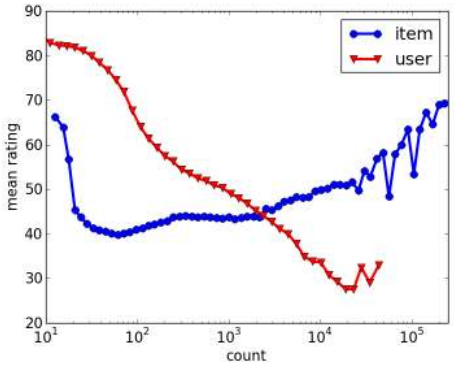


Figure 2: Rating distribution in the training and validation datasets

Fig. 2 depicts a histogram of the ratings in the train and validation sets using a logarithmic vertical scale. The vast majority of the ratings are multiples of ten. This distribution results from the different user interfaces that generally encouraged users to provide round ratings. Specifically, the popularity of a widget used to enter ratings at a 1-to-5 star scale is reflected by the dominance of the peaks at $0, 30, 50, 70$ and 90 into which the star ratings were translated. Notice that the distributions of ratings of the train and validation sets differ significantly on lower ratings. This is caused by the fact that the validation set has exactly 4 ratings per user, thus significantly down-sampling the heavy raters who are responsible for many of the lower ratings, an effect that will be detailed shortly. The different sampling also results in a significantly higher mean rating in the validation set (68.23) and the test set (68.34) compared to the training set (48.87). The different statistical characteristics of the datasets were an impediment that some participants found difficult to handle.



(a) Mean ratings distributions



(b) Mean rating vs. the number of ratings

Figure 3: Item and user rating characteristics

Fig. 3(a) depicts the distribution of mean ratings of users and items. The location of the modes (at 89 and 50 respectively) and the variances of the two distributions are quite distinct. In addition, the distribution of the mean user ratings is significantly more skewed. To understand this difference, we grouped users into sets characterized by the user activity. Each set consisted of users such that the number of items they rated falls in a certain range, e.g., between 100 and 120 ratings. For each set of users we calculated their mean rating and the mean number of ratings. We followed the same procedure for items, to produce the graphs in Figure 3(b). The graph for users (red) reveals an important feature of the Yahoo! Music dataset: the more ratings a user did, the lower her mean rating is. Specifically, "heavy" users, those who rate tens of thousands of items, have mean ratings of around 30. One possible explanation of this effect is that users who rate more items tend to encounter and rate items that don't match their musical taste, and thus the rating scores tend to be lower. This pattern explains the difference between the distribution of ratings in the training and validation sets, pronounced in lower ratings at Fig. 2. Unlike the training set, the validation and test set, equally weigh all users, and are therefore dominated by the many light-raters whose average rating is considerably higher.

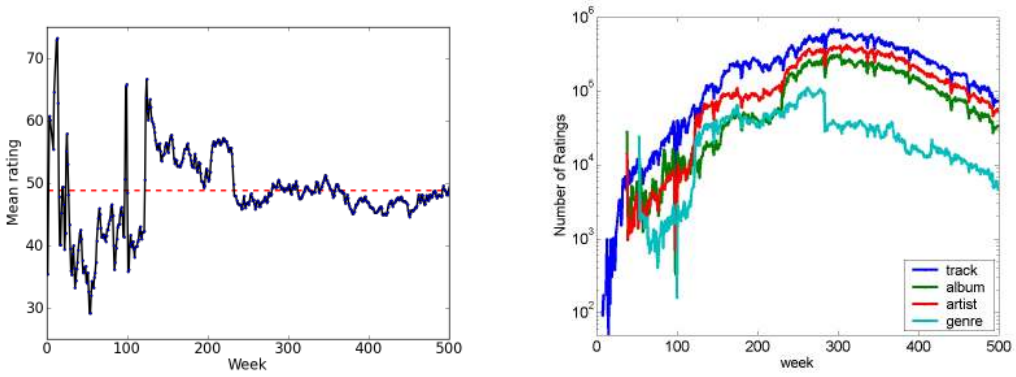
With respect to items, the picture is slightly more complicated: rare items as well as the most popular items have a relatively high mean rating, while items with medium number of ratings tend to fall considerably below the average rating.

A unique characteristic of the Yahoo! Music dataset is that user ratings are given to entities of four different types: *tracks*, *albums*, *artists*, and *genres*. Moreover, all rated items are tied together within a taxonomy. That is, for a track we know the identity of its album, performing artist and associated genres. There is exactly one artist and one album associated with each track but possibly several genres. Similarly, we have artist and genres annotation for the albums. There is no genre information for artists, as artists may switch between many genres in their career. The same song that is performed by different artists on different albums is considered as a different track. Similarly, when two or more artists collaborate they receive a new artist label that is different than each of the original labels.

We measured the correlation coefficient of ratings across the hierarchy and discovered strong ties between related items. The correlation coefficient of track ratings by the same

artist is 0.418 and 0.494 for tracks on the same album. The correlation coefficient of track ratings and their album ratings is 0.411, and 0.416 for track ratings and their artist ratings. The correlation of album ratings with ratings of their artists is 0.467 and especially correlated are ratings of albums by the same artist with a correlation coefficient of 0.608. In Sec. 5 we discuss the benefits of incorporating taxonomy into predictive models in each track.

The majority of items (81.15%) are tracks, followed by albums (14.23%), artists (4.46%) and genres (0.16%). The ratings however, are not uniformly distributed: Only 46.85% of the ratings belong to tracks, followed by 28.84% to artists, 19.01% to albums and 5.3% to genres. Moreover, these proportions may vary significantly between users. Specifically, we found that heavier raters cover more of the numerous tracks, while the light raters mostly concentrate on artists; Interestingly, all four item types have very similar mean rating, 49.17, 45.12, 51.04 and 47.83 to tracks, albums, artists and genres respectively.



(a) Mean rating vs. week (b) Number of ratings vs. week

Figure 4: Temporal statistics. Time is measured in weeks since the first rating

Several temporal phenomena affected the rating patterns of users. First, over the years the repository of items evolved considerably. Second, the service featured different rating user interfaces ("widgets") with different appearances, which have been changing over the years, encouraging different rating patterns. Thirdly, social, economical, and even technological phenomena are likely to have a non-trivial effect on the popularity of items and usage patterns of the service. Fig. 4(a) depicts the mean weekly rating value vs. the number of weeks that passed since the launch of the service. The dashed line represents the overall mean rating. We can see that the first 125 weeks are characterized by strong fluctuations in the mean rating. This is also related to a relatively low overall activity, with an average of about thirty thousand ratings per week. Later changes in the Yahoo! Music service resulted in a substantial increase in the number of ratings and the mean rating values start to stabilize. Considerable changes in the website's interface occurred around the 125-th and 225-th weeks which are expressed as noticeable changes in the mean rating in fig. 4(a).

Fig. 4(b) depicts the number of ratings per each item type (track, album, artist, genre) vs. the number of weeks that passed since the launch of the service on 1999. Noticeable changes in the distribution of types, correlated with substantial change in mean rating, can be observed around the 125-th week, when the rating activity of artists increases by

Task	#users	#items	Train	Validation	Test
Track1	1,000,990	624,961	252,800,275	4,003,960	6,005,940
Track2	249,012	296,111	61,944,406	N/A	607,032

Table 1: Datasets size statistics

an order of magnitude, and on the 225-th week, when rating activity of both albums and artists increases significantly.

3. Task description

The contest offered two different tasks. Track1 calls for predicting users' rating of musical items. Items can be tracks, albums, artists and genres. Each user and item have at least 20 ratings in the dataset (train, validation and test sets combined). A detailed description of the provided dataset was given in Sec. 2. The main dataset statistics are described in Table 1. Each user has at least 10 ratings in the training data. Then, each user has exactly four ratings in the validation data, which come later in time than the ratings by the same user in the training data. Finally, the test data holds the last 6 ratings of each user. The contestants were asked to provide predictions for these test scores. The evaluation criterion is the Root Mean Squared Error (RMSE) between predicted ratings and true ones. Formally:

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}_1|} \sum_{(u,i)\in\mathcal{T}_1} (r_{ui} - \hat{r}_{ui})^2} \tag{1}$$

Where $(u, i) \in \mathcal{T}_1$ are all the user-item pairs in Track1 test data set, r_{ui} is the actual rating of user u to item i, and $\hat{r}_{ui}$ is the predicted value for r_{ui}. This amounts to a traditional collaborative filtering rating prediction task.

Track2 involves a less conventional task. A similar music rating dataset was compiled, covering 4 times less users than Track1. Once again, each user and item have at least 20 ratings in the dataset (train and test sets combined); see Table 1 for the main dataset statistics. The train set includes ratings (scores between 0 to 100) of both tracks, albums, artists and genres performed by Yahoo! Music users.

For each user participating in the Track2 test set, six items are listed. All these items must be tracks (not albums, artist or genres). Three out of these six items have never been rated by the user, whereas the other three items were rated "highly" by the user, that is, scored 80 or higher. The three items rated highly by the user were chosen randomly from the user's highly rated items, without considering rating time. The three test items not rated by the user are picked at random with probability proportional to their odds to receive "high" (80 or higher) ratings in the overall population. More specifically, for each user a rating is sampled uniformly from all track ratings which are 80 or higher. The track associated with the rating is selected if it has not been rated by the user and has not been already selected as an unrated item for the user. The sampling for each user terminates when three different tracks, disjoint to the tracks rated by the user, have been selected. Note that many users do not participate in this test set at all.

The goal of such a task would be differentiating high ratings from missing ones. Participants were asked to identify exactly three highly rated items for each user included in the test. The evaluation criterion is the error rate, which is the fraction of wrong predictions.

We had several objectives in designing the second track of the contest. Initially, we wanted to add a dataset of a lower scale, in order to appeal to competitors with less capable systems, which might get discouraged by the large size of the Track1 dataset. More importantly, the Track2 dataset provided us an opportunity to experiment with a less established evaluation methodology, which may better relate to real life scenarios. The proposed metric is related to the common recommendation task of predicting the items that the user will like (rather than predicting the rating value of items). Furthermore, the task of Track2 requires extending the generalization power of the learning algorithm to the truly missing entries (items that were never rated by the user), as required in real life scenarios. Finally, the way we drew negative examples (in proportion to their dataset popularity), discourages known trivial solutions to the top-K recommendation task, where most popular items are always suggested, regardless of the user taste.

We decided to exclude timestamps (both dates and times) from the Track2 dataset. Our purpose was to center the focus of the time-limited competition on a specific aspect of the problem. In particular, we assume that there are many strong short-term correlations between various actions of a user, which would greatly help in separating rated from unrated items. For example, some users have the tendency to listen to multiple tracks of the same album in a row. We wanted to encourage solutions catering to the longer-term user taste, rather than analyzing short term trends, thus we hid the temporal information in Track2.

At both tracks, the test set was internally split into two equal halves known as Test1 and Test2, in a way not disclosed to the competitors. Participants were allowed to submit their test set predictions every 8 hours, and receive feedback on their Test1 performance that was reflected in the public leader-board. The true rankings of competitors were based on Test2 scores, which were not disclosed during the competition.

Comparison to real world evaluation Real world systems usually perform recommendations in two phases: The first phase is the *modeling* phase, in which a prediction model is trained to capture the patterns in the dataset. The model's quality is usually measured using an evaluation criteria such as RMSE, precision at K or error rate, similar to the performance measure used in Track2. The second phase is *retrieval of recommendations* in which a retrieval algorithm is developed, sometimes with the help of marketing experts, which uses the model to determine which recommendations should be presented to the users at each time. The most naive retrieval algorithm ranks the items for each user using the predicted rating and recommends her the top ranking items. However, business-wise it is usually better to diversify the set of recommended items and to control how often an item can be recommended. This comes at the cost of choosing items that are not ranked highest, but are still liked by the user. The overall quality of a real world systems is usually evaluated by some Key Performance Indicator (KPI) such as the clickthrough rate or purchase rate of recommended items. This of course, is very difficult to reconstruct in a competition framework.

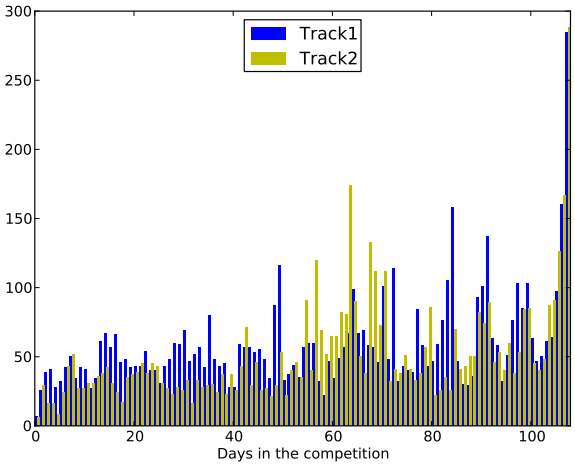


Figure 5: Submissions per day

4. Contest conduct

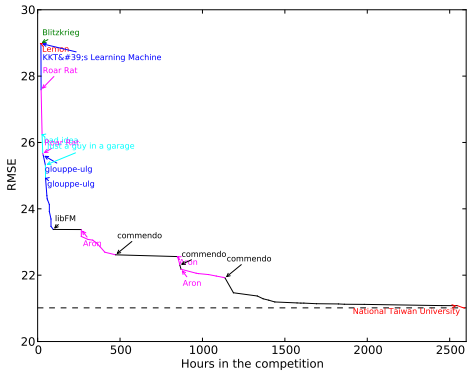
The contest took part over a period of three and a half months from March 15 till June 30, 2011. The datasets were made available to contestants on the first day of the competition. In addition, the contestants could register to the contest and have access to sample data to prepare their software two weeks in advance of the beginning of the competition.

During the competition, each team was allowed to submit one solution every eight hours for each track of the competition. The submission system provided the contestants with immediate feedback on the performance of the submitted solution on the test set. The contest website also featured a public leader-board that showed an up-to-date ranking of the best submissions so far (based on scores on the Test1 set).

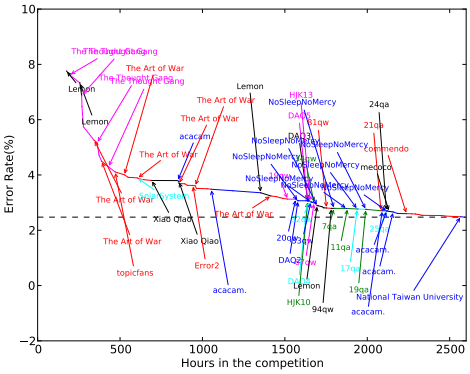
The registration system required users to form teams and to be part of only one team at a time. Teams could leave the competition by marking themselves as inactive. The contestants made heavy use of that functionality, but not to actually leave the contest. Instead, it was used mostly to regroup: Out of the 2389 users that registered for the contest, 2100 were still part of an active team at the end of the competition. During the competition, 4252 teams were registered (more than users!) of which 1287 were active at the end of the competition.

Each of the two contest tracks had about the same number of teams submitting to it: 1878 teams ever submitted to Track1, 1854 to Track2. The same is not true for the number of submissions. During the competition, there were 6344 solutions submitted to Track1, while Track2 received "only" 5542 submissions. Table 2 shows a histogram of the number of submissions per track and how many teams submitted that many times. Fig. 5 shows the number of submissions per day and track. As can be clearly seen from this plot, the contestants were eager to (re-)submit their best solutions on the last day of the competition.

The almost equal number of submissions and teams submitting is somewhat surprising, because the leadership of Track2 was much more volatile than that of Track1 as can be seen in Fig. 6(a) and 6(b). During the competition, the top spot on the public leader-board



(*a*) Track1



(*b*) Track2

Figure 6: Leadership and Performance over time.

changed 52 times for Track1 and almost twice as often, 96 times, for Track2. Even more drastically, only 12 teams ever held the top spot in Track1 during the competition, while 35 teams held that position for some period of time in Track2. We believe that this greater volatility in the leadership of Track2 can be attributed to the novel task to be solved in that track of the competition.

According to the rules of the competition, the last submission of each team was used for determining the winners, not the best. Thus, the question arises whether all teams managed to submit their best solution in time or whether an earlier submission actually would have won. Much to the relief of the teams involved, the order of the top spots in the leader-board would be the same if we would have used the best submission. Similarly, for the top spots, the publicly reported rankings based on the Test1 set, equal the ranking based on the undisclosed Test2 set, meaning that contestants did not strongly over-fit the Test1 set. As becomes apparent from Figures 7(*a*) and 7(*b*), the final submissions from all teams participating till the end are also fairly spread-out in terms of performance.

Submissions	Teams Track1	Teams Track2
0-9	1728	1738
10-19	80	65
20-29	36	29
30-39	13	2
40-49	8	7
50-59	4	6
60-69	0	1
70-79	4	1
80-89	1	1
90-99	2	2
100-109	0	1
110-119	1	0
120-129	1	1

Table 2: Submissions statistics: Number of submissions per team

Track1		
Team	Rmse-Test1	Rmse-Test2
NTU	21.0147	21.0004
Commendo	21.0815	21.0545
InnerPeace	21.2634	21.2335

Track2		
Team	Error-Test1	Error-Test2
NTU	2.4749	2.4291
Lemon	2.4808	2.4409
Commendo	2.4887	2.4442

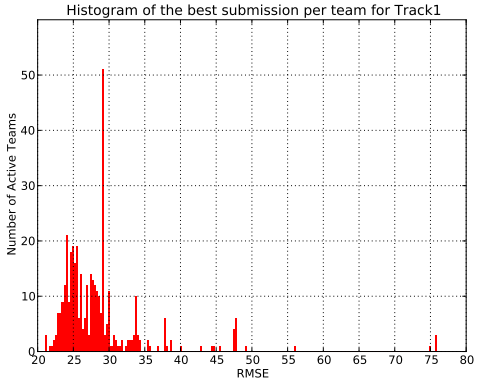
Table 3: The winning teams and their performances on Test1 and Test2. NTU is an acronym for "National Taiwan University" and Lemon is a shorthand for "The Art of Lemon". Full details on the winning teams can be found in the contest website.

Table 3 details the three winning teams on Track1 and on Track2, together with their performances on Test1 and Test2. We observe that the performance of all winners on both tracks is better on Test2 than on Test1.

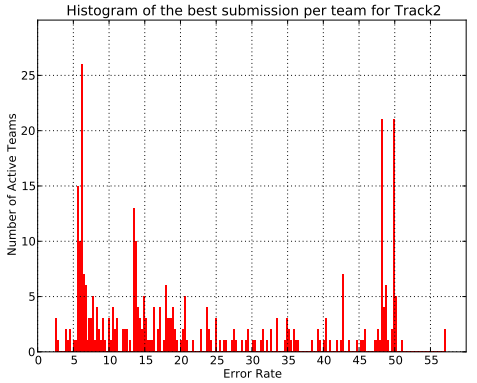
5. Lessons

This JMLR volume includes 13 papers describing the techniques employed by top teams. Short versions of these papers were previously published in the KDD-Cup'11 workshop. These papers describe many interesting algorithms, which enhance different recommendation methods. We encourage the reader to look at these papers for more details. Here, we will highlight some of the higher-level lessons that we take away from the competitors' works.

The Track1 solutions followed much of the techniques that were found successful at the Netflix Prize competition (Bennett and Lanning, 2007), which aimed at the same RMSE metric. In order to provide a proper context, we will shortly describe the related Netflix Prize setup. Netflix published a dataset including more than 100 million movie ratings that were performed by about 480,000 users on 17,770 movies. Each rating is an integer



(a) Track1



(b) Track2

Figure 7: Histogram of the final submissions' performance.

from one (worst) to 5 (best) and is associated with a date. The Netflix Prize challenge called competitors to predict 2.8 million test ratings, with accuracy judged by RMSE. The winning solution could yield a test RMSE of 0.8556 concluding a massive three year effort.

In a sense, reported Track1 results reinforce many findings on the Netflix data, which still hold despite the usage of a relatively different dataset. Solutions were created by blending multiple techniques, including nearest neighbor models, restricted Boltzmann machines and matrix factorization models. Among those, matrix factorization models proved to perform best. Modeling temporal changes in users' behavior and items' popularity was essential for improving solution quality. Among nearest neighbors models, item-item models outperformed user-user models. Such a phenomenon is notable due to the fact that the number of items in our dataset is large and roughly equals number of users, hence one could have expected that in such a setup user-user and item-item techniques will deliver similar performance. The given item taxonomy helped in accounting for the large number of sparsely rated items, however its effect was relatively low in terms of RMSE reduction.

The best result achieved on Track1 was RMSE=21 (Chen et al., 2011a; Jahrer and Toscher, 2011), which was achieved by an ensemble blending many solutions. The best

result by a single (non-blended) method was RMSE=22.12 (Chen et al., 2011b). This pretty much confirms our pre-contest expectations based on Netflix Prize experience. Note that Netflix Prize results need to get calibrated in order to account for the different rating scales. While in the Netflix Prize the ratings range (difference between highest (=5) and lowest (=1) rating) is 4, in our dataset the ratings range is 100. Hence, if one linearly maps our ratings to the 1–5 stars scale, RMSE of the same methods will shrink by a factor of 25. This means that on Netflix score range, the best score achieved at Track1 is equivalent to 0.84, which is strikingly close to the best score known on the Netflix dataset (0.8556). Comparing the fraction of explained variance (known as R^2), reveals more of a difference between Track1 and Netflix Prize. The total variance of the ratings in the Netflix test set is 1.276, corresponding to an RMSE of 1.1296 by a constant predictor. Three years of multi-team concentrated efforts reduced the RMSE to 0.8556, thereby leaving the unexplained ratings variance at 0.732. Hence, the fraction of explained variance is $R^2 = 42.6\%$, whereas the rest 57.4% of the ratings variability is due to unmodeled effects (e.g., noise). Moving to the Yahoo! Music Track1 test set, the total variance of the test dataset is 1084.5 (reflecting the 0-100 rating scale). The best submitted solution could reduce this variance to around 441, yielding $R^2 = 59.3\%$. Hence, with the Yahoo! Music dataset a considerably larger fraction of the variance is explainable by the models.

Track2 of the competition offered a less traditional metric: error rate of a classifier separating loved tracks from unrated ones. We were somewhat surprised by the low error rate achieved by the contestants — 2.47% for best solution (McKenzie et al., 2011). This means that for over 97% of test tracks, the model could correctly infer whether they are among the loved ones. However, such a performance should be correctly interpreted in light of real life expectations. Our chosen metric contrasted the same amount of positive ("loved items") and negative examples. In real life, one would argue that positives and negatives are not balanced, but rather negatives grossly outnumber positives. This makes the task of identifying the "top-k" items harder.

In comparison to other ranking-oriented metrics, the employed error rate offers two benefits. First, it is fast to compute, without the need to rank positive items within a large set of background items. Second, it can be tailored to being neutral to popularity effects, which tend to greatly influence ranking-based metrics. This was achieved by sampling the negative examples by their overall popularity (i.e., popular items are sampled more frequently as negative examples), so that positive and negative examples come from the same distribution. This favors methods that go deeper than merely promoting the overall popular items. In other studies, we also experimented with a variant of the metric that picks negative examples uniformly, without over-emphasizing popular items. This tends to expose other aspects of performance, which shed additional light on the algorithms' properties.

A key technique shared by the Track2 solutions is sampling the missing ratings, to be used as negative examples for training the learning algorithm. This way, the proposed methods do not only model the observed ratings, but also those ratings sampled from the missing ones. This greatly improved performance at Track2. This also confirms recent works (Cremonesi et al., 2010; Steck, 2010) showing that models that account also for the missing ratings provide significant improvements over even sophisticated models that are optimized on observed ratings only. We note that as dictated by the test set balanced structure, competitors resorted to balanced sampling. That is, the number of sampled

missing ratings equals to the number of non-missing ratings for the same user. This also bodes well with practices used when learning unbalanced classification models, where often training is performed on a balanced number of positive and negative examples. Yet, we would like to explore what would have happened had the objective function been different, such as the recall@K metric, which is not utilizing an equal number of positive and negative examples. Would we still want to construct a balanced training set with equal number of positive and negative examples per user, or maybe we would like to use the full set of missing values as in (Cremonesi et al., 2010; Hu et al., 2008; Steck, 2010)?

The Track2 dataset did not expose timestamps for reasons explained in Sec. 3. However, in real life where such timestamps are available and considered highly beneficial (see, e.g., Lai et al. (2011)), the way negative examples are sampled becomes non-trivial and raises the question of how would one attach a timestamp to the sampled negative examples. Given the importance of sampling methods at implicit-feedback recommenders, we believe that this is an important question deserving further research.

Another theme shared by many Track2 solutions is the usage of pairwise ranking objective functions. Such schemes do not strive to recover the right score for each item, but rather maintain the correct ranking for each pair of items. Pairwise ranking objective seems more natural and elegant in our ranking-oriented setup, where item ordering matters, not individual scores. A notable recommendation algorithm of this family is BPR (Rendle et al., 2009), which was indeed employed by several teams. In fact, the best reported result by a single (non-blended) method (error rate of 2.911%) is by an extension of BPR (Mnih, 2011). Yet, in future research we would like to further experiment with the value of pairwise ranking solutions, compared to the arguably cheaper regression-based solutions, which try to recover each single user-item score (Hu et al., 2008; Steck, 2010). In fact, the top two leading teams on Track2 (Lai et al., 2011; McKenzie et al., 2011) utilized regression techniques which minimize squared loss of recovered user-item scores. Their single-method results were quite close to those reported by pairwise ranking techniques.

Comparing results of Track1 and Track2 reinstates the known observation that achievable RMSE values on the test set lie in a quite compressed range. This was also evident at the Netflix contest where major modeling improvements could only slightly reduce RMSE. This observation becomes much more striking when considering comparable improvements on the error rate metric applied at Track2. Let us analyze the influence of two components that improve modeling accuracy.

1. **Dimensionality of the matrix factorization model** On the RMSE front it was observed that after reaching 50-100 factors, there is barely an improvement when increasing the number of factors; see, e.g., Figure 7 of (Dror et al., 2011). However, the case is very different with Track2, where the error rate metric was used. Competitors have resorted to very high dimensionalities of matrix factorization models in order to improve performance. For example, McKenzie et al. (2011)(Table 2) reports error rate dropping from 6.56% to 4.25% when increasing dimensionality from 800-D to 3200-D.

2. **Utilizing the given item taxonomy** We have observed a quite subtle RMSE reduction from around 22.85 to 22.6 by utilizing the taxonomy; see (Dror et al., 2011). Similar RMSE reduction is also reported in Table 3 of Chen et al. (2011b). When moving to the error rate metric, utilization of the taxonomy results in much more

significant improvements. Lai et al. (2011) reports a matrix factorization model (BinarySVD) achieving around 6% error rate, which is significantly reduced to 3.49% by considering the items taxonomy (BinarySVD+). Similarly, Mnih (2011)(Table 3) reports reducing the error rate from 5.673% to 3.314% by accounting for taxonomy in a latent factor model.

In conclusion, we have found a consistent evidence that the same modeling improvements that only modestly impact RMSE, have a substantial effect on the error rate metric. Similar evidence was also hinted elsewhere (Koren, 2008). Given the large popularity of the RMSE metric in recommenders research, it is important to interpret RMSE improvements well. Clearly, small RMSE changes can correspond to significantly different ranking of items. For example, RMSE minimizing algorithms tend to diminish variance by concentrating all predictions close to averages, with small deviations from the average dictating item rankings. We believe that the impact of RMSE reduction on user experience is a topic deserving further exploration.

5.1. Maneuvers by the contestants

While the contest was clearly decided by algorithmic tools, we would like to highlight some interesting strategies used by the competitors that were tailored to the specific setup of the contest.

Recall that the test set was split into two halves: Test1 and Test2, whereas Test1 results were exposed on the Leader-board and Test2 results were used for determining the winners and remained undisclosed during the contest. Contestants have used the reported Test1 results in order to tweak their solutions. This was particularly doable for Track1 that used the RMSE metric. It can be shown that once observing RMSE of individual predictors, their optimal linear combination can be derived. Hence, contestants have created a linear combination of individual submissions based on their Leaderboard-reported Test1 RMSE values. Interestingly, Balakrishnan et al. (2011) report using a variant of this technique even for the error-rate metric employed at Track2. In essence, the Test1 set partially served as an additional validation set. It is hard to evaluate the extent to which such a technique was significant given that the best performing blending was achieved by non-linear combinations where such an RMSE-blending cannot work. It is important to note that the contestants have used this practice moderately and did not over-fit the Test1 set. In fact, final ranking of top teams remain the same whether scored by Test1 or by Test2.

We also observed some competitors hiding their true performance on Track2 by making inverted submissions (where predicted positives and negatives are exchanged). The resulting "flipped" submissions was scored exactly 100% minus the error-rate of the original predictor, and hence would land close to the bottom of the Leaderboard. We are not certain as to why teams have employed this practice, but it could have been avoided by always reporting the lower error-rate among the actual submission and its inversion.

Finally, Lai et al. (2011) have nicely found that ratings in Track2 are arranged in a chronological order, which allowed them to use temporal information which was not explicitly available for Track2. This has proved very significant to reducing their error rate.

References

Suhrid Balakrishnan, Rensheng Wangand Carlos Scheidegger, Angus MacLellan, Yifan Hu, Aaron Archer, Shankar Krishnan, David Applegate, Guang Qin Ma, and S. Tom Au. Combining predictors for recommending music: the false positives' approach to kdd cup track 2. In *KDD-Cup'11 Workshop*, 2011.

J. Bennett and S. Lanning. The netflix prize. In *Proc. KDD Cup and Workshop*, 2007.

Po-Lung Chen, Chen-Tse Tsai, Yao-Nan Chen, Ku-Chun Chou, Chun-Liang Li, Cheng-Hao Tsai, Kuan-Wei Wu, Yu-Cheng Chou, Chung-Yi Li, Wei-Shih Lin, Shu-Hao Yu, Rong-Bing Chiu, Chieh-Yen Lin, Chien-Chih Wang, Po-Wei Wang, Wei-Lun Su, Chen-Hung Wu, Tsung-Ting Kuo, Todd G. McKenzie, Ya-Hsuan Chang, Chun-Sung Ferng, Chia-Mau Ni, Hsuan-Tien Lin, Chih-Jen Lin, and Shou-De Lin. A linear ensemble of individual and blended models for music rating prediction. In *KDD-Cup'11 Workshop*, 2011a.

Tianqi Chen, Zhao Zheng, Qiuxia Lu, Xiao Jiang, Yuqiang Chen, Weinan Zhang, Kailong Chen, Yong Yu, Nathan N. Liu, Bin Cao, Luheng He, and Qiang Yang. Informative ensemble of multi-resolution dynamic factorization models. In *KDD-Cup'11 Workshop*, 2011b.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proc. 4th ACM Conference on Recommender Systems(RecSys'10)*, pages 39–46, 2010.

Gideon Dror, Noam Koenigstein, and Yehuda Koren. Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In *Proc. 5th ACM Conference on Recommender Systems (RecSys'11)*, 2011.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proc. 8th IEEE Conference on Data Mining (ICDM'08)*, pages 263–272, 2008.

Michael Jahrer and Andreas Toscher. Collaborative filtering ensemble. In *KDD-Cup'11 Workshop*, 2011.

Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. 14th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'08)*, pages 426–434, 2008.

Siwei Lai, Liang Xiang, Rui Diao, Yang Liu, Huxiang Gu, Liheng Xu, Hang Li, Dong Wang, Kang Liu, Jun Zhao, and Chunhong Pan. Hybrid recommendation models for binary user preference prediction problem. In *KDD-Cup'11 Workshop*, 2011.

Todd G. McKenzie, Chun-Sung Ferng, Yao-Nan Chen, Chun-Liang Li, Cheng-Hao Tsai, Kuan-Wei Wu, Ya-Hsuan Chang, Chung-Yi Li, Wei-Shih Lin, Shu-Hao Yu, Chieh-Yen Lin, Po-Wei Wang, Chia-Mau Ni, Wei-Lun Su, Tsung-Ting Kuo, Chen-Tse Tsai, Po-Lung Chen, Rong-Bing Chiu, Ku-Chun Chou, Yu-Cheng Chou, Chien-Chih Wang, Chen-Hung

Wu, Hsuan-Tien Lin, Chih-Jen Lin, and Shou-De Lin. Novel models and ensemble techniques to discriminate favorite items from unrated ones for personalized music recommendation. In *KDD-Cup'11 Workshop*, 2011.

Andriy Mnih. Taxonomy-informed latent factor models for implicit feedback. In *KDD-Cup'11 Workshop*, 2011.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proc. 25th Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI'09)*, pages 452–461, 2009.

Harald Steck. Training and testing of recommender systems on data missing not at random. In *Proc. 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'10)*, pages 713–722, 2010.