A Linear Ensemble of Individual and Blended Models for Music Rating Prediction

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, Shou-De Lin

{r99922038, r98922028, r99922008, r99922095, b97018, b97705004, b96018, b96115, b96069, b96113, b95076, b97114, b97042, d98922007, b97058, b96110, b96055, d97944007, d97041, b96025, r99922054, b96092, htlin, cjlin, sdlin}@csie.ntu.edu.tw

Department of Computer Science and Information Engineering,
National Taiwan University

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

Abstract

Track 1 of KDDCup 2011 aims at predicting the rating behavior of users in the Yahoo! Music system. At National Taiwan University, we organize a course that teams up students to work on both tracks of KDDCup 2011. For track 1, we first tackle the problem by building variants of existing individual models, including Matrix Factorization, Restricted Boltzmann Machine, k-Nearest Neighbors, Probabilistic Latent Semantic Analysis, Probabilistic Principle Component Analysis and Supervised Regression. We then blend the individual models along with some carefully extracted features in a non-linear manner. A large linear ensemble that contains both the individual and the blended models is learned and taken through some post-processing steps to form the final solution. The four stages: individual model building, non-linear blending, linear ensemble and post-processing lead to a successful final solution, within which techniques on feature engineering and aggregation (blending and ensemble learning) play crucial roles. Our team is the first prize winner of both tracks of KDD Cup 2011.

1. Introduction

KDDCup 2011 contains two tracks that are related to predicting user rating behaviors in Yahoo! Music via collaborative filtering. At National Taiwan University, similar to what was done for KDDCup 2010 (Yu et al., 2011), we organize a course for the competition. Our members include three instructors, three TAs and 19 students. During the competition, students are free to actively work on the track(s) of their choice, while the instructors and TAs encourage the whole team to devote sufficient efforts on both tracks. This paper summarizes our work on track 1 of the competition.

The data set of track 1 is collected over an 11-year span and contains 262,810,175 examples. Each example can be viewed as a vector $(u, i, r_{ui}, t_{ui}, \tau_{ui})$ where u denotes the user ID, i denotes the item ID, r_{ui} is the rating given to item i by user u, t_{ui} is the date that the rating is given and τ_{ui} represents the time (to minute resolution) at which the rating is given. The ratings r_{ui} are integers between 0 and 100. The data set has been split into three subsets—training (252,800,275 examples), validation (4,003,960 examples) and test (6,005,940 examples). The test set consists of the last 6 ratings from each user; the validation set consists of the last 4 ratings earlier than the test-set ones per user, and the training set takes the rest (earliest ones). Ratings in the test set are hidden from the contestants and the goal of track 1 is to predict those hidden ratings accurately. During the competition, a team is allowed to submit the predictions of the test set every eight hours. Then, the root-mean-squared-error (RMSE) on a fixed-but-unknown random half of the test set is reported back to the team and shown on the leaderboard (called Test1 RMSE); the RMSE on the other half (called Test2 RMSE) is the final judging criterion that determines the winner of the competition. Because of the setting, using the Test1 RMSE to tune the prediction models is both slow (once every eight hours) and dangerous (can cause overfitting). Thus, teams are expected to use the validation set, because of its relative closeness to the test set, as a local proxy of evaluating model performance. In addition to the examples, taxonomy information between items, which can be one of genre, artist, album and track, is also provided. More details on the data set can be found in the official description (Dror et al., 2011).

The (u, i, r_{ui}, t_{ui}) parts of the data set of track 1 are similar to the one provided by the well-known Netflix competition[1] , and the judging criterion is also similar. Nevertheless, three properties make the task of KDDCup track 1 different from the task of the Netflix competition. First, the number of examples, the number of users and the number of items are much larger. Second, there is a new piece of information: the taxonomy between items. Third, there are more details on the rating time (the time τ_{ui}) and the long time span of the data set seems to include several major changes of the data collecting system.

The strategy of our team is to first study the mature individual models in the rich literature of collaborative filtering (Töscher and Jahrer, 2009; Hofmann, 2004; Koren and Bell, 2011; Louppe, 2010). During our studies, we brainstorm for ideas that improve the model by respecting the three properties above. For instance, many of the models require modification to conquer the computational bottleneck on the larger data set and some of the models can include taxonomy and time information properly to improve the performance.

Blending (or ensemble) techniques are shown to be useful in many past competitions (Yu et al., 2011; Töscher and Jahrer, 2009). To carry out the full power of blending, two types of individual models are used in our blending procedure: the first type is learned from the training set only and the second type is learned from both the training and the validation sets. The first type then goes through another stage, non-linear blending using the validation set, to produce more powerful models that exploit the strength of different individual models for different kinds of examples. The individual and validation-set-blended models are then combined linearly using the test-set blending technique (Töscher and Jahrer, 2009) to produce an ensemble that is well-aligned with the distribution of test examples. The final solution applies from some post-processing steps on the resulting ensemble, where the steps come from our observations on the data collecting system. We find that many of the

[1] http://www.netflixprize.com/

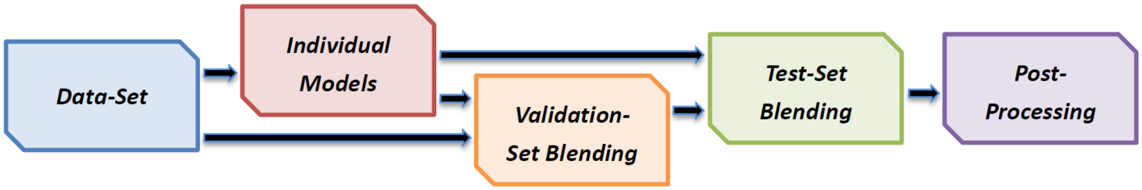

Figure 1: Framework setting of our strategy

individual models readily reach promising results, both validation-set blending and test-set blending add significant boosts to the performance, and post-processing provides a minor but consistent improvement. The visualized framework setting described above is as shown by Figure 1.

The paper is organized as follows. The six categories of individual models and our ideas on their improvements are introduced in Section 2. Section 3 discusses the validation-set blending and Section 4 describes the test-set blending. We illustrate the steps of post-processing in Section 5 and conclude in Section 6.

2. Individual Models

In this section, we introduce six categories of individual models that are studied by our team. Each category includes basic formulation, some variant model formulations adopted in our final solution and our novel ideas for potential improvements.

In addition to denoting an example by $(u, i, r_{ui}, t_{ui}, \tau_{ui})$, we use $\hat{r}_{ui}$ to indicate the predicted rating of the user-item pair (u, i) using a given model. The terms u and v are reserved for user indices and the terms i and j are reserved for item indices. The data set taken to learn the individual model is denoted by $\mathcal{D}$, which can be either the training set or a combination of the training and validation sets. The set of items that have been rated by user u in $\mathcal{D}$ is called $\mathcal{D}(u)$ and the set of users who have rated item i in $\mathcal{D}$ is called $\mathcal{D}(i)$. Other locally-used notations are specified in each subsection.

2.1. Matrix Factorization

Matrix Factorization (MF) is a popular model in building collaborative filtering systems (Koren et al., 2009; Bell et al., 2008; Töscher and Jahrer, 2008; Ma, 2008; Piotte and Chabbert, 2009). The key idea behind MF is to approximate the rating r_{ui} as the inner-product of a length-k vector $\mathbf{p}_u$ of user factors and another vector $\mathbf{q}_i$ of item factors, where k is a pre-specified parameter. In other words, the prediction formula of MF is

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i. \tag{1}$$

Consider a matrix $\mathbf{P}$ that contains the vectors $\mathbf{p}_u$ and another matrix $\mathbf{Q}$ that contains $\mathbf{q}_i$. In the basic formulation of MF, the two matrices are learned by minimizing the least-squared error between r_{ui} and $\hat{r}_{ui}$:

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{(u,i)\in\mathcal{D}} (r_{ui} - \hat{r}_{ui})^2 \text{ subject to (1).}$$

A modern optimization algorithm for solving the minimization problem is Stochastic Gradient Descent (SGD) (Koren et al., 2009), as shown in Appendix D. We also explore the use of an established MF solver from GraphLab[2] in parts of our solution.

Six variants that improve the basic MF are commonly used in previous works and are included in our solution.

1. Regularized MF adds regularization terms $\lambda_P||\mathbf{p}_u||^2$ and $\lambda_Q||\mathbf{q}_i||^2$ to each squared error term $(r_{ui} - \hat{r}_{ui})^2$ to prevent overfitting, where λ_P and λ_Q are the regularization parameters.

2. Biased MF changes (1) to $\hat{r}_{ui} = \bar{r} + \mu_u^{\text{user}} + \mu_i^{\text{item}} + \mathbf{p}_u^T\mathbf{q}_i$ and learns not only $\mathbf{p}_u$ and $\mathbf{q}_i$ but also the user-bias μ_u^{user} and the item-bias μ_i^{item} with SGD, where $\bar{r}$ is the average rating 48.6893 of the training set.

3. Binned Biased MF further extends Biased MF by considering μ_u^{user} and μ_i^{item} that can vary with different date regions in which the rating is made.

4. Biased MF with Time Deviation is another extension of Biased MF that predicts with

$$\hat{r}_{ui} = \bar{r} + \mu_u^{\text{user}} + \mu_i^{\text{item}} + \mathbf{p}_u^T\mathbf{q}_i + \sigma_i \cdot \frac{\delta}{\delta + (t_{ui} - t_i^{\text{begin}})}$$

   and learns additional variables σ_i during training. Here t_i^{begin} denotes the date when item i was first rated. In our solution, parameter δ is selected as 300 based on the validation performance of a few choices.

5. Compound SVD (Ma, 2008) restricts $\mathbf{p}_u$ to be a linear combination of $\mathbf{q}_i$ approximately.

6. Non-negative Matrix Factorization (Lee and Seung, 2001) restricts the value in each component of $\mathbf{p}_u$ and $\mathbf{q}_i$ to be non-negative.

Variants of MF often contain more parameters to choose. For the task of choosing appropriate parameter combinations, we adopt an existing automatic parameter tuner (APT) method (Töscher and Jahrer, 2008). A 150-feature Biased MF tuned by APT can reach 23.0667 in Test1 RMSE.

In addition to the six variants, we apply novel ideas to improve the MF models for this competition, as discussed below.

[Add Day/Hour Bias Terms]

We observe that users can present different rating behaviors between weekdays and weekends. For instance, the number of ratings is halved on weekends. Inspired by this observation, we change from Biased MF to $\hat{r}_{ui} = \bar{r} + \mu_u^{\text{user}} + \mu_i^{\text{item}} + \mathbf{p}_u^T\mathbf{q}_i + \mu_{d,h}^{\text{time}}$, where the date indices $d \in \{0, \cdots, 6\}$ and the hour indices $h \in \{0, \cdots, 23\}$, to capture the basic trend of the different rating behavior in different time slots of the week. The new formulation is similar to Binned Biased MF but considers bins with time slots (of the week) rather than

[2] http://graphlab.org/

date regions. A 500-feature binned Biased MF model with time deviation and day/hour bias terms achieves 22.9022 in Test1 RMSE.

[Kernelized MF]
The prediction formula $\mathbf{p}_u^T\mathbf{q}_i$ is an inner product. Following the rich literature in kernel methods (Schölkopf and Smola, 2001), we explore the possibility of using a generalized inner product (kernel) during factorization. In particular, $\mathbf{p}_u^T\mathbf{q}_i$ is replaced with $K(\mathbf{p}_u, \mathbf{q}_i)$, where K can be any kernel function. We consider three different kernel functions with parameters (α, β) chosen by validation performance:

- stump (Lin and Li, 2008): $K(\mathbf{p}_u, \mathbf{q}_i) = \beta - \alpha \|\mathbf{p}_u - \mathbf{q}_i\|_1$
- power-2: $K(\mathbf{p}_u, \mathbf{q}_i) = \beta - \alpha \|\mathbf{p}_u - \mathbf{q}_i\|_2^2$
- summed Gaussian: $K(\mathbf{p}_u, \mathbf{q}_i) = \sum_{\kappa=1}^{k} \beta \cdot e^{-\alpha \cdot (p_{u\kappa} - q_{i\kappa})^2}$

Some other kernel functions such as the standard Gaussian are tested but not included in the final solution. The novel Kernelized MF can be solved by SGD like the basic MF.

An internal experiment shows for a Kernelized MF with validation RMSE 21.4632 and a Biased MF which reaches 21.3391. A linear combination of the two achieves 21.2714. The result suggests that Kernelized MF can be blended with Biased MF to achieve better performance.

[Ordinal Regression MF (ORMF)]
The formulation respects the ordinal nature of the ratings rather than directly treating them as real-valued labels. ORMF adopts similar reduction steps from ordinal regression to metric regression (Li and Lin, 2006; Ruan et al., 2011), by using square function as the mislabeling cost. In particular, for this competition, ORMF considers thresholds θ between $\{5, 15, 25, \cdots, 95\}$. For every threshold θ, ORMF transforms the original data set $\{(u, i, r_{ui})\}$ to a binary classification data set $\left\{(u, i, y_{ui}^{(\theta)})\right\}$ with $y_{ui}^{(\theta)} = 1$ if $r_{ui} \geq \theta$, and 0 otherwise. Then, ORMF solves the following weighted and regularized MF problem on the transformed data set:

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{(u,i)\in\mathcal{D}} \left| r_{ui} - \theta \right| \left(y_{ui}^{(\theta)} - \hat{y}_{ui}^{(\theta)} \right)^2 + \lambda_P ||\mathbf{p}_u||^2 + \lambda_Q ||\mathbf{q}_i||^2$$

$$\text{subject to} \quad \hat{y}_{ui}^{(\theta)} = \mathbf{p}_u^T \mathbf{q}_i.$$

Each $\hat{y}_{ui}^{(\theta)}$ can be viewed as an estimator of $P(r \geq \theta | u, i)$ and thus $\hat{r}_{ui}$ can be reasonably computed by $10 \sum_{\theta=5,15,\cdots,95} \hat{y}_{ui}^{(\theta)}$, the expected rank over $P(r|u, i)$ (Ruan et al., 2011). A similar approach with a slightly different objective function and a different optimization algorithm can be found in Weimer et al. (2008).

Some experimental results of ORMF during our limited exploration are listed in Table 1. Although the results are worse than those of the used MF variants, we include them in the final ensemble solution for the purpose of diversity.

[Emphasize Newer Examples in SGD]
Because the test set contains the newest ratings for each user on the time line, we hope to

Table 1: Results of ORMF

Number of Features	$\mathcal{D}$	Test1 RMSE
20	training	24.7686
20	training + validation	23.5633

emphasize the ratings that are temporally closer to the test ratings. We have empirically observed that SGD fits the examples in the last few updates better. To utilize this property, rather than presenting the examples to SGD in a random order, we heuristically take a special fixed order instead. The ordering places the last rating from each user at the end of the sequence, preceded by the second-to-last rating from each user, etc. While the heuristic violates the stochastic assumption in the SGD derivation, it respects the time information in the provided data set and indeed reaches better performance. This technique enhances the Test1 RMSE by around 0.05 consistently on most of our MF models.

In addition to the ordering technique, we also adopt another technique that emphasizes newer examples by designing a second stage of SGD training that only includes the last four ratings from each user (see Appendix D). In the experiments, we let the second stage run for 3 epochs over those ratings (using more epochs results in overfitting). We observe an improvement by around 0.1 in the Test1 RMSE with the added second stage.

2.2. Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBM) (Hinton, 2010; Louppe, 2010) have been widely used for collaborative filtering (Salakhutdinov et al., 2007). The model takes two layers of units: the visible units that represent a binary pattern observed and the hidden units that represent a binary pattern memorized (which can be viewed as features). For the purpose of this competition, we take multinomial visible units of L levels such that for each user u, unit v_i^ℓ stands for whether the rating r_{ui} of item i is of level ℓ. For memory efficiency, we quantize r_{ui} to only $L = 11$ levels $\{0, (1 \text{ to } 10), (11 \text{ to } 20), \cdots, (91 \text{ to } 100)\}$, instead of using the original 101 levels. All possible visible units v_i^ℓ can be viewed as a long vector $\mathbf{v}$. All possible k hidden units h_j, on the other hand, can be viewed as a vector $\mathbf{h}$. The number of hidden units k acts like the number of factors in the MF model. The visible units $\mathbf{v}$ and hidden units $\mathbf{h}$ are connected by the following probability distributions.

$$\begin{aligned} P_v(v_i^\ell = 1|\mathbf{h}) &= \frac{\exp\left(b_i^\ell + \sum_{\kappa=1}^{k} h_\kappa W_{i,\kappa}^\ell\right)}{\sum_{l=1}^{L} \exp\left(b_i^l + \sum_{\kappa=1}^{k} h_\kappa W_{i,\kappa}^l\right)} \\ P_h(h_\kappa = 1|\mathbf{v}) &= \frac{1}{1 + \exp\left(-\left(b_\kappa + \sum_{i=1}^{|\mathcal{D}(u)|} \sum_{l=1}^{L} v_i^l W_{i,\kappa}^l\right)\right)} \end{aligned} \tag{2}$$

For a given user u, consider a vector $\tilde{\mathbf{v}}_u$ where each of its $\tilde{v}_i^\ell = [\![r_{ui} \text{ is of level } \ell]\!]$. The training phase of RBM feeds the vector $\tilde{\mathbf{v}}_u$ from a batch of users and uses the contrastive divergence algorithm (Salakhutdinov et al., 2007) to iteratively optimize the weights $W_{i,\kappa}^\ell$ and the bias terms b_i^ℓ and b_κ. For predicting the rating made by a user u, the vector $\tilde{\mathbf{v}}_u$ is again fed into

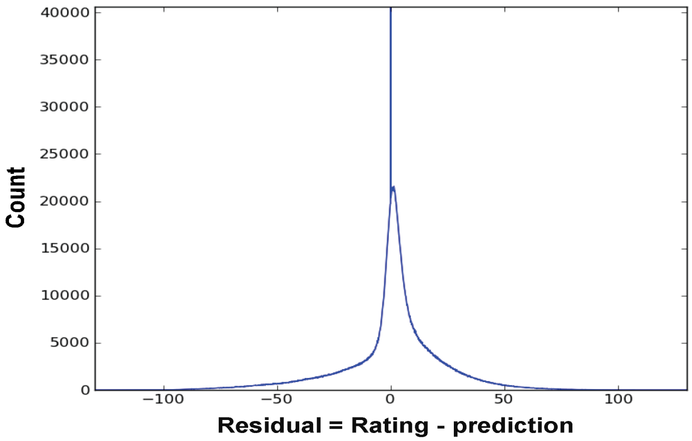

Figure 2: Residual distribution from MF model

the RBM model (with the weights and bias terms fixed) to get the output from the hidden units $\tilde{\mathbf{h}}_u$ and the visible probability distribution $P_v(v_i^\ell = 1|\tilde{\mathbf{h}}_u)$. The estimated rating $\hat{r}_{ui}$ can then be computed by taking an expectation of ℓ over P_v and scaling appropriately.

Each iteration in an RBM with 100 hidden units takes around 30 minutes on our machines and it takes around 100 iterations to converge. Thus, tuning the parameters (such as the number of features k and the learning rate) can be difficult. We follow the literature (Louppe, 2010; Hinton, 2010) and take a grid search on a reasonable range of parameters.

Two variants of the RBM model are also considered in our final solution. The first one is conditional RBM (cRBM) (Salakhutdinov et al., 2007), which couples more constraints into the RBM formulation. Another variant is factorized RBM (fRBM) (Salakhutdinov et al., 2007), which decomposes the weight matrix $\mathbf{W}$ (with one dimension indexed by (i, ℓ) and the other dimension indexed by κ) to the product of two lower-dimensional matrices for efficiency. Both variants are somewhat worse than the original RBM in our experiments, but we choose to include them in the final ensemble for the sake of diversity.

[Gaussian RBM for Residual Prediction]

For a given base model (say, MF), define its residuals as $r_{ui} - \hat{r}_{ui}$. We observe that the residuals of the MF model can be closely approximated by a Gaussian distribution (see Figure 2). Thus, we take another variant of the RBM model: the Gaussian RBM (gRBM) (Hinton, 2010), which replaces P_v in (2) with $b_i^\ell + \sum_{\kappa=1}^{k} h_\kappa W_{i,\kappa}^\ell$, under the assumption that the visible inputs are Gaussian-distributed. During training, we take either MF or RBM as the base model and use its residuals, normalized to zero mean and one standard deviation, to train a gRBM model. Then we sum the predictions from both the base model and the residual gRBM model as $\hat{r}_{ui}$. Experimental results demonstrate that gRBM is a promising model for residual prediction.

2.3. k-Nearest Neighbors

We adopt item-based k-Nearest Neighbor (k-NN) in our work (Piotte and Chabbert, 2009). The basic formulation of item-based k-NN makes predictions by

$$\hat{r}_{ui} = \frac{\sum_{j \in G_k(u,i)} w_{ij} \cdot r_{uj}}{\sum_{j \in G_k(u,i)} w_{ij}},$$

where $G_k(u,i)$ denotes the group of k nearest items to the query (u,i). The training part of k-NN needs to determine the neighborhood function G_k and assign a suitable weighting scheme w_{ij}. We only conducted few experiments for user-based k-NN, but found that the RMSE is not as good as item-based k-NN. Further, for this data set, user-based k-NN needs about two times of space and four times of time toward item-based k-NN.

One major choice of $G_k(u,i)$ is to include k items (with indices j) that have been rated by user u and of the largest w_{ij}. For the item weights w_{ij}, a baseline approach is to consider $w_{ij} = c(i,j)$ using some correlation function c that measures the similarity between i and j. During the competition, we explore the following baseline correlation functions that have been popularly used in existing works: Common Support, Pearson, Pearson with Fisher Transform, Mean Squared Error, Common User Support, Set, and Residual (Koren and Bell, 2011; Piotte and Chabbert, 2009; Töscher and Jahrer, 2009). The detailed formula of the correlation functions can be found in Appendix E.

We also consider two existing variations of k-NN to improve performance.

1. Time Decay k-NN shrinks the weight of the ratings that are far away on the time line. That is,

$$w_{ij} = \frac{c(i,j)}{1 + \alpha |t_{ui} - t_{uj}|},$$

with a fixed $\alpha = 10$ selected using the validation set.

2. Rating Decay k-NN regularizes the prediction by adding a constant value β during averaging, to shrink the prediction according to the value of $\sum_{j \in G_k(u,i)} w_{ij}$.

$$\hat{r}_{ui} = \frac{\sum_{j \in G_k(u,i)} w_{ij} \cdot r_{uj}}{\sum_{j \in G_k(u,i)} w_{ij} + \beta}.$$

The parameter β is selected by validation performance for each different weighting scheme.

At the beginning of the competition, we conduct experiments on k-NN using the original ratings, but find that the performance is unsatisfactory. Thus, we apply k-NN on the residuals of other models instead of on the original ratings, similar to what is done for gRBM. The resulting residual k-NN are coupled with MF and RBM models in our final solution. Table 2 shows how a particular k-NN using Pearson correlation with Fisher Transform can improve MF and RBM models significantly. Please refer to Appendix E for more results.

In addition to the existing variants, we study three new directions on improving the k-NN model, as discussed below.

Table 2: Improvement by k-NN on residuals

	Test1 RMSE	
Base Model	Base	Base + k-NN
MF	23.3918	23.1006
RBM	25.7833	24.0637

[New Correlation Functions]
We propose two new correlation functions. The first is called Temporal Common User Support,

$$c^{\text{TCUS}}(i,j) = \frac{n_{ij}}{n_i n_j},$$

where each user gets a heuristic vote of weight $\frac{1}{\log(|\mathcal{D}(u)|+3)}$ for each of her rating. The term n_i or n_j stores the sum of votes from users who have rated item i or item j, respectively. The term n_{ij} stores the sum of votes from users who have rated both items i and j, with each vote inversely scaled by $1 + \gamma|t_{ui} - t_{uj}|$. The scaling indicates that the "joint vote" across a longer time span will be de-emphasized. We take a fixed choice of $\gamma = 0.1$ base on its promising validation RMSE.

The second function is called Normalized Common Count.

$$c^{\text{NCC}}(i,j) = \sum_{u \in \mathcal{D}(i) \cap \mathcal{D}(j)} \frac{1}{|(\mathcal{D}(j)| \times |\mathcal{D}(u)|}.$$

The correlation function is similar to Common User Support, which only cares about whether an item is rated rather than the exact rating received. Then, the test instances can also be used in computing this correlation function.

[Emphasize Same-time Neighbors]
We find that ratings that come from the same user at exactly the same time are very similar. In addition, 37.4% of the testing instances have same-time neighbors in the training and validation sets. We exploit the finding by increasing the weights of same-time item neighbors, as follows.

$$w_{i,j} = \begin{cases} 10000 \cdot c(i,j) & \text{if } i,j \text{ are rated at the same time;} \\ c(i,j) & \text{otherwise.} \end{cases}$$

[Emphasize Artist Neighbors]
We find that the artist information in the taxonomy of the data set can provide some help in k-NN. In particular, we modify the Time Decay k-NN by

$$w_{ij} = \frac{(1 + \text{isArtist}(i,j)) \cdot c(i,j)}{1 + \alpha|t_{ui} - t_{uj}|},$$

where $\alpha = 10$ and isArtist(i,j) equals a tunable positive parameter if i is the corresponding artist of item j (or vice versa), and 0 otherwise. We try using other taxonomy information but only the artist one shows some improvements.

Table 3: Transformed k-NN with normalized common count correlation on MF residuals

Transform	Test1 RMSE
ϕ_1	23.3436
ϕ_2	22.9413
ϕ_3	22.9809

[Polynomial Transform of Weights]
We consider a transformed k-NN formulation with

$$\hat{r}_{ui} = \sum_{j \in G_k(u,i)} \phi\left(\frac{w_{ij}}{\sum_{j \in G_k(u,i)} w_{ij}}\right) \cdot r_{uj}.$$

The original k-NN simply takes the identity transform. We explore polynomial transforms $\phi_d(a) = a^d$ for $d = 1, 2, 3$. Higher order transforms would push small weights to zero very quickly. Some representative results are shown in Table 3 and demonstrate that the transforms can be helpful. In addition, we observe that the models resulting from different transforms are diverse and can boost the final ensemble.

2.4. Probabilistic Latent Semantic Analysis

Probabilistic Latent Semantic Analysis (pLSA) is a generative model for collaborative filtering (Hofmann, 2004). The key idea is to introduce a hidden variable z, which indicates the latent class within $\{1, 2, \cdots, k\}$, for every user-item pair. The number of possible classes k is a pre-specified parameter. Given a user-item pair (u, i), pLSA models the probability of r_{ui} by

$$P(r|u,i) = \sum_{\kappa=1}^{k} P(r|i, z=\kappa) P(z=\kappa|u). \tag{3}$$

The training of pLSA corresponds to obtaining all the probability terms in the summation. Then, during prediction, the expected rank over $P(r|u,i)$ is taken as $\hat{r}_{ui}$.

Since z is a discrete random variable, each probability term in (3) can be represented by an item-wise table and a user-wise table. We initialize the tables by random values within $[\frac{1}{2}, \frac{1}{2} + 10^{-6}]$. Then, the traditional Expectation Maximization (EM) algorithm (Dempster et al., 1977) can be used to find the maximum likelihood estimation of the terms. In addition to the traditional EM, we also use the tempered EM (Hofmann, 2004) to prevent overfitting. The results shown in Table 4 demonstrate that tempered EM improves the Test1 RMSE by 0.468. Because (3) treats user and item in an asymmetric manner, we also consider an Inversed pLSA model by exchanging the roles of u and i.

Next, we discuss two of our specific ideas that make pLSA a promising individual model for the competition.

[Compress Rating Space]
Note that r_{ui} takes possibly 101 different values in this competition. Directly treating r_{ui}

Table 4: Results of pLSA

Methods ($k = 80$)	Test1 RMSE
Traditional EM	25.3936
Tempered EM	24.9256
Tempered EM + Emphasizing	24.8282

as 101 possible discrete values leads to a large probability table and can cause overfitting. Thus, we compress r_{ui} to binary values $b_{ui} \in \{0, 1\}$, embed each original rating r_{ui} as a soft evidence of $b_{ui} = 1$ with evidence level $r_{ui}/100$, and conduct EM to obtain $P(b|i, z)$ and $P(z|u)$. During prediction, the expected value over $P(b|u, i)$ is scaled back to $[0, 100]$ as $\hat{r}_{ui}$. At the beginning of the competition, we observe that the compression trick is indeed helpful in alleviating overfitting. All the results in Table 4 already include this trick.

[Emphasize Validation Set]
We have discussed during the SGD training of MF models in Subsection 2.1 that it is important to focus more on the examples that are closer to the test instances in time. In pLSA, we do so by emphasizing the validation set because of its time-wise closeness to the test set. In particular, each example in the validation set receives a weight of 1.5 during the M-step of EM. The third row in Table 4 shows that the emphasizing technique improves the Test1 RMSE by 0.0974.

2.5. Probabilistic Principle Component Analysis

Probabilistic Principle Component Analysis (pPCA) can be viewed as a probabilistic form of matrix factorization (Lawrence and Urtasun, 2009; Tipping and Bishop, 1999). Recall that in MF models, the ratings that an item receives from all of the users, denoted as vector $\mathbf{r}_i$, are modeled as $\mathbf{P}\mathbf{q}_i$, where $\mathbf{P}$ is a matrix that contains all the user-factor vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ is the item-factor vector. The pPCA model assumes that values in $\mathbf{q}_i$ are sampled from a k-dimensional standard Gaussian distribution, where the number of factors k is a prescribed parameter. In addition, pPCA assumes that the rating vector $\mathbf{r}_i$ is sampled from

$$\mathbf{r}_i|\mathbf{q}_i \sim \mathcal{N}(\mathbf{P} \cdot \mathbf{q_i} + \bar{\mathbf{r}}, \sigma^{\mathbf{2}}\mathbf{I}),$$

based on the user-factor matrix $\mathbf{P}$. The vector $\bar{\mathbf{r}}$ contains the average rating $\bar{r}_u$ received from user u in the u-th dimension. During training, pPCA learns $\mathbf{P}$, $\mathbf{q}_i$, and the most probable $\mathbf{r}_i$ with the EM algorithm (Dempster et al., 1977). Under Gaussian assumptions, we can analytically compute the expectation of r_{ui} over $P(r_{ui})$, which is $\mathbf{p}_u^T\mathbf{q}_i + \bar{r}_u$ and is taken as $\hat{r}_{ui}$. In addition to the pPCA model discussed above, similar to what we have done for pLSA, we also include an Inversed pPCA model in our final solution.

[Sparse pPCA Formulation]
The original pPCA formulation treats the unrated items by a given user as missing values that need to be estimated. That is, every user-item pair (u, i) holds a place in the rating vector $\mathbf{r}_i$ during the E-step of the EM algorithm. Such a dense formulation is not feasible for this competition. We thus propose a sparse formulation that only considers the rated pairs

(u, i) during the E-step and skips the unrated pairs to make pPCA computationally feasible. When the matrix $\mathbf{P}$ and all vectors $\mathbf{q}_i$ are randomly initialized among $[0, 1]$, $\sigma^2 = 10$ and $k = 20$, the sparse pPCA formulation reaches Test1 RMSE 24.4613.

2.6. Supervised Regression

Regression aims at learning a function that maps input features to a real-valued output (which is the rating in the special case of the track-1 task) and is a well-established field in supervised learning. Nevertheless, regression has been used only in a limited manner in previous collaborative filtering work (Töscher and Jahrer, 2009). The key difficulty of applying regression is that examples in collaborative filtering contain only abstract features (u, i) rather than meaningful ones. Following our experience in KDDCup 2010 (Yu et al., 2011), we consider feature engineering techniques to extract meaningful features and feed them to mature regression algorithms.

We include three regression algorithms in our solution, where the first two follow from the work of Töscher and Jahrer (2009). The first one is linear/ridge regression, which learns a linear regressor and is relatively fast in our C implementation. The second one is a Neural Network in WEKA (Hall et al., 2009), which can learn a nonlinear regressor. The third one is initialized Gradient Boosting Regression Trees (iGBRT) implemented in RTRank (Mohan et al., 2011), which learns an ensemble of decision trees.

Because of the huge size of the training set, the regression models are only trained with the validation set. Neural Network and iGBRT implementations are highly memory-consuming. Thus, we adopt the random subspace method as commonly used in Random Forest (Breiman, 2001). In each round of the method, F features are randomly chosen for regression. We include models from many different rounds in the final ensemble. We take $F \in \{3, 5, 20\}$ for Neural Network and $F \in \{3, 4\}$ for iGBRT. Next, we describe our efforts in extracting meaningful features for regression.

[Feature Engineering]

We extract two kinds of meaningful features: statistical and model-based features. Statistical features are constructed by carefully analyzing the data set. Some features are related to the date, hour and taxonomy information and others are related to the rating behavior of users or the rating history of items. A list of statistical features is in Appendix B.

Model-based features are extracted by considering the representative variables in the models introduced. For instance, the vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ extracted from MF can be readily used as features for (u, i). The original MF takes a specific quadratic formula of those features to make the prediction, and regression can possibly make other uses of the features. The hidden variables $\tilde{\mathbf{h}}_u$ in RBM, $P(u|z = \kappa)$ and $P(i|u, z = \kappa)$ in pLSA and the total neighbor weights $\sum_{j \in G_k(u,i)} w_{ij}$ in k-NN are all included as our model-based features. We observe that the k-NN features usually work the best.

3. Validation-set Blending

The individual models described in Section 2 can be trained with either the training set or the combined (training + validation) set. In the beginning, we make a few attempts to blend the models linearly using the validation set. Then, we switch our efforts to conduct *nonlinear*

Table 5: Validation blending with neural network

Architecture	Number of Features	Test1 RMSE
1 layer	53	21.7141
2 layer	53	21.6789
3 layer narrow[a]	53	21.4564

[a] has only 5 neurons in a layer.

blending using the validation set. The goal of the nonlinear blending is to introduce diverse and better-performing models for the next stage of test-set blending (to be described in Section 4). For diversity, we explore some different algorithms with various parameter settings; for better performance, we not only use the model predictions, but also include some promising features that are generated during the feature engineering step in supervised regression (see Subsection 2.6).

More specifically, validation-set blending works like supervised regression except that many of the features are simply the predictions from individual models. During training, the features come from the models that are trained on only the training set; during prediction, the features come from the corresponding models that are trained on the combined (training + validation) set (Töscher and Jahrer, 2009). We use AdaBoost.RT (Solomatine and Shrestha, 2004), a boosting based regression algorithm, and the blending method introduced in (Bell et al., 2008), in a few of our validation blending attempts. The major efforts on validation blending are spent on two other algorithms: Neural Networks and Binned Linear Regression.

3.1. Neural Network Blending

We apply Neural Networks to validation blending after observing its promising performance for supervised regression in Subsection 2.6. We choose the hyperbolic tangent as the transfer function and set the number of neurons in a layer equal to the number of features. Promising performance is also observed for validation blending, as shown in Table 5.

3.2. Binned Linear Regression

Binned Linear Regression (BLR) (Töscher and Jahrer, 2009) tries to respect the strengths and weaknesses of various models in different parts of the feature space, and blend the models differently within each part. In particular, during the blending phase, the validation set is partitioned into disjoint bins according to some particular criteria. Then, a linear/ridge regression model is applied to each bin of the validation set to learn a local model. During prediction, an instance (u, i) is assigned to one bin according to its features, and then $\hat{r}_{ui}$ is estimated with the associated local model of the bin.

Existing BLR algorithms only consider partitioning criteria based on one single feature. For instance, we can do partitioning by a single categorical feature "taxonomy type" that indicates whether the item refers to a track, album, artist and genre. In our solution, we consider 17 partitioning criteria, each of which is based on one single feature, as described in Appendix C. In addition, we develop two novel techniques to improve the performance of BLR. The first is to use multiple-feature bins instead of single-feature ones; the second is to conduct ridge regression for multiple stages instead of one single stage.

[Multiple-feature BLR]
Consider a multi-feature binning scheme as follows. For each feature, we partition the validation set to 1, 2, 4, 8, $\cdots$ single-feature bins. We then define a multiple-feature bin as the intersection of single-feature bins. That is, if feature 1 results in two bins $B_1^{(1)}, B_2^{(1)}$, feature 2 is not used in partitioning (only one $B_1^{(2)}$), and feature 3 results in four bins $B_1^{(3)}, \cdots, B_4^{(3)}$, then all the possible multiple-feature bins are

$$\left\{B_1^{(1)} \cap B_1^{(2)} \cap B_1^{(3)}, B_1^{(1)} \cap B_1^{(2)} \cap B_2^{(3)}, \cdots, B_2^{(1)} \cap B_1^{(2)} \cap B_4^{(3)}\right\}.$$

The challenge is to locate suitable single-feature bins such that the resulting multiple-feature bins perform well in BLR. Since the number of combinations on single-feature bins is large, a brute-force search over the optimal combination may not be feasible. We thus adopt a greedy search strategy to locate suitable single-feature bins. Also, we observe that BLR overfits when a bin contains too few examples. Thus we set a lower bound $(4,000)$ on the number of examples in each bin and an upper bound (64) on the number of bins.

The greedy search strategy works as follows. Initially, we assume that all the features are not used for partitioning (only one single-feature bin for each feature). Then, the strategy examines each feature and attempts to partition with the feature further (double its number of bins). The feature that results in the best validation performance retains its partitions. When doubling the number of bins from each feature, we try to keep the number of examples in each bin roughly balanced. That is, for a 4-bin partitioning on a single feature, each single-feature bin will contain roughly a quarter of the validation examples. The attempt continues for at most 6 iterations, creating at most $2^6 = 64$ multiple-feature bins as the outcome of the greedy search strategy. The specific greedy search strategy is shown in Appendix D.

[Multi-stage Regression]
The blending steps above are fast enough to be executed many times in the eight-hour interval between two submissions. Thus, we decide to locally merge the blended models first before checking the Test1 performance. In particular, the various blended models (from single-bin and multiple-bin BLR) are merged by linear regression on the validation set to form a joint model. We call such a merging step two-stage blending, which improves the Test1 RMSE by about 0.2 from the best single-bin BLR model. We follow the same line of thought and reuse the validation set for another merging step on two-stage blended models, which results in a three-stage blended model that improves the Test1 RMSE slightly.

Some representative validation-set blending results are listed in Table 6.

4. Test-set Blending

We apply the test-set blending technique to combine the individual models introduced in Section 2 and the validation-set-blended models from Section 3. The test-set blending technique uses Test1 RMSE to estimate the linear blending weights and is widely used in

Table 6: Results of Binned Linear Regression

Method	Test1 RMSE
17 models, single-feature bin	22.0829
26 models, multi-feature bin	21.8128
31 models, two-stage	21.6455
44 models, two-stage	21.4591
48 models, three-stage	21.4238

past competitions when the evaluation criterion is RMSE (Töscher and Jahrer, 2009; Yu et al., 2011).

Let N be the number of examples in the test set, $\mathbf{r} \in R^N$ be the true ratings in the test set and $\mathbf{z}_m \in R^N, m = 1, \cdots, M$ be the predicted ratings from the m-th model. A linear blending of the predicted ratings by ridge regression optimizes the target function

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{r} - \mathbf{Z}\mathbf{w}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2.$$

The optimal solution of $\mathbf{w}$ is $\mathbf{w} = (\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^T\mathbf{r}$. Here λ is the regularization parameter and $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_M]$. The obtained weight vector $\mathbf{w}$ is then used to compute the ensemble prediction $\hat{\mathbf{r}} = \mathbf{Z}\mathbf{w}$, and $\hat{\mathbf{r}}$ is then clipped and quantized to the desired range.

It is easy to calculate $\mathbf{Z}^T\mathbf{Z}$ directly, but the vector of true ratings $\mathbf{r}$ is unknown. Test-set blending (Töscher and Jahrer, 2009) uses Test1 information to estimate $\mathbf{Z}^T\mathbf{r}$. In particular, let the RMSE of the m-th model on the whole test set be $e_m = \sqrt{\frac{\|\mathbf{r}-\mathbf{z}_m\|^2}{N}}$. Then,

$$\mathbf{z}_m^T\mathbf{r} = \frac{\|\mathbf{r}\|^2 + \|\mathbf{z}_m\|^2 - Ne_m^2}{2}. \tag{4}$$

Although e_m and $\|\mathbf{r}\|^2$ are both unavailable, they can be estimated as follows.

$$e_m \approx \tilde{e}_m \text{ and } \|\mathbf{r}\|^2 \approx N\tilde{e}_0^2, \tag{5}$$

where $\tilde{e}_m$ is the Test1 RMSE of the m-th model and $\tilde{e}_0$ is the Test1 RMSE of the all-zero prediction.

[Linear Blending RMSE Estimator]

The competition allows only one submission every eight hours. The limited number of submissions makes it difficult to use the Test1 RMSE on leaderboard to do parameter tuning on λ for test-set blending or evaluate the contribution of each model in the ensemble, both of which need extensive submissions to the leaderboard. Following the reasons behind test-set blending, we develop a technique to estimate the test RMSE of the blended model *before* actually submitting it. Recall that the prediction of the blended model is $\hat{\mathbf{r}} = \mathbf{Z}\mathbf{w}$, and thus its test RMSE is

$$e = \sqrt{\frac{\|\mathbf{r} - \hat{\mathbf{r}}\|^2}{N}} = \sqrt{\frac{\|\mathbf{r}\|^2 - 2\mathbf{r}^T\hat{\mathbf{r}} + \|\hat{\mathbf{r}}\|^2}{N}}.$$

Table 7: Estimated and actual Test1 RMSE for test-set blending

Models	Estimated RMSE	Test1 RMSE
All models	21.0253	21.0254
− validation blending[a]	21.6375	21.6330
− vb − regression	21.6636	21.6618
− vb − PPCA	21.6389	21.6341
− vb − PLSA	21.6418	21.6355
− vb − KNN	21.6673	21.6623
− vb − RBM	21.6689	21.6639
− vb − MF	22.0020	21.9865

[a] "−" denotes excluding the category.

The term $\|\mathbf{r}\|^2$ has readily been estimated by (5) and the term $\|\hat{\mathbf{r}}\|^2$ can be easily computed. The only other term is

$$\mathbf{r}^T\hat{\mathbf{r}} = \mathbf{r}^T\mathbf{Z}\mathbf{w},$$

and a simple reuse of (4) for $\mathbf{z}_m^T\mathbf{r}$ estimates this term. Thus, for each test-set-blended model, we can estimate its Test1 RMSE without actually submitting it to the leaderboard. We thus conduct offline choices on what models to include in the final ensemble while saving the precious submissions for individual and validation-blended models. In the final test-set ensemble, we take $\lambda = 10^{-6}$ after examining some choices using the offline estimator. We also conduct one particular offline experiment, leave-category-out, to evaluate the contribution of each category of models, as discussed below.

We separate the models in the final ensemble into seven categories: validation-blended and the six categories of individual models. In each experiment, we leave some categories out of the test ensemble and use the offline RMSE estimator to evaluate the performance. The results are shown in Table 7. We see that validation blending provides a significant boost (estimated 0.6) of performance. For the strength of individual models, MF is the strongest, followed by k-NN, RBM and regression, while pPCA and pLSA provide marginal improvements. Furthermore, the estimated RMSE is consistently quite close to the Test1 RMSE and is thus useful for making choices offline.

5. Post-processing Steps

After obtaining the result from test-set blending, we further apply the following two post-processing steps whose purposes are to adjust our predictions according to effects observed in the data collection system.

[Four-star Rating System]
We observe that the rating system of Yahoo! Music has gone through many different versions during the 11-year span of the track 1 data set. In particular, the changes in the user interface design may significantly influence the distribution of ratings. During the

Table 8: Results of the best individual and blended models

Model	Test1 RMSE
Best individual model (MF-45)	22.9022
Best residual based model (KNN-17)	22.7915
Best validation-set blended model (VB-94)	21.3598
Test set blended ensemble	21.0254
Final result with post-processing	21.0147

See Appendix A for details of all 221 model

period of $4281 \leq t_{ui} \leq 6170$, a four-star rating system is employed by default and the four-star rating is transformed to $r_{ui} = 90$ rather than 100. Thus, during the period, there are very few ratings that are greater than 90. On the basis of this discovery, we perform the following adjustment on the predictions after test-set blending:

$$\hat{r}_{ui} = \begin{cases} 90, & \text{if } \hat{r}_{ui} > 90 \text{ and } 4281 \leq t_{ui} \leq 6170, \\ \hat{r}_{ui}, & \text{otherwise.} \end{cases} \tag{6}$$

In an earlier experiment on one particular model, using (6) can lower the Test1 RMSE from 22.3802 to 22.3437.

[Automatic Zero Rating]
We discover that in many cases, if an album is rated as zero by a user, all the tracks within the album are also rated zero for the user. This result may come from an automatic rating mechanism in the data collection system. We thus apply the following post-processing rule:

$$\hat{r}_{ui} = \begin{cases} 0, & \text{if } i \text{ is a track and its album}_{ij} \text{ is rated } 0, \\ \hat{r}_{ui}, & \text{otherwise.} \end{cases}$$

Some earlier results suggest that this rule can improve the Test1 RMSE marginally.

6. Conclusions

We introduce four key components of our solution to track 1 of KDDCup 2011: well-developed individual models with some new ideas to deliver promising performance, validation-set blending for power and diversity, test-set ensemble with off-line RMSE estimation as the capstone and post-processing techniques as the final touch.

At the beginning of the competition, our systematic studies on individual models build a solid basis for our solutions. Then, in the last few weeks of the competition, our efforts on validation-set blending lead to a significant boost in performance over individual models, and eventually make it possible to build a large ensemble of diverse models based on six categories of basic individual models. Table 8 presents the performance improvement observed from the best individual model, the best residual-based model, the best validation-blended model and our final test-set-blended ensemble.

We believe that the success of our solution lies in our never-ending pursuit of digesting and combining different types of information from the given data set, such as hidden user/item factors for ratings, taxonomy, rating time, statistical features, the local strengths or weaknesses of different models and the observations on the data collection system. In other words, the techniques of feature engineering and blending—two precious building blocks of our winning solution in KDDCup 2010—are again important for our solution in track 1 of KDDCup 2011.

Acknowledgments

We thank the organizers for holding this interesting and meaningful competition. We thank the reviewers for valuable comments. We also thank National Taiwan University for providing a stimulating research environment, and thank Jane Hsu and En-Hsu Yen for fruitful discussions. This work was supported by National Science Council, National Taiwan University and Intel Corporation under grants NSC99-2911-I-002-201, 99R70600, and 10R70500.

References

Robert M. Bell, Yehuda Koren, and Chris Volinsky. The BellKor 2008 solution to the Netflix prize. Technical report, 2008.

Leo Breiman. Random forests. *MLJ*, 2001.

Arthur Pentland Dempster, Nan M. Laird, and Donald Bruce Rubin. Maximum likelihood from incomplete data via the EM algorithm. *JRSS-B*, 1977.

Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The Yahoo! Music Dataset and KDD-Cup'11. *KDD-Cup Workshop*, 2011.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 2009.

Geoffrey Hinton. A practical guide to training restricted Boltzmann machines. Technical report, 2010.

Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 2004.

Yehuda Koren and Robert M. Bell. Advances in collaborative filtering. *Recommender Systems Handbook*, 2011.

Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with Gaussian processes. In *ICML*, 2009.

Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

Ling Li and Hsuan-Tien Lin. Ordinal regression by extended binary classification. In *NIPS*, 2006.

Hsuan-Tien Lin and Ling Li. Support vector machinery for infinite ensemble learning. *JMLR*, 2008.

Gilles Louppe. Collaborative filtering: Scalable approaches using restricted boltzmann machines. Master's thesis, Université de Liège, 2010.

Chih-Chao Ma. Large-scale collaborative filtering algorithms. Master's thesis, National Taiwan University, 2008.

Ananth Mohan, Zheng Chen, and Kilian Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. *JMLR W&CP*, 2011.

Martin Piotte and Martin Chabbert. The Pragmatic Theory solution to the Netflix Grand prize. Technical report, 2009.

Yu-Xun Ruan, Hsuan-Tien Lin, and Ming-Feng Tsai. Improving ranking performance with cost-sensitive ordinal classification via regression. Technical report, National Taiwan University, 2011.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann machines for collaborative filtering. In *ICML*, 2007.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. The MIT Press, 2001.

Dimitri P. Solomatine and Durga L. Shrestha. AdaBoost.RT: A boosting algorithm for regression problems. In *IJCNN*, 2004.

Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *JRSS-B*, 1999.

Andreas Töscher and Michael Jahrer. The BigChaos solution to the Netflix prize 2008. Technical report, 2008.

Andreas Töscher and Michael Jahrer. The BigChaos solution to the Netflix grand prize. Technical report, 2009.

Markus Weimer, Alexandros Karatzoglou, and Alexander J. Smola. Improving maximum margin matrix factorization. In *ECML/PKDD*, 2008.

Hsiang-Fu Yu, Hung-Yi Lo, Hsun-Ping Hsieh, Jing-Kai Lou, Todd G. McKenzie, Jung-Wei Chou, Po-Han Chung, Chia-Hua Ho, Chun-Fu Chang, Yin-Hsuan Wei, Jui-Yu Weng, En-Syu Yan, Che-Wei Chang, Tsung-Ting Kuo, Yi-Chen Lo, Po Tzu Chang, Chieh Po, Chien-Yuan Wang, Yi-Hung Huang, Chen-Wei Hung, Yu-Xun Ruan, Yu-Shi Lin, Shou-De Lin, Hsuan-Tien Lin, and Chih-Jen Lin. Feature engineering and classifier ensemble for KDD Cup 2010. In *JMLR W&CP*, 2011. To appear.

Appendix A. Predictor List

In this section we list all 221 predictors which are included in our final test-set blending. All listed RMSEs represent the Test1 RMSEs on the leaderboard. In the predictor list, the combined set means the union of the training and validation sets.

A.1. Matrix Factorization Predictors

η denotes the learning rate in SGD.

- MF-1: Test1 RMSE=24.5603
  Biased MF, trained on training set, $k = 150$.
- MF-2: Test1 RMSE=25.1210
  Binned Biased MF, trained on training set, $k = 65, \eta = 0.0002, \lambda_P = 0.01, \lambda_Q = 0.01$.
- MF-3: Test1 RMSE=26.9394
  Biased MF with summed Gaussian kernel, trained on training set, $k = 20$.
- MF-4: Test1 RMSE=24.4112
  Biased MF, trained on training set, $k = 150$.
- MF-5: Test1 RMSE=24.5826
  CSVD, trained on training set.
- MF-6: Test1 RMSE=24.7073
  Biased MF, trained on training set, regularization term is affected by number of ratings, $k = 60$.
- MF-7: Test1 RMSE=24.4863
  CSVD, trained on combined set where validation set is before training set, $k = 1000$.
- MF-8: Test1 RMSE=24.5052
  Biased MF with power-2 kernel, trained on training set, $k = 20$.
- MF-9: Test1 RMSE=25.2340
  NMF, trained on training set, $k = 150$.
- MF-10: Test1 RMSE=23.4803
  Biased MF, trained on combined set where validation set is before training set, $k = 1000, \lambda_P = 0.1, \lambda_Q = 0.1$.
- MF-11: Test1 RMSE=23.2459
  Biased MF, $k = 120, \eta = 0.0004, \lambda_P = 2.4, \lambda_Q = 0.5$.
- MF-12: Test1 RMSE=23.9886
  Biased MF with stump kernel, trained on combined set, $k = 20$.
- MF-13: Test1 RMSE=23.5622
  CSVD, trained on combined set where validation set is before training set, $k = 112$.

- MF-14: Test1 RMSE=23.7053
  CSVD, trained on combined set where validation set is before training set, $k = 168$.
- MF-15: Test1 RMSE=23.6196
  CSVD, trained on combined set where validation set is before training set, $k = 500$.
- MF-16: Test1 RMSE=23.5509
  CSVD, trained on combined set where validation set is before training set, $k = 800$.
- MF-17: Test1 RMSE=23.7911
  CSVD, trained on combined set where validation set is before training set, $k = 110$.
- MF-18: Test1 RMSE=23.6570
  CSVD, trained on combined set where validation set is before training set, $k = 900$.
- MF-19: Test1 RMSE=23.7021
  CSVD, trained on combined set where validation set is before training set, $k = 168$.
- MF-20: Test1 RMSE=24.1967
  Biased MF, trained on training set, $k = 150$.
- MF-21: Test1 RMSE=23.1266
  Biased MF, trained on combined set, $k = 150$.
- MF-22: Test1 RMSE=23.0931
  Biased MF with day biased terms, trained on combined set, $k = 150$.
- MF-23: Test1 RMSE=23.0783
  Biased MF with day/hour biased terms, trained on combined set, $k = 150$.
- MF-24: Test1 RMSE=25.0517
  MF in GraphLab, $k = 40, \lambda = 1.0$.
- MF-25: Test1 RMSE=23.0023
  Biased MF, ratings truncated to $[0, 100]$ during SGD, trained on combined set, $k = 1000, \lambda_{\mu_u^{\text{user}}} = 1.5, \lambda_{\mu_i^{\text{item}}} = 0.17, \lambda_P = 20, \lambda_Q = 0.21$.
- MF-26: Test1 RMSE=23.0018
  Biased MF, ratings truncated to $[0, 100]$ during SGD, trained on combined set, $k = 1000, \lambda_{\mu_u^{\text{user}}} = 1.5, \lambda_{\mu_i^{\text{item}}} = 0.17, \lambda_P = 20, \lambda_Q = 0.21$.
- MF-27: Test1 RMSE=23.0100
  Biased MF, ratings truncated to $[0, 100]$ during SGD, trained on combined set, $k = 1000, \lambda_{\mu_u^{\text{user}}} = 1.5, \lambda_{\mu_i^{\text{item}}} = 0.17, \lambda_P = 20, \lambda_Q = 0.21$.
- MF-28: Test1 RMSE=23.0202
  Biased MF, ratings truncated to $[0, 100]$ during SGD, trained on combined set. $k = 500$.
- MF-29: Test1 RMSE=23.4200
  Biased MF, trained on combined set, $k = 600$.

- MF-30: Test1 RMSE=23.7568
  Biased MF, trained on combined set where validation set is before training set, $k = 150, \lambda_{\mu_u^{\text{user}}} = 1.5, \lambda_{\mu_i^{\text{item}}} = 0.17, \lambda_P = 20, \lambda_Q = 0.21$.

- MF-31: Test1 RMSE=23.0579
  Biased MF, trained on combined set where validation set is before training set, ratings truncated to $[0, 100]$ during SGD, $k = 150, \lambda_{\mu_u^{\text{user}}} = 1.5, \lambda_{\mu_i^{\text{item}}} = 0.17, \lambda_P = 20, \lambda_Q = 0.21$.

- MF-32: Test1 RMSE=24.2748
  Biased MF, ratings truncated to $[0, 100]$ during SGD, trained on training set, $k = 150, \lambda_{\mu_u^{\text{user}}} = 1.5, \lambda_{\mu_i^{\text{item}}} = 0.17, \lambda_P = 20, \lambda_Q = 0.21$.

- MF-33: Test1 RMSE=23.0726
  Biased MF, ratings truncated to $[0, 100]$ during SGD, trained on combined set, $k = 150, \lambda_{\mu_u^{\text{user}}} = 1.5, \lambda_{\mu_i^{\text{item}}} = 0.17, \lambda_P = 20, \lambda_Q = 0.21$.

- MF-34: Test1 RMSE=23.4729
  Biased MF, train on combined set, training examples are sorted by dates, $k = 150$.

- MF-35: Test1 RMSE=23.0297
  Biased MF, predict ratings right after finishing training the corresponding user, trained on combined set, $k = 150$.

- MF-36: Test1 RMSE=23.7863
  Biased MF with time deviation and day/hour biased terms, reduce number of training examples in each round, trained on combined set, $k = 150$.

- MF-37: Test1 RMSE=23.9114
  MF in GraphLab, $k = 200, \lambda = 1.0$.

- MF-38: Test1 RMSE=22.9241
  Binned Biased MF with time deviation and day/hour biased terms, trained on combined set, $k = 250$.

- MF-39: Test1 RMSE=23.7761
  Binned Biased MF with time deviation and day/hour biased terms, reduce number of training example in each rounds, trained on combined set, $k = 150$.

- MF-40: Test1 RMSE=22.9694
  Binned Biased MF with time deviation and day/hour biased terms, trained on combined set, $k = 150$.

- MF-41: Test1 RMSE=23.0587
  2-stage Biased MF with time deviation and day/hour biased terms, trained on combined set, $k = 150$.

- MF-42: Test1 RMSE=22.9129
  2-stage Binned Biased MF, trained on combined set, $k = 150$.

- MF-43: Test1 RMSE=23.1756
  2-stage Binned Biased MF, trained on training set in the first stage, trained on combined set, $k = 150$.

- MF-44: Test1 RMSE=22.9156
  Binned Biased MF with time deviation and day/hour biased terms, trained on combined set, $k = 300$.

- MF-45: Test1 RMSE=22.9022
  Binned Biased MF with time deviation and day/hour biased terms, trained on combined set, $k = 500$.

- MF-46: Test1 RMSE=25.7221
  MF, trained on training set, $k = 1000, \#\text{iter} = 300$.

- MF-47: Test1 RMSE=22.9644
  Binned Biased MF, trained on combined set, $k = 1000, \lambda_{\mu_u^{\text{user}}} = 1.5, \lambda_{\mu_i^{\text{item}}} = 0.17, \lambda_P = 20, \lambda_Q = 0.21$.

- MF-48: Test1 RMSE=23.4654
  Binned Biased MF, trained on training set, not converged yet, $k = 2000, \lambda_{\mu_u^{\text{user}}} = 1.5, \lambda_{\mu_i^{\text{item}}} = 0.17, \lambda_P = 20, \lambda_Q = 0.21$.

- MF-49: Test1 RMSE=25.3412
  Biased MF, trained on training set, overfitting, $k = 200$.

- MF-50: Test1 RMSE=24.8993
  ORMF, trained on training set, $k = 20$.

- MF-51: Test1 RMSE=23.6376
  Biased MF, trained on combined set, $k = 150$.

- MF-52: Test1 RMSE=23.0836
  2-stage Biased MF with time deviation and day/hour biased terms, trained on combined set, $k = 150$.

- MF-53: Test1 RMSE=23.5742
  Biased MF with time deviation and day/hour biased terms, trained on combined set, overfitting, $k = 200$.

- MF-54: Test1 RMSE=23.6737
  Biased MF with time deviation and day/hour biased terms, trained on combined set, overfitting, $k = 200$.

- MF-55: Test1 RMSE=24.1032
  Biased MF with time deviation and day/hour biased terms, trained on training set, $k = 1000$.

- MF-56: Test1 RMSE=25.6337
  Biased MF, trained on training set, $k = 150$.

- MF-57: Test1 RMSE=22.9533
  Binned Biased MF with day/hour biased terms, trained on combined set, $k = 1000$.
- MF-58: Test1 RMSE=24.5578
  Biased MF with power-2 kernel, trained on training set, $k = 100$.
- MF-59: Test1 RMSE=22.9063
  2-stage Binned Biased MF with day/hour biased terms, trained on combined set, $k = 150$.
- MF-60: Test1 RMSE=23.5633
  ORMF, trained on combined set, $k = 20$.
- MF-61: Test1 RMSE=26.4979
  NMF, trained on training set, $k = 150, \lambda_P = 1, \lambda_Q = 1$.
- MF-62: Test1 RMSE=26.3865
  NMF, trained on training set, $k = 150, \lambda_P = 1, \lambda_Q = 1$.
- MF-63: Test1 RMSE=24.0701
  Binned Biased MF with day/hour biased terms, trained on training set, $k = 1000$.
- MF-64: Test1 RMSE=23.0046
  As MF-65, a overfitting version.
- MF-65: Test1 RMSE=24.1793
  Binned Biased MF, trained on training set, $k = 150$.
- MF-66: Test1 RMSE=22.9959
  As MF-65, but the parameters were tuned for albums only.
- MF-67: Test1 RMSE=22.9965
  As MF-65, but the parameters were tuned for tracks only.
- MF-68: Test1 RMSE=22.9938
  As MF-65, with different learning rate.
- MF-69: Test1 RMSE=23.6544
  As MF-65, but leave out one third data in training set.
- MF-70: Test1 RMSE=23.0180
  Binned Biased MF, regularization term is affected by number of ratings, trained on combined set, $k = 150$.
- MF-71: Test1 RMSE=24.6699
  Biased MF, trained on training set, $k = 150$.
- MF-72: Test1 RMSE=24.8675
  Biased MF, trained on training set, regularization term is affected by number of ratings, $k = 20$.

- MF-73: Test1 RMSE=24.5550
  Biased MF, trained on training set, $k = 50$.
- MF-74: Test1 RMSE=25.2278
  Biased MF with summed Gaussian kernel, trained on combined set, $k = 50$.
- MF-75: Test1 RMSE=25.7034
  NMF, trained on combined set, $k = 150, \lambda_P = 1.0, \lambda_Q = 1.0$.
- MF-76: Test1 RMSE=23.7287
  Biased MF, trained on combined set, $k = 112$.
- MF-77: Test1 RMSE=23.2360
  Binned Biased MF, trained on combined set, $\lambda_P = 2.0, \lambda_Q = 0.5, k = 200$.
- MF-78: Test1 RMSE=23.2562
  Biased MF, trained on combined set, $\lambda_P = 2.5, \lambda_Q = 0.5, k = 200$.
- MF-79: Test1 RMSE=25.5372
  NMF, trained on combined set, $k = 150, \#\text{iter} = 140, \lambda_P = 1.0, \lambda_Q = 1.0$.
- MF-80: Test1 RMSE=25.2665
  NMF, trained on combined set, $k = 150, \#\text{iter} = 160, \lambda_P = 1.0, \lambda_Q = 1.0$.
- MF-81: Test1 RMSE=23.3918
  Biased MF with power-2 kernel, trained on combined set, $k = 100$.

A.2. Restricted Boltzmann Machine Predictors

- RBM-1: Test1 RMSE=22.9820
  gRBM on MF residual (RMSE=22.9974), trained on combined set, 30 hidden units.
- RBM-2: Test1 RMSE=25.7833
  cRBM, trained on training set, 120 hidden units.
- RBM-3: Test1 RMSE=24.7748
  RBM, 80 hidden units.
- RBM-4: Test1 RMSE=25.9986
  fRBM, trained on training set, 100 hidden units.
- RBM-5: Test1 RMSE=22.8008
  gRBM on MF residual (RMSE=22.9974), trained on combined set, 100 hidden units.
- RBM-6: Test1 RMSE=24.8597
  RBM, trained on combined set, 100 hidden units.
- RBM-7: Test1 RMSE=24.7433
  RBM, trained on combined set, 160 hidden units.
- RBM-8: Test1 RMSE=26.0846
  RBM, 50 hidden units, overfitting version, trained on combined set, iteration=200.

A.3. k-Nearest Neighbors Predictors

- KNN-1: Test1 RMSE=27.6396
  k-NN, common support correlation, trained on training set, $k = 20$.

- KNN-2: Test1 RMSE=27.3624
  k-NN, set correlation, trained on training set, $k = 20$.

- KNN-3: Test1 RMSE=24.2742
  k-NN with artist effect on MF residual, Pearson correlation, trained on combined set, $k = 20$.

- KNN-4: Test1 RMSE=23.1237
  k-NN on MF residual, Pearson correlation with Fisher transform, trained on combined set, $k = 20$.

- KNN-5: Test1 RMSE=24.9820
  k-NN on RBM residual, Pearson correlation with Fisher transform, trained on training set, $k = 20$.

- KNN-6: Test1 RMSE=23.0645
  k-NN on MF residual, common user support correlation, trained on combined set, $k = 20$.

- KNN-7: Test1 RMSE=23.3436
  k-NN on MF residual (RMSE=22.9974), normalized common count correlation, trained on combined set, $k = 20$.

- KNN-8: Test1 RMSE=23.8853
  k-NN on MF residual (RMSE=24.5666), normalized common count correlation, trained on combined set, $k = 20$.

- KNN-9: Test1 RMSE=23.0683
  k-NN with time decay on MF residual (RMSE=22.9974), normalized common count correlation, trained on combined set, $k = 20$.

- KNN-10: Test1 RMSE=22.9413
  k-NN with polynomial transform $\phi_2(x)$ on MF residual (RMSE=22.9974), normalized common count correlation, trained on combined set, $k = 20$.

- KNN-11: Test1 RMSE=22.9809
  k-NN with polynomial transform $\phi_3(x)$ on MF residual (RMSE=22.9974), normalized common count correlation, trained on combined set, $k = 20$.

- KNN-12: Test1 RMSE=42.9415
  k-NN with polynomial transform $\phi_1(x)$ on MF residual (RMSE=22.9974), normalized common count correlation, trained on combined set, $k = 20$.

- KNN-13: Test1 RMSE=22.8068
  k-NN on MF residual, temporal common user support correlation, trained on combined set, $k = 20$.

- KNN-14: Test1 RMSE=22.9041
  k-NN on MF residual, Pearson correlation with Fisher transform, trained on combined set, $k = 20$.

- KNN-15: Test1 RMSE=22.8954
  k-NN on MF residual, residual correlation, trained on combined set, $k = 20$.

- KNN-16: Test1 RMSE=26.8065
  k-NN, MF feature correlation, trained on training set, $k = 20$.

- KNN-17: Test1 RMSE=22.7915
  k-NN on MF residual, Pearson correlation with Fisher transform, trained on combined set, $k = 20$.

- KNN-18: Test1 RMSE=23.2083
  k-NN on MF residual (RMSE=22.9533), normalized common count correlation, trained on combined set, $k = 20$.

A.4. Probabilistic Latent Semantic Analysis Predictors

- PLSA-1: Test1 RMSE=26.0911
  Average of a pLSA model ($k = 40$) and an inversed pLSA model ($k = 100$), trained on training set.

- PLSA-2: Test1 RMSE=25.9100
  Inversed pLSA, trained on combined set, $k = 200$.

- PLSA-3: Test1 RMSE=26.0343
  Inversed pLSA, trained on combined set, $k = 200$.

- PLSA-4: Test1 RMSE=25.50
  PLSA, trained on combined set, $k = 150$.

- PLSA-5: Test1 RMSE=25.3936
  PLSA, trained on combined set, $k = 80$.

- PLSA-6: Test1 RMSE=24.8282
  PLSA, use tempered EM, emphasize the validation set, trained on combined set, $k = 80$.

- PLSA-7: Test1 RMSE=24.9256
  PLSA, using tempered EM, trained on combined set, $k = 80$.

A.5. Probabilistic Principle Component Analysis Predictors

- PPCA-1: Test1 RMSE=24.4613
  pPCA, trained on combined set, the matrix $\mathbf{P}$ and all vectors $\mathbf{q}_i$ are randomly initialized between $[0, 1]$ and $\sigma^2 = 10$, $k = 20$.

- PPCA-2: Test1 RMSE=24.7506
  Inversed pPCA, trained on combined set, the matrix $\mathbf{P}$ and all vectors $\mathbf{q}_i$ are randomly initialized between $[0, 1]$ and $\sigma^2 = 10$, $k = 20$.

A.6. Supervised Regression Predictors

All supervised regression predictors are trained on validation set.

- REG-1: Test1 RMSE=24.7379
  Linear regression solved by SGD on features.
- REG-2: Test1 RMSE=24.2551
  Ridge regression solved by SGD on features.
- REG-3: Test1 RMSE=24.2590
  Linear regression on features.
- REG-4: Test1 RMSE=24.1251
  Linear regression on features.
- REG-5: Test1 RMSE=24.4040
  Linear regression on features.
- REG-6: Test1 RMSE=30.8504
  1-layer Neural Network on 3 randomly selected features.
- REG-7: Test1 RMSE=29.4195
  1-layer Neural Network on 3 randomly selected features.
- REG-8: Test1 RMSE=30.6383
  GBDT on 3 randomly selected features.
- REG-9: Test1 RMSE=35.1370
  GBDT on 3 randomly selected features.
- REG-10: Test1 RMSE=32.2341

A.7. Validation-set Blending Predictors

All validation blending predictors are trained on validation set.

- VB-1: Test1 RMSE=21.9985
  1-layer Neural Network validation blending with features.
- VB-2: Test1 RMSE=22.6238
  Linear regression on MF models.
- VB-3: Test1 RMSE=24.1491
  1-layer Neural Network on randomly selected predictions and features.
- VB-4: Test1 RMSE=22.9630
  1-layer Neural Network on randomly selected predictions and features.

- VB-5: Test1 RMSE=22.9390
  Ridge regression validation blending with features.
- VB-6: Test1 RMSE=27.5290
  1-layer Neural Network on randomly selected predictions and features.
- VB-7: Test1 RMSE=27.4327
  1-layer Neural Network on randomly selected predictions and features.
- VB-8: Test1 RMSE=22.7551
  Linear regression on many kernel MF models.
- VB-9: Test1 RMSE=22.8033
  BLR on many KNN models using different correlations with different item types.
- VB-10: Test1 RMSE=22.5137
  2-layer Neural Network on 4 predictions.
- VB-11: Test1 RMSE=24.0045
  1-layer Neural Network on randomly selected predictions and features.
- VB-12: Test1 RMSE=24.0378
  1-layer Neural Network on randomly selected predictions and features.
- VB-13: Test1 RMSE=27.8017
  1-layer Neural Network on randomly selected predictions and features.
- VB-14: Test1 RMSE=27.9672
  1-layer Neural Network on randomly selected predictions and features.
- VB-15: Test1 RMSE=28.2184
  1-layer Neural Network on randomly selected predictions and features.
- VB-16: Test1 RMSE=27.9061
  1-layer Neural Network on randomly selected predictions and features.
- VB-17: Test1 RMSE=28.0946
  1-layer Neural Network on randomly selected predictions and features.
- VB-18: Test1 RMSE=30.0536
  2-layer Neural Network on randomly selected predictions and features.
- VB-19: Test1 RMSE=24.0347
  2-layer Neural Network on randomly selected predictions and features.
- VB-20: Test1 RMSE=22.9961
  2-layer Neural Network on randomly selected predictions and features.
- VB-21: Test1 RMSE=23.8811
  2-layer Neural Network on randomly selected predictions and features.

- VB-22: Test1 RMSE=23.0419
  2-layer Neural Network on randomly selected predictions and features.
- VB-23: Test1 RMSE=23.0218
  Ridge regression with large regularization on predictions and features.
- VB-24: Test1 RMSE=22.9841
  Linear regression on predictions and features.
- VB-25: Test1 RMSE=22.8755
  Ridge regression with $\lambda = 2 \times 10^8$ on predictions and features.
- VB-26: Test1 RMSE=22.8666
  Ridge regression with $\lambda = 3 \times 10^8$ on predictions and features.
- VB-27: Test1 RMSE=23.3446
  Linear regression on predictions and features.
- VB-28: Test1 RMSE=22.9730
  Linear regression on predictions and features.
- VB-29: Test1 RMSE=25.6320
  Linear regression on many k-NN predictions.
- VB-30: Test1 RMSE=22.5583
  Linear regression on 4 selected predictions.
- VB-31: Test1 RMSE=22.6201
  Linear regression on 4 selected predictions.
- VB-32: Test1 RMSE=28.4018
  2-layer Neural Network on 20 randomly selected predictions and features.
- VB-33: Test1 RMSE=27.4580
  2-layer Neural Network on 20 randomly selected predictions and features.
- VB-34: Test1 RMSE=28.1559
  2-layer Neural Network on 20 randomly selected predictions and features.
- VB-35: Test1 RMSE=23.0079
  2-layer Neural Network on 20 randomly selected predictions and features.
- VB-36: Test1 RMSE=27.6423
  2-layer Neural Network on 20 randomly selected predictions and features.
- VB-37: Test1 RMSE=22.7901
  Ridge regression with $\lambda = 3 \times 10^8$, on selected predictions and features.
- VB-38: Test1 RMSE=22.9374
  Linear regression on selected predictions and features.

- VB-39: Test1 RMSE=23.3011
  3-layer Neural Network on 4 randomly selected predictions and features.

- VB-40: Test1 RMSE=31.1280
  3-layer Neural Network on 4 randomly selected predictions and features.

- VB-41: Test1 RMSE=23.6005
  3-layer Neural Network on 4 randomly selected predictions and features.

- VB-42: Test1 RMSE=22.7518
  3-layer Neural Network on 4 randomly selected predictions and features.

- VB-43: Test1 RMSE=24.1434
  3-layer Neural Network on 4 randomly selected predictions and features.

- VB-44: Test1 RMSE=31.7023
  3-layer Neural Network on 4 randomly selected predictions and features.

- VB-45: Test1 RMSE=30.1118
  3-layer Neural Network on 4 randomly selected predictions and features.

- VB-46: Test1 RMSE=22.7338
  Ridge regression with $\lambda = 1 \times 10^9$, on selected predictions and features.

- VB-47: Test1 RMSE=22.7446
  Adaboost.RT on 5 models.

- VB-48: Test1 RMSE=22.6371
  Kernel ridge regression.

- VB-49: Test1 RMSE=22.5343
  BLR on 4 models + features, use [taxonomy information] to partition bin.

- VB-50: Test1 RMSE=22.5761
  BLR on 4 models, use [date of rating] to partition bin.

- VB-51: Test1 RMSE=22.5323
  BLR on 4 models, use [date of rating] to partition bin.

- VB-52: Test1 RMSE=22.3478
  BLR on 4 models + features, use [number of ratings of items] to partition bin.

- VB-53: Test1 RMSE=22.4396
  BLR on 4 models + features, use [support] to partition bin.

- VB-54: Test1 RMSE=22.3663
  BLR on 4 models + features, use [taxonomy information] to partition bin.

- VB-55: Test1 RMSE=22.4392
  BLR on 4 models + features, use [number of ratings of users] to partition bin.

- VB-56: Test1 RMSE=22.3839
  BLR on 4 models + features, use [date of rating] to partition bin.

- VB-57: Test1 RMSE=22.5259
  BLR on 4 models, use [number of ratings of items] to partition bin.

- VB-58: Test1 RMSE=22.6019
  BLR on 4 models, use [user's standard deviation](each bin has same number of instances) to partition bin.

- VB-59: Test1 RMSE=22.5993
  BLR on 4 models, use [user's standard deviation] to partition bin.

- VB-60: Test1 RMSE=22.5737
  BLR on 4 models, use [user's standard deviation](when testing, calculate std with validation set) to partition bin.

- VB-61: Test1 RMSE=22.5411
  BLR on 4 models, use [number of ratings of users] to partition bin.

- VB-62: Test1 RMSE=22.3236
  BLR on 7 models + features, use [number of ratings of items] to partition bin.

- VB-63: Test1 RMSE=22.4176
  BLR on 7 models + features, use [support] to partition bin.

- VB-64: Test1 RMSE=22.3474
  BLR on 7 models + features, use [taxonomy information] to partition bin.

- VB-65: Test1 RMSE=22.4170
  BLR on 7 models + features, use [number of ratings of users] to partition bin.

- VB-66: Test1 RMSE=22.4945
  BLR on 7 models, use [number of ratings of items] to partition bin.

- VB-67: Test1 RMSE=22.5474
  BLR on 7 models, use [user's standard deviation](when testing, calculate std with validation) to partition bin.

- VB-68: Test1 RMSE=22.5168
  BLR on 7 models, use [support] to partition bin.

- VB-69: Test1 RMSE=22.5209
  BLR on 7 models, use [number of ratings of users] to partition bin.

- VB-70: Test1 RMSE=22.6658
  Linear regression on selected predictions.

- VB-71: Test1 RMSE=22.1000
  BLR on 20 models + features, use [number of ratings of users] to partition bin.

- VB-72: Test1 RMSE=22.0829
  BLR on 17 models + features, use [number of ratings of items] to partition bin.

- VB-73: Test1 RMSE=22.1274
  BLR on 21 models + features, use [date of rating] to partition bin.

- VB-74: Test1 RMSE=23.6476
  Bellkor 2008 blending.

- VB-75: Test1 RMSE=21.8553
  BLR on 20 models + features, use 5 bin partitioning criteria and bagging.

- VB-76: Test1 RMSE=21.8456
  BLR on 20 models + features, use [number of ratings of users] + [taxonomy] + [same time as the last rating] to partition bin.

- VB-77: Test1 RMSE=21.8128
  BLR, use [sum of k-NN similarities] + [same time as the last rating] to partition bin.

- VB-78: Test1 RMSE=22.3503
  2-stage BLR.

- VB-79: Test1 RMSE=21.6918
  2-stage BLR, 10 bins.

- VB-80: Test1 RMSE=21.7107
  2-stage BLR, 8 bins.

- VB-81: Test1 RMSE=21.8338
  2-stage BLR, 8 bins.

- VB-82: Test1 RMSE=21.8605
  BLR on 31 models + features, use [clustering] + [same time as the last rating] + [user mean day] + [item mean day] to partition bin.

- VB-83: Test1 RMSE=21.6455
  2-stage BLR on 33 models + features.

- VB-84: Test1 RMSE=21.6647
  2-stage BLR on 41 models + features, 7 bins.

- VB-85: Test1 RMSE=23.1815
  Bellkor 2008 blending.

- VB-86: Test1 RMSE=21.5317
  BLR on 39 models + features, use greedy search strategy to partition bin.

- VB-87: Test1 RMSE=21.7141
  Bellkor 2008 blending.

- VB-88: Test1 RMSE=21.5061
  2-stage BLR on 44 models + features, use greedy search strategy to partition bin, 10 bins.

- VB-89: Test1 RMSE=21.4591
  2-stage BLR on 44 models + features, use greedy search strategy to partition bin, 15 bins.

- VB-90: Test1 RMSE=21.4287
  3-stage BLR on 44 models + features, use greedy search strategy to partition bin, 6 bins.

- VB-91: Test1 RMSE=21.6789
  2-layer Neural Network on 53 models + features.

- VB-92: Test1 RMSE=21.6331
  3-stage BLR on 48 models + features, use greedy search strategy to partition bin, 13 bins.

- VB-93: Test1 RMSE=21.4564
  3-layer narrow Neural Network on 53 models + features.

- VB-94: Test1 RMSE=21.3598
  1-layer Neural Network on 63 models + features.

- VB-95: Test1 RMSE=21.4238
  3-stage BLR.

Appendix B. Statistical Features

- **User Features**
  - Number of ratings
  - Rating mean
  - Rating variance
  - Rating standard deviation
  - Number of zero ratings
  - Ratio of zero ratings to all ratings
  - Date of first appearance
  - Date of last disappearance
  - Rating Frequency
  - Average rating hour
  - Rating hour variance
- **Item Features**
  - Number of ratings
  - Rating mean
  - Rating variance
  - Rating standard deviation
  - Number of zero ratings
  - Ratio of zero ratings to all ratings
  - Date of first appearance
  - Date of last disappearance
  - Frequency of being rated
  - Is a track or not
  - Is an album or not
  - Is an artist or not
  - Number of genre
  - Have a missing album ID or not
  - Have a missing artist ID or not
- **Rating Features**
  - Date
  - Shifted hour $((h_{ui} + 18) \mod 24)$
  - Day or night
  - Weekday or weekend
  - Was done by auto-rating-zero mechanism or not

Appendix C. Bin Partitioning Criteria

- **User's standard deviation**
  The standard deviation of each user's rating in the training set.

- **Taxonomy information (Track, Album, Artist, Genre)**
  Separate Item into 4 bins according to their types.

- **Same time as the last rating**
  Whether the time of the rating is the same as the last rating from the same user in the training set.

- **Support**
  The value of $\min(|\mathcal{D}(u)|, |\mathcal{D}(i)|)$.

- **#User rating**
  The number of user's ratings in the training set.

- **#Item rating**
  The number of item's ratings in the training set.

- **Cross model standard deviation**
  The standard deviation of predicted rating by 30 models.

- **User mean day**
  The mean of the date of ratings done by the user. $\bar{t}_u = \frac{\sum_{i,(u,i)\in\mathcal{D}} t_{ui}}{|\mathcal{D}(u)|}$

- **Item mean day**
  The mean of the date of ratings on the item. $\bar{t}_i = \frac{\sum_{u,(u,i)\in\mathcal{D}} t_{ui}}{|\mathcal{D}(i)|}$

- **Sum of similarities**
  The sum of similarities of the top-six most similar items.

- **Time difference between the rating and the last rating in the training set**
  The difference in days between the rating and the last rating made by the user in the training set.

- **User frequency**
  The number of daily user ratings.

- **User deviation**
  The difference in days between the rating and the user mean day (as defined above) of the user.
  $(t_{ui} - \bar{t}_u)^{\alpha}$ with α set to 0.4.

- **Item frequency**
  The number of daily item ratings.

- **Item deviation**
  The difference in days between the rating and the item mean day (as defined above) of the item.
  $(t_{ui} - \bar{t}_i)^\alpha$ with α set to 0.4.

- **Clustering**
  Co-clustering aiming on maximize the performance of the user mean and item mean model $\hat{r}_{ui} = \bar{r} + \mu_u^{\text{user}} + \mu_i^{\text{item}}$.

- **Date of rating**
  The date of the rating t_{ui}.

Appendix D. Pseudo-code

Algorithm 1: SGD for Basic Matrix Factorization

```
initialize matrices $\mathbf{P}$ and $\mathbf{Q}$
for $t = 1$ to NumofIter do
    for each ratings $r_{ui}$ in the training set do
        $\mathbf{p}_u := \mathbf{p}_u + \eta \cdot (\mathbf{q}_i(r_{ui} - \hat{r}_{ui}) - \lambda \mathbf{p}_u)$
        $\mathbf{q}_i := \mathbf{q}_i + \eta \cdot (\mathbf{p}_u(r_{ui} - \hat{r}_{ui}) - \lambda \mathbf{q}_i)$
    end
    compute the RMSE improvement on validation data
    if improvement less then $\epsilon$ then
        break
    end
end
```

Algorithm 2: Two-stage SGD for Matrix Factorization

```
initialize matrices $\mathbf{P}$ and $\mathbf{Q}$
do Algorithm 1
for $k = 1 \ldots 3$ do
    for last four ratings of each user in training set do
        $\mathbf{p}_u := \mathbf{p}_u + \eta \cdot (\mathbf{q}_i(r_{ui} - \hat{r}_{ui}) - \lambda \mathbf{p}_u)$
        $\mathbf{q}_i := \mathbf{q}_i + \eta \cdot (\mathbf{p}_u(r_{ui} - \hat{r}_{ui}) - \lambda \mathbf{q}_i)$
    end
end
```

Algorithm 3: Greedy Bin Search

```
binPartitionSet $:= \phi$
for $i = 1 \ldots 6$ do
    find the best $j$ which minimizes binPartitionRMSE(binPartitionSet $\cup \{j\}$)
    binPartitionSet $:=$ binPartitionSet $\cup \{j\}$
end
return binPartitionSet
```

Appendix E. Correlation Functions

We use U to denote the total number of users and $\mathcal{D}(i,j) = \mathcal{D}(i) \cap \mathcal{D}(j)$, where $\mathcal{D}(i)$ and $\mathcal{D}(j)$ are as defined in Section 2.

- **Common Support Correlation**

$$c^{\text{CS}}(i,j) = \frac{|\mathcal{D}(i,j)| \cdot U}{|\mathcal{D}(i)||\mathcal{D}(j)|}.$$

- **Pearson Correlation and Fisher Transform**

$$c^{\text{Pearson}}(i,j) = \frac{\sum_{u \in \mathcal{D}(i,j)} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in \mathcal{D}(i,j)} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{u \in \mathcal{D}(i,j)} (r_{uj} - \bar{r}_j)^2}},$$

$\bar{r}_i$ is the average rating of item i.

The Fisher Transform correlation is based on Pearson correlation and is defined by

$$c^{\text{Fisher}}(i,j) = 0.5 * \ln\left(\frac{1 + c^{\text{Pearson}}(i,j)}{1 - c^{\text{Pearson}}(i,j)}\right).$$

- **Mean Squared Error Correlation**

$$c^{\text{MSE}}(i,j) = \frac{1}{\frac{1}{|\mathcal{D}(i,j)|} \sum_{u \in \mathcal{D}(i,j)} (r_{ui} - r_{uj})^2}.$$

- **Common User Support Correlation**

$$c^{\text{CUS}}(i,j) = \frac{n_{i,j} \cdot U}{n_i n_j} \text{ , where } n_{i,j} = \sum_{u \in \mathcal{D}(i,j)} \frac{1}{|\mathcal{D}(u)|} \text{ , } n_i = \sum_{u \in \mathcal{D}(i)} \frac{1}{|\mathcal{D}(u)|}.$$

- **Set Correlation**

$$c^{\text{Set}}(i,j) = \frac{|\mathcal{D}(i,j)|}{\min(|\mathcal{D}(i)|, |\mathcal{D}(j)|)}.$$

- **Residual Correlation**

$$c^{\text{Residual}}(i,j) = \frac{\sum_{u \in \mathcal{D}(i,j)} (r_{ui} - \hat{r}^{\text{base}}_{ui})(r_{uj} - \hat{r}^{\text{base}}_{uj})}{\sqrt{\sum_{u \in \mathcal{D}(i,j)} (r_{ui} - \hat{r}^{\text{base}}_{ui})^2} \sqrt{\sum_{u \in \mathcal{D}(i,j)} (r_{uj} - \hat{r}^{\text{base}}_{uj})^2}},$$

where $\hat{r}^{\text{base}}_{ui}$ is the prediction of another base model.

- **Temporal Common User Support Correlation**

$$c^{\text{TCUS}}(i,j) = \frac{n_{i,j}}{n_i n_j} \ ,$$

where $n_{i,j} = \sum_{u \in \mathcal{D}(i,j)} (\frac{1}{\log(|\mathcal{D}(u)|+3)})(\frac{1}{1+\lambda|t_{ui}-t_{uj}|}), n_i = \sum_{u \in \mathcal{D}(i)} (\frac{1}{\log(|\mathcal{D}(u)|+3)})$.

We set λ to a constant 0.1.

- **Normalized Common Count**

$$c^{\text{NCC}}(i,j) = \sum_{u \in \mathcal{D}(i) \cap \mathcal{D}(j)} \frac{1}{|(\mathcal{D}(j)| \times |\mathcal{D}(u)|}.$$

Table 9 demonstrates the performance of k-NN based on different correlation functions above.

Table 9: k-NN on MF residual

Correlation	Date Effect (α)	Potential Score (s)	Rate Shrink (β)	Validation RMSE	Test1 RMSE
Pure MF	N/A	N/A	N/A	20.8611	22.9974
CS	10	15	45	20.7764	22.9727
Fisher	10	15	0.3	20.7309	22.8823
MSE	10	15	550	20.6936	22.8816
CUS	10	10	15	20.6308	22.8917
Set	10	15	0.3	20.7054	22.9421
Residual	10	15	0.08	20.6466	22.8954
TCUS	10	7	10^{-6}	20.5620	22.8068

The pure MF model in this table and Table 2 are different in number of features and initial settings.