Journal of Machine Learning Research 18:61–74, 2012 Proceedings of KDD-Cup 2011 competition

Collaborative Filtering Ensemble

Michael Jahrer MICHAEL.JAHRER@COMMENDO.AT
Andreas Töscher ANDREAS.TOESCHER@COMMENDO.AT
8580 Köflach, Austria

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

Abstract

This paper provides the solution of the team "commendo" on the Track1 dataset of the KDD Cup 2011 Dror et al.. Yahoo Labs provides a snapshot of their music-rating database as dataset for the competition. We get approximately 260 million ratings from 1 million users on 600k items. Timestamp and taxonomy information are added to the ratings. The goal of the competition was to predict unknown ratings on a testset with RMSE as error measure. Our final submission is a blend of different collaborative filtering algorithms. The algorithms are trained consecutively and they are blended together with a neural network.
Keywords: Collaborative Filtering, Ensemble Learning, KDD Cup, Rating Prediction, Music Ratings

1. Introduction

Yahoo! Music released one of the largest rating datasets for public analysis Dror et al.. The goal is to submit the lowest RMSE score on a test set.

1.1. Notation

We consider the dataset as sparse matrix $\mathbf{R} = [r_{ui}]$, where we use the letter u for users and i for items during this writeup. Bold letters are used for matrices and vectors. $\mathbb{I}$ is the set of items and $|\mathbb{I}|$ is the total number of items, $|\mathbb{I}|$=624,961. $\mathbb{U}$ is the set of users and $|\mathbb{U}|$ is the total number of users, $|\mathbb{U}|$=1,000,990. Predictions for user/item pairs are denoted as $\hat{r}_{ui}$. The date of the rating is given by t_{ui}. Resolution of t_{ui} is given in minutes, hence we can calculate *minute*, *hour* and *day* values per rating. The set of items, which is rated by user u is $\mathbb{I}(u)$. The set of users, which rates item i is $\mathbb{U}(i)$.

For all models we want to minimize the quadratic error $E(\theta)$ on the set of ratings r_{ui}. The letter θ indents the trainable weights/parameter of the model. To avoid overfitting we use L2-penalty on all parameters during learning. $||\mathbf{A}||_F$ denotes the Frobenius norm and it is defined as $||\mathbf{A}||_F = \sqrt{\sum_{\forall i,j} a_{ij}^2}$. This leads to the following error function:

$$E(\theta) = \frac{1}{|\mathbf{R}|} \sum_{u,i \in \mathbf{R}} (r_{ui} - \hat{r}_{ui}(\theta))^2 + \lambda ||\theta||_F^2 \tag{1}$$

The following notation is used for model parameters. $\mathbf{p}_u$ is a user-dependent feature and $\mathbf{q}_i$ is an item-dependent feature. All features are stored in matrices, for example all the user features $\mathbf{p}_u$ are stored in the matrix $\mathbf{p} \in \mathbb{R}^{F \times |U|}$ and the columns are F-dimensional

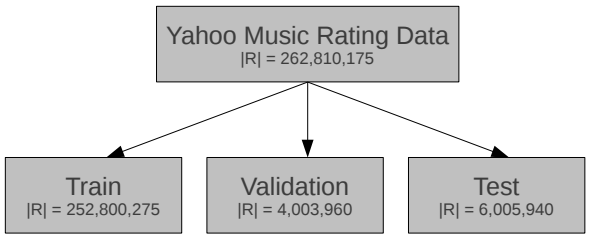

Figure 1: KDD Cup 2011 Track1 dataset

features. The feature values are typically initialized with small random numbers and they are trained with stochastic gradient descent, using the data r_{ui}. As learning rate we use the constant η and λ for L2-regularization. In a basic matrix factorization model a prediction $\hat{r}_{ui}$ is calculated by the dot product of the user and the item feature $\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$.

1.2. The Dataset

The provided dataset for Track1 Dror et al. consists of 262,810,175 ratings from 1,000,990 users on 624,961 items. The range of the rating value is between 0 and 100. An item can be a track, an album, an artist or a genre. Most of the ratings are given to tracks (47%). The distribution of types within all items are given by 507,172 tracks, 88,909 albums, 27,888 artists and 992 genres. The data is considered as sparse user x item matrix $\mathbf{R} = [r_{ui}]$, with the fill-rate of 0.04% being very low. For example in the Netflix Prize dataset Bennett et al. (2007) the fill-rate is about 1%. The timestamps of the ratings are given in minute-resolution.

Additional to the rating data there is taxonomy information available. Taxonomy means the relationship between the items within the music domain. For example track x belongs to album y, and album y belongs to artist z. For a more simple and holistic view we calculate item-to-parent and item-to-child tables. The total number of children is 2,453,808 (same number of parents).

Figure 1 provides an overview of how the dataset is split to train, validation and test.

1.3. The Training of the Models

Our solution consists of a blend of many single predictors. The standard way of training a predictor consists of two steps. In the first step the validation set is removed from the dataset and the model is trained. Then predictions for the validation set are stored. In the second step training is done on all available data with the same meta parameters as in the first step, such as learning rate η, regularization λ, number of epochs and so on. Last, the predictions for the test set are stored. A summary of all step is table 1.

The validation predictions from the first step are used to train the blender. The training of the first predictors is stopped when the RMSE on the validation set is minimal. For all other models a linear combination of all available predictions is directly optimized, this technique is also known as blend-stopping Töscher and Jahrer (2008).

step	description
1	load the complete data
2	remove the validation set from data
3	optimize the model on the validation error
4	store the validation prediction with lowest error
5	insert the validation set into train
6	re-train the model with best parameters
7	predict + store the test set

Table 1: How one predictor is created

predictor	validation RMSE
global mean (μ=48.8)	38.2138
item mean	32.3941
user mean	27.6525

Table 2: Baseline predictions

For all models in section 2 we report RMSE values on the validation set. In table 2 we list three baseline RMSE values. A low RMSE of a single predictor, is not necessarily needed in order to blend well within the ensemble.

2. Algorithms

In the following the used collaborative filtering models are described. Most of the models are trained on the rating data directly, but there is the possibility of training on residuals from another model. How this is done can be read in the Netflix Grand Prize Report from team BigChaos Töscher and Jahrer (2008).

All different models get listed here with their corresponding explanation. The formula for predicting one user/item pair is also added to each of the algorithms. For completeness we add the RMSE value on the validation set of each of the models including the used meta-parameters. The listed numbers are taken from a predictor which is actually in our final blend.

2.1. Baseline Model

The baseline is used for data cleaning. The model does not cover any user/item interaction, similar to the one proposed by Y.Koren in Koren (2010) and the Netflix Prize team Pragmatic Theory in Piotte and Chabbert (2008). The idea of the baseline model is to clean the data and erase effects from all the ratings such as day biases or hour biases. The Baseline prediction $\hat{r}_{ui}$ consists of many parts. The following equation shows all factors.

$$\begin{aligned}\hat{r}_{ui}^{(0)} &= \mu + \mu_u + \mu_i + \mu_{minute} + \mu_{hour} + \mu_{day} + \mu_{u,day} + \\ &\mu_{u,freq(u,day)} + \mu_{i,freq(i,day)} + \\ &\mathbf{p}_u^{(0)^T}\mathbf{y}_{minute}^{(0)} + \mathbf{p}_u^{(1)^T}\mathbf{y}_{hour}^{(0)} + \mathbf{p}_u^{(2)^T}\mathbf{y}_{day}^{(0)} + \\ &\mathbf{q}_i^{(0)^T}\mathbf{y}_{minute}^{(1)} + \mathbf{q}_i^{(1)^T}\mathbf{y}_{hour}^{(1)} + \mathbf{q}_i^{(2)^T}\mathbf{y}_{day}^{(1)}\end{aligned} \tag{2}$$

We divide the prediction formula into three lines. In the first line there are simple biases: globel bias μ, user bias μ_u and item bias μ_i. We have separate biases for each minute μ_{minute}, hour μ_{hour} and day μ_{day} in the dataset. The value for *minute*, *hour* and *day* are calculated from the rating date t_{ui}. The last expression in the first line is a user-day bias $\mu_{u,day}$. In the second line there are frequency-dependent user and item biases. As frequency $freq(u, day)$ we define the number of ratings from user u on the particular *day*. The frequency biases are used for model effects when for example a user gives more ratings on the same day. We limit the value of $freq(u, day)$ and $freq(i, day)$ to 10. In the third line there are factorized user-vs-date biases. For example the first term $\mathbf{p}_u^{(0)^T}\mathbf{y}_{minute}^{(0)}$ is a user x minute dot product. The features $\mathbf{p}^{(0)}$ have the size of $|\mathbb{U}|$ x F. The features $\mathbf{y}^{(0)}$ has the size of $|minutes|$ x F.

In the forth line one can find item-vs-date biases. From the dataset we get a total number of $|minutes| = 5{,}726{,}101$, $|hours| = 95{,}436$ and $|days| = 3{,}978$.

During training all parameters are penalized by the L2-norm. Training of the baseline is done by stochastic gradient descent. We get an error on the validation set of RMSE=23.3766 with F=50, η=0.0002 and λ=0.58 after 40 iteration epochs.

2.2. Matrix Factorization Models

This section is about different matrix factorization techniques. The idea of matrix factorzation is to project users and items to a low-dimensional space. So each user and each item is represented by a latent feature vector. The dot product of them $\mathbf{p}_u^T\mathbf{q}_i$ is the prediction.

2.2.1. SVD

The SVD model is a simple matrix factorzation of the rating matrix. Predictions of a user/item pair are given by the following formula.

$$\hat{r}_{ui}^{(1)} = \mathbf{p}_u^T\mathbf{q}_i \tag{3}$$

This has been one of the most popular models in collaborative filtering since the Netflix Prize Bennett et al. (2007) in 2006. When using gradient descent as learning algorithm, training time grows linear with the number of ratings $|\mathbf{R}|$, prediction time of one user/item

pair can be done in constant time $O(1)$, hence very fast. For completeness we sketch the stochastic gradient descent training in Algorithm 1.

Algorithm 1: Pseudo code for training a SVD on rating data.

Input: Sparse rating matrix $\mathbf{R} \in \mathbb{R}^{|U| \text{x} |I|} = [r_{ui}]$
Tunable: Learning rate η, Regularization λ, feature size F

Initialize user weights $\mathbf{p} \in \mathbb{R}^{F \text{x} |U|}$ and item weights $\mathbf{q} \in \mathbb{R}^{F \text{x} |I|}$ with small random values
while *error on validation set decreases* **do**
  forall the $u, i \in \mathbf{R}$ **do**
    $\hat{r}_{ui} \leftarrow \mathbf{p}_u^T \mathbf{q}_i$
    $e \leftarrow \hat{r}_{ui} - r_{ui}$
    for $k = 1...F$ **do**
      $c \leftarrow p_{uk}$
      $p_{uk} \leftarrow p_{uk} - \eta \cdot (e \cdot q_{ik} + \lambda \cdot p_{uk})$
      $q_{ik} \leftarrow q_{ik} - \eta \cdot (e \cdot c + \lambda \cdot q_{ik})$
    end
  end
end

With F=100, η=0.0002, λ=0.5 and 30 iteration epochs we get RMSE=21.9962 on the validation set. Total training time of the model is about 5 hours.

2.2.2. SVD + Baseline Model

In this model we add the SVD model and the Baseline model together. The SVD part models user/item interactions.

$$\hat{r}_{ui}^{(2)} = \hat{r}_{ui}^{(0)} + \hat{r}_{ui}^{(1)} \tag{4}$$

With F=100, $F_{baseline}$=50, η=0.0002, λ=0.6 and 50 iteration epochs we get RMSE=20.9507 on the validation set.

2.2.3. AFM

This model is called asymmetric factor model because it has only item-dependend parameters. It was first mentioned by Paterek in Paterek (2007). It learns a different kind of data variablity compared to the simple SVD model, hence it blends well within the ensemble. The following equation is the prediction of a user/item pair. In a simplified view, the prediction is again a dot product of the item feature $\mathbf{q}_i$ and the user feature (right side). The user feature is given by the set of items $\mathbb{I}(u)$ that user u has rated. Therefore the user is expressed purely by his rated items.

$$\hat{r}_{ui}^{(3)} = \mu_u + \mu_i + \mathbf{q}_i^T \left(\frac{1}{\sqrt{|\mathbb{I}(u)|}} \sum_{j \in \mathbb{I}(u)} \mathbf{q}_j^{(0)} \right) \tag{5}$$

The two matrices $\mathbf{q}$ and $\mathbf{q}^{(0)}$ have both the size $|\mathbb{I}|$ x F. The term $\frac{1}{\sqrt{|\mathbb{I}(u)|}}$ is used for normalization purpose. User and item biases are μ_u and μ_i. With F=30, η=0.01, λ=0.0008 and 35 iteration epochs we get RMSE=23.5492 on the validation set.

2.2.4. AFM Single Feature

Here we have the same AFM as described above, but one set of item features $\mathbf{q}$.

$$\hat{r}_{ui}^{(4)} = \mu_u + \mu_i + \mathbf{q}_i^T \left(\frac{1}{\sqrt{|\mathbb{I}(u)|}} \sum_{j \in \mathbb{I}(u)} \mathbf{q}_j \right) \tag{6}$$

The item features $\mathbf{q}$ has the size of $|\mathbb{I}|$ x F. With F=30, η=0.0004, λ=0.001 and 50 iteration epochs we get RMSE = 26.5411 on the validation set.

2.2.5. AFM Flipped

Again the same asymmetric idea, but flipped to the item side. We have a fixed user feature, but the item is expressed by the set of users $\mathbb{U}(i)$ which rated this item. Thus, the item feature is build from a normalized sum of rated user features.

$$\hat{r}_{ui}^{(5)} = \mu_u + \mu_i + \mathbf{p}_u^T \left(\frac{1}{\sqrt{|\mathbb{U}(i)|}} \sum_{j \in \mathbb{U}(i)} \mathbf{p}_j^{(0)} \right) \tag{7}$$

The two matrices $\mathbf{p}$ and $\mathbf{p}^{(0)}$ have both the size $|\mathbb{U}|$ x F. With F=30, η=0.00003, λ=0.06 and 50 iteration epochs we get RMSE=22.3234 on the validation set. It is remarkable that the "AFM flipped" model has a better performance than the AFM model itself and comes close to the performance of the SVD model.

2.2.6. ASVD

The ASVD model combines the basic SVD approach with the idea from the AFM. The user is described by the user-dependend feature $\mathbf{p}_u$ and the normalized sum of rated items. This model is the same as SVD++ from Y.Koren Koren (2008).

$$\hat{r}_{ui}^{(6)} = \mu_u + \mu_i + \mathbf{q}_i^T \left(\mathbf{p}_u + \frac{1}{\sqrt{|\mathbb{I}(u)|}} \sum_{j \in \mathbb{I}(u)} \mathbf{q}_j^{(0)} \right) \tag{8}$$

The two matrices $\mathbf{q}$ and $\mathbf{q}^{(0)}$ have both the size $|\mathbb{I}|$ x F, matrix $\mathbf{p}$ has the size $|\mathbb{U}|$ x F. With F=30, η=0.0002, λ=0.005 and 15 iteration epochs we get RMSE=21.9202 on the validation set.

2.2.7. ASVD Flipped

As in the section above, we apply the asymmetric idea from the ASVD model to the item side. Now, the user feature $\mathbf{p}_u$ is fixed and we have on the item side (right brackets) the item-dependend feature $\mathbf{q}_i$ and the normalized sum of rated user features.

$$\hat{r}_{ui}^{(7)} = \mu_u + \mu_i + \mathbf{p}_u^T \left(\mathbf{q}_i + \frac{1}{\sqrt{|\mathbb{U}(i)|}} \sum_{j \in \mathbb{U}(i)} \mathbf{p}_j^{(0)} \right) \tag{9}$$

The two matrices $\mathbf{p}$ and $\mathbf{p}^{(0)}$ have both the size $|\mathbb{U}|$ x F, matrix $\mathbf{q}$ has the size $|\mathbb{I}|$ x F. With F=100, η=0.0002, λ=0 and 3 iteration epochs we get RMSE=22.817 on the validation set.

2.2.8. ASVD + Baseline Model

The next step is to add the factors of the baseline model to the ASVD model, which results in the following formula. This model can explain user/item interactions, as well as remove time effects from the data, which results in the following model:

$$\hat{r}_{ui}^{(8)} = \hat{r}_{ui}^{(6)} + \hat{r}_{ui}^{(0)} \tag{10}$$

With F=150, $F_{baseline}$=60, η=0.0001, λ=0.06 and 30 iteration epochs we get RMSE=21.1875 on the validation set. Total training time of this model is 36h.

2.2.9. ASVD + Baseline Model + time features

In this model we add user- and item-dependend time features to the "ASVD + Baseline" model. Biases are covered by the Baseline model part $\hat{r}_{ui}^{(0)}$. On the item side (second line in equation 11) there is an item-day feature $\mathbf{q}_i^{day}$ added. On the user side we add three separate features based on different time bases. An user-minute feature $\mathbf{p}_u^{minute}$, a user-hour feature $\mathbf{p}_u^{hour}$ and a user-day feature $\mathbf{p}_u^{day}$.

$$\begin{aligned}\hat{r}_{ui}^{(9)} = \hat{r}_{ui}^{(0)} + \\ \left(\mathbf{q}_i + \mathbf{q}_i^{day}\right)^T \cdot \\ \left(\mathbf{p}_u + \frac{1}{\sqrt{|\mathbb{I}(u)|}} \sum_{j \in \mathbb{I}(u)} \mathbf{q}_j^{(0)} + \mathbf{p}_u^{minute} + \mathbf{p}_u^{hour} + \mathbf{p}_u^{day}\right)\end{aligned} \tag{11}$$

The "ASVD + Baseline Model + time features" results in a huge number of parameters, which makes the model very memory intensive. Hence, the number of features F are limited to a small amount. The two matrices $\mathbf{q}$ and $\mathbf{q}^{(0)}$ have both the size $|\mathbb{I}|$ x F, $\mathbf{p}$ has the size $|\mathbb{U}|$ x F. The item-day features $\mathbf{q}^{day}$ has a size of $|\mathbb{I}|$ x F x $|days|$. User-minute feature $\mathbf{p}^{minute}$ has size $|\mathbb{U}|$ x F x $|minutes|$, followed by $\mathbf{p}^{hour}$ with a size of $|\mathbb{U}|$ x F x $|hours|$ and $\mathbf{p}^{day}$ with size $|\mathbb{U}|$ x F x $|days|$.

In this model, the largest feature is the user-minute feature. Assuming $F = 10$ and 4-Byte floats, we would get a size of $1,000,990 \cdot 10 \cdot 5,726,101 \cdot 4Byte = 229$TB, if we would store the full matrix. Instead of doing that, we store only the minute features, where the user gives a rating. An average user gives ratings on 104 different minutes, which reduces the size of the example to $1,000,990 \cdot 10 \cdot 104 \cdot 4Byte = 4$GB.

With F=10, $F_{baseline}$=10, η=0.0004, λ=0.3 and 20 iteration epochs we get RMSE=21.771 on the validation set.

2.2.10. SVD + Baseline Model + time features

This model is quite similar to the one above, the difference is that we have no asymmetric part $\frac{1}{\sqrt{|\mathbb{I}(u)|}} \sum_{j \in \mathbb{I}(u)} \mathbf{q}_j^{(0)}$ and we add an item-hour feature.

$$\begin{aligned} \hat{r}_{ui}^{(10)} &= \hat{r}_{ui}^{(0)} + \\ &\left(\mathbf{q}_i + \mathbf{q}_i^{day} + \mathbf{q}_i^{hour}\right)^T \left(\mathbf{p}_u + \mathbf{p}_u^{minute} + \mathbf{p}_u^{hour} + \mathbf{p}_u^{day}\right) \end{aligned} \tag{12}$$

With F=20, $F_{baseline}$=30, η=0.0004, λ=0.2 and 20 iteration epochs we get RMSE=21.1782 on the validation set. This model is one of the most accurate single models within the ensembe.

2.3. K-Nearest Neighbors Models (KNN)

Neighborhood models can be effectively applied, if the correlation matrix can be precomputed. The size of the correlation matrix is data dependent. Within an item-item KNN we should store all item-to-item correlations. But the KDD Cup 2011 dataset has a huge number of items ($|I|$=624,961). The current hardware is not feasible to store all the entries. Assume we would calculate all the correlations with float accuracy it would require $|\mathbb{I}|$ x $|\mathbb{I}|$ x 0.5 x 4Byte = 781GB of memory, which is simply too much.

Calculate correlations on-the-fly will solve the problem of memory overflow. But this dramatically slows down the prediction time. We tried to use Pearson correlation but this turned out to be too slow to apply it on the KDD Cup 2011 Track1 dataset. In the next two algorithms we present a trick how to make the neighborhood approach runable.

2.3.1. Item-Item KNN with SVD Features

The Item-item neighborhood model uses similarities of items as weights to compute the prediction of a user/item pair. To make an item-item neighborhood model applicable we compute the correlations on the fly by using the inverse of the normalized Euclidean distance between two item features. The correlation c_{ij} between item i and item j with their corresponding item features $\mathbf{q}_i$ and $\mathbf{q}_j$ is given by the following equation, which can be computed in constant time $O(1)$. F is the size of the features.

$$c_{ij} = \left(\frac{\sum_{k=1}^{F} (q_{ik} - q_{jk})^2}{\sqrt{\sum_{k=1}^{F} q_{ik}^2} \sqrt{\sum_{k=1}^{F} q_{jk}^2}} \right)^{-2} \tag{13}$$

Furthermore, the prediction for an item-item neighborhood model is a weighted sum of the ratings from user u multiplied by the item-item correlations and normalized with the absolute sum of the correlations. $\mathbb{I}(u)$ is the set of items that the user rated.

$$\hat{r}_{ui}^{(11)} = \frac{\sum_{j \in \mathbb{I}(u)} r_{uj} c_{ij}}{\sum_{j \in \mathbb{I}(u)} |c_{ij}|} \tag{14}$$

Our item-item neighborhood model uses a sigmoid function to map the correlations c_{ij} to $\hat{c}_{ij}$ by introducing two new parameters σ (scale) and γ (offset).

$$\hat{c}_{ij} = 0.5 \cdot tanh(\sigma \cdot c_{ij} + \gamma) + 0.5 \tag{15}$$

In order to predict the rating $\hat{r}_{ui}$. A k-nearest neighbor algorithm selects K ratings with the highest correlations to the item i. Therefore we indroduce the set of items $\mathbb{I}(u, i, K)$ which consists of rated items from user u with the K-highest correlations to item i. $\mathbb{I}(u, i, K) \subset \mathbb{I}(u)$. Here we have the final prediction formula for the model "Item-Item KNN with SVD Features".

$$\hat{r}_{ui}^{(12)} = \frac{\sum_{j \in \mathbb{I}(u,i,K)} r_{uj} \hat{c}_{ij}}{\sum_{j \in \mathbb{I}(u,i,K)} |\hat{c}_{ij}|} \tag{16}$$

The model requires item features for all items, hence we use the SVD from section 2.2.1 in order to get the item features which are required to compute the "Item-Item KNN with SVD Features". We train the model as follows. First, the algorithm uses the SVD as feature-learner. The number of iteration epochs of the gradient descent procedure is constant. In the second step, the three meta-parameters scale σ, offset γ and neighborhood size K are optimized by using the parameter searcher APT2 from Töscher and Jahrer (2008). APT2 is a simple corrdinate search. The optimization target is the linear combination of all avaliable predictions within the current ensemble. Linear regression is used as linear combiner.

The item-item KNN is more effective when it is applied on the residuals of another model. One of our results is based on the residuals from a "AFM flipped" model. The meta-parameters are σ=3.49, γ=-4.28 and K=184. For the SVD we use F=50, η=0.0003, λ=0.1 and 50 iteration epochs. The resulting error on the validation set is RMSE=20.6802.

2.3.2. User-User KNN with SVD Features

A user-user KNN uses the same idea, but it requires user-user correlations. The correlation c_{uv} between user u and user v with their corresponding user features $\mathbf{p}_u$ and $\mathbf{p}_v$ is given by the following equation, which can be computed in constant time $O(1)$. F is the size of the features.

$$c_{uv} = \left(\frac{\sum_{k=1}^{F} (p_{uk} - p_{vk})^2}{\sqrt{\sum_{k=1}^{F} p_{uk}^2} \sqrt{\sum_{k=1}^{F} p_{vk}^2}} \right)^{-2} \tag{17}$$

In order to predict the rating $\hat{r}_{ui}$. A k-nearest neighbor algorithm selects K ratings with the highest correlations to the item i. Again, we map the correlations c_{ij} to $\hat{c}_{ij}$ by using equation 15.

We indroduce the set of users $\mathbb{U}(i, u, K)$ which consists of user rated item i with the K-highest correlations to user u. $\mathbb{U}(i, u, K) \subset \mathbb{U}(i)$. The prediction of a user/item pair is

$$\hat{r}_{ui}^{(13)} = \frac{\sum_{v \in \mathbb{U}(i,u,K)} r_{vi} \hat{c}_{uv}}{\sum_{v \in \mathbb{U}(i,u,K)} |\hat{c}_{uv}|} \tag{18}$$

During the competition the user-user KNN models does not improve the blend significantly. One possible reason is that the item-item KNN covers much of the neighborhood structure.

The best model from the User-User KNN with SVD Features has an RMSE=22.7863, following meta-parameters are used: σ=24.4, γ=-11.6 and K=1631. For the SVD we use F=50, η=0.0005, and 15 iteration epochs.

2.4. Restricted Boltzmann Machine (RBM)

RBM's for collaborative filtering was first mentioned in 2007 by Salakhutdinov et.al. Salakhutdinov et al. (2007). We use the Conditional Restricted Boltzmann Machine described in the herein before mentioned paper and apply it on the KDD Cup 2011 dataset. In contrast to the paper we do not use mini batches, but apply updates after every training example. Another difference is that we do not factorize the weight matrix.

The set of items rated by user u is denoted as $\mathbb{I}(u)$. Every item i is represented via a set of binary visible units $v_i^{(l)}$, with one visible unit for every discrete rating. The dataset uses ratings ranging from 0 to 100, so there are 101 different ratings. This leads to 101 binary visible units for every item, where only one of these units is activated. The unit $v_i^{(l)}$ is 1, if the user u rated the item i with the rating l, otherwise it is 0. For the conditioning of the hidden states we use the items rated by user u denoted as $\mathbb{I}(u)$.

The probabilities of the visible and hidden units for being high are given by:

$$p(v_i^{(l)} = 1|\mathbf{h}) = \frac{exp(a_i^{(l)} + \sum_j h_j w_{ij}^{(l)})}{\sum_{\tilde{l}} exp(a_i^{(\tilde{l})} + \sum_j h_j w_{ij}^{(\tilde{l})})} \tag{19}$$

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_{i \in \mathbb{I}(u)} \left(d_{ij} + \sum_l w_{ij}^{(l)} v_i^{(l)}\right)\right) \tag{20}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{21}$$

We initialize the biases of the visible units $a_i^{(l)}$ to the logarithm of the probabilities of the corresponding unit of being high. The biases of the hidden units b_j are initialized to zero. The weights w_{ij} and d_{ij} are initialized to values drawn from a zero mean Gaussian distribution with a small standard deviation of 0.001. In the appendix section we list all RBM predictions, for each of the results a value of nRatings is listed. This is the number of discrete rating steps of the model. We found out that the RBMs work better when we use 11 ratings instead of all 101 ratings. The best single RBM uses 100 hidden units to achieve an RMSE of 23.096 on the validation set.

2.5. Basic Taxonomy

The dataset contains taxonomy information. Items consist of tracks, albums, artists and genres. The information which track belongs to which album, the connection from the albums to the artists and from artists to genres was given. The idea of the basic taxonomy models is to produce predictors which blend well by using taxonomy information, which is not captured by the used collaborative filtering models.

In total we use 14 different basic taxonomy predictors:

1. $\tilde{r}_{ui}$ = the number of users which gave a high rating to item i
2. $\tilde{r}_{ui}$ = the number of users which rated item i
3. $\tilde{r}_{ui}$ = the number of high ratings the user u has given to tracks included in the album of the given track i
4. $\tilde{r}_{ui}$ = the number of high ratings the user u has given to tracks belonging to the artist(s) of the given track i
5. $\tilde{r}_{ui}$ = the number of high ratings the user u has given to tracks belonging to the genre(s) of the given track i
6. $\tilde{r}_{ui}$ = the number of ratings the user u has given to tracks included in the album of the given track i
7. $\tilde{r}_{ui}$ = the number of ratings the user u has given to tracks belonging to the artist(s) of the given track i
8. $\tilde{r}_{ui}$ = the number of ratings the user u has given to tracks belonging to the genre(s) of the given track i
9. $\tilde{r}_{ui}$ = the number of high ratings the user u has given to a parent (album, artist, genre) of the given track i
10. $\tilde{r}_{ui}$ = the number of ratings the user u has given to a parent (album, artist, genre) of the given track i
11. $\tilde{r}_{ui}$ = the number of high ratings the user u has given to parents (album, artist, genre) and all children of those parents (distance 2) of track i
12. $\tilde{r}_{ui}$ = the number of ratings the user u has given to parents (album, artist, genre) and all children of those parents (distance 2) of track i
13. $\tilde{r}_{ui}$ = the number of high ratings to item i and parents of item i
14. $\tilde{r}_{ui}$ = the number of ratings to item and parents of item i

The final predictions are given by adding 1 in order to avoid a zero and using the logarithm, which leads to:

$$\hat{r}_{ui} = log(\tilde{r}_{ui} + 1) \tag{22}$$

3. Blending

As blending algorithm we use a neuronal network as proposed in Jahrer et al. (2010). Based on our experience from previous competitions (Netflix Prize, KDD Cup 2010) where the error measure was RMSE too, a neural network as blender is easy to parameterize and delivers great results. Recall the limited time we had for deliver the final prediction, the well-working neural network implementation from Jahrer (2010) was very helpful to keep us focused on creating the ensemble.

All inputs (the predictions) are normalized having zero mean and constant standard deviation. Targets are normalized to [-1...+1]. The net topology is set to two hidden layers, size of 50 and 30 neurons. Stochastic gradient descent is used as training algorithm, the number of epochs is set to 300. The initial learning rate $\eta = 0.003$ is decreased linearly to 0 during training. Therefore η is subtracted by $\frac{0.003}{300} = 0.00001$ every epoch. All units in the net have $tanh()$ as activation function. For validation we use 16-fold cross-validation.

The source for blending is freely available under Jahrer (2010).

4. Conclusion

During the competition the goal was to try many different models, which can explain the variability of the data. Since the dataset is very large (260M ratings) time and space considerations are important. For simple SVD algorithms we use stochastic gradient descent over a randomized list of ratings. For models with an asymmetric part in the prediction formula we use user-wise or item-wise training.

The dataset was strictly split according to the rating date. This means the dates of the ratings have ascending order within these three ordered sets {train,validation,test}. Hence, models with user-time features can not predict future behavior. This is the same issue that Y.Koren discussed in Koren (2010). These features can only clean the data from user-time effects. This has the advantage that user-item interactions can be learned better by the model itself when such effects are removed.

During building the ensemble our goal was to try many different settings. The listed algorithms have a lot of meta parameters, which can be set by hand or tuned automatically. As an addition we train some models on the residuals of anothers. This works best for item-item KNN models. In gradient-descent based algorithms, such as matrix factorizations, we tune the number of training epochs automatically by minimizing the linear combination of the existing predictions.

Unfortunately, there was too little time remaining at the end to test models based on taxonomy integration, similar to those we had used for Track2.

References

James Bennett, Stan Lanning, and Netflix Netflix. The Netflix Prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

Gideon Dror, Yehuda Koren, and Markus Weimer. Yahoo! Music Rating Dataset for the KDD Cup 2011 Track1. http://kddcup.yahoo.com/datasets.php.

Michael Jahrer. ELF - Ensemble Learning Framework. An open source C++ framework for supervised learning. http://elf-project.sourceforge.net, 2010.

Michael Jahrer, Andreas Töscher, and Robert Legenstein. Combining predictions for accurate recommender systems. In *KDD: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010.

Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4): 89–97, 2010. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1721654.1721677.

Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 2007.

Martin Piotte and Martin Chabbert. The Pragmatic Theory solution to the Netflix Grand Prize. Technical report, 2008.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, pages 791–798, 2007.

Andreas Töscher and Michael Jahrer. The BigChaos Solution to the Netflix Prize 2008. Technical report, commendo research & consulting, 2008.

Appendix A. Single predictor performance

no.	model	RMSE Test1	residual no	meta parameters
1	AFM	25.15	-	F=30 epochs=35 η=0.01 λ=0.0008
2	AFM single feature	27.78	-	F=30 epochs=50 η=0.0003 λ=0.001
3	AFM flipped	23.54	15	F=30 epochs=50 η=0.00003 λ=0.15
4	AFM flipped	24.25	-	F=30 epochs=50 η=0.00003 λ=0.06
5	AFM flipped	22.91	37	F=30 epochs=50 η=0.00003 λ=0.06
6	AFM flipped single feature	32.53	-	F=30 epochs=3 η=0.00001 λ=0.0002
7	ASVD	23.85	-	F=30 epochs=16 η=0.0002 λ=0.005
8	ASVD + Baseline	23.11	-	F=150 F_{base}=50 epochs=33 η=0.0001 λ=0.06
9	ASVD + Baseline	23.46	-	F=200 F_{base}=150 epochs=15 η=0.0001 λ=0
10	ASVD + Baseline	23.81	4	F=100 F_{base}=100 epochs=20 η=0.0002 λ=0.1
11	ASVD + Baseline + timef	23.92	-	F=10 F_{base}=10 epochs=20 η=0.0004 λ=0.3
12	ASVD + Baseline + timef	23.82	-	F=20 F_{base}=1 epochs=50 η=0.0003 λ=0
13	ASVD flipped	24.63	-	F=100 epochs=3 η=0.00025 λ=0
14	Basic Taxonomy 1-14	-	-	-
15	Baseline	25.20	-	F=50 epochs=30 η=0.0002 λ=0.6
16	Item-Item KNN SVD feat	22.80	5	F=50 epochs=50 η=0.0002 λ=0.1 σ=3.5 γ=-4.3 K=184
17	Item-Item KNN SVD feat	29.08	-	F=100 epochs=25 η=0.0005 λ=1 σ=25.1 γ=-45.3 K=1
18	Item-Item KNN SVD feat	24.46	3	F=150 epochs=40 η=0.0002 λ=0 σ=8.5 γ=-21 K=21
19	Item-Item KNN SVD feat	27.10	-	F=50 epochs=4 η=0.0003 λ=0.1 σ=0.4 γ=-0.36 K=42
20	Item-Item KNN SVD feat	24.90	43	F=50 epochs=50 η=0.0003 λ=0 σ=0.13 γ=-18.7 K=2
21	Item-Item KNN SVD feat	24.35	30	F=80 epochs=20 η=0.0003 λ=0 σ=1.9 γ=-2.9 K=8
22	Item-Item KNN SVD feat	23.42	38	F=20 epochs=15 η=0.0003 λ=0 σ=820 γ=-3.8 K=9
23	Item-Item KNN SVD feat	25.94	31	F=50 epochs=45 η=0.0005 λ=1 σ=7.1 γ=-1.7 K=3
24	Item-Item KNN SVD feat	23.67	33	F=30 epochs=7 η=0.0005 λ=0 σ=212 γ=-6.6 K=5
25	Item-Item KNN SVD feat	26.38	41	F=60 epochs=15 η=0.0005 λ=0 σ=60200 γ=-4.7 K=5
26	Item-Item KNN SVD feat	24.94	29	F=50 epochs=25 η=0.0003 λ=0.5 σ=26.9 γ=-11.6 K=2
27	User-User KNN SVD feat	24.53	29	F=50 epochs=15η=0.0005 λ=0 σ=24.4 γ=-0.12 K=1631
28	User-User KNN SVD feat	27.14	-	F=10 epochs=3η=0.0005 λ=0 σ=3.9 γ=-0.45 K=425
29	RBM discrete	24.81	-	nRatings=11 nHid=100 epochs=15 η=0.001 λ=0
30	RBM discrete	27.05	-	nRatings=2 nHid=50 epochs=15 η=0.002 λ=0.001
31	RBM discrete	27.47	-	nRatings=2 nHid=300 epochs=15 η=0.001 λ=0
32	RBM gauss	23.83	15	nHid=100 epochs=15 η=0.0003 λ=0.0005
33	SVD	23.92	-	F=100 epochs=30 η=0.00025 λ=0.5
34	SVD	24.85	-	F=100 epochs=30 η=0.00025 λ=0
35	SVD	24.89	-	F=10 epochs=20 η=0.0005 λ=0.1
36	SVD	38.91	-	F=200 epochs=2 λ_u=0.04 λ_i=1.5 trained by ALS
37	SVD + Baseline	22.97	-	F=100 F_{base}=50 epochs=50 η=0.0002 λ=0.5
38	SVD + Baseline + time	23.49	-	F=20 F_{base}=30 epochs=20 η=0.0004 λ=0.2
39	SVD + Baseline + time	23.86	-	F=30 F_{base}=30 epochs=20 η=0.0003 λ=0
40	SVD + Baseline + time	24.68	-	F=3 F_{base}=3 epochs=20 η=0.0003 λ=0
41	SVD + Baseline + time	26.89	-	F=15 F_{base}=15 epochs=30 η=0.0005 λ=10
42	SVD + Baseline + time	23.52	-	F=100 F_{base}=50 epochs=50 η=0.0003 λ=0.4
43	SVD + Baseline + time	24.79	-	F=10 F_{base}=5 epochs=50 η=0.0003 λ=0.4
44	binary type indicator	-	-	-

Table 3: Predictors in the final blend. The Neural-Network blender achieves an RMSE of 18.9092 on the validation set and an RMSE of 21.0815 on the KDD Cup 2011 Track1 leaderbord. Test1 is a 50% subset of the leaderboad set, it contains 3002148 ratings. If a model is trained on the residuals of an other model, it is denoted in the column "residual no". For example the model "AFM flipped" (no=3) is trained on the residuals of the Baseline model (no=15).