Additive Regression Applied to a Large-Scale Collaborative Filtering Problem

Eibe Frank[1] and Mark Hall[2]

[1]Department of Computer Science, University of Waikato, New Zealand
[2]Pentaho Corporation, 5950 Hazeltine National Drive, Suite 340, Orlando, FL, USA
`Email addresses to be disclosed`

Abstract. The much-publicized Netflix competition has put the spotlight on the application domain of collaborative filtering and has sparked interest in machine learning algorithms that can be applied to this sort of problem. The demanding nature of the Netflix data has lead to some interesting and ingenious modifications to standard learning methods in the name of efficiency and speed. There are three basic methods that have been applied in most approaches to the Netflix problem so far: stand-alone neighborhood-based methods, latent factor models based on singular-value decomposition, and ensembles consisting of variations of these techniques. In this paper we investigate the application of forward stage-wise additive modeling to the Netflix problem, using two regression schemes as base learners: ensembles of weighted simple linear regressors and k-means clustering—the latter being interpreted as a tool for multivariate regression in this context. Experimental results show that our methods produce competitive results.

1 Introduction

Collaborative filtering is a challenging application domain for machine learning algorithms, which has gained prominence due to popular web-based recommendation services. The Netflix competition represents a particularly interesting instance of a collaborative filtering problem, in the form of a large movie recommendation dataset that consists of actual movie ratings generated in production use by real users.

The Netflix data is a very demanding problem for prediction methods, due to its size and sparsity. It consists of movie ratings for 17,770 movies, with 100,480,507 ratings in total. The ratings have been provided by 480,189 users. There are approximately 209 ratings for each user on average, so the data is very sparse. It can be viewed as a very large, sparse matrix with 17,770 columns and 480,189 rows. The task is to predict missing entries in this matrix, which correspond to unknown ratings. These predictions can then be used to provide recommendations to users.

One way of tackling this matrix completion problem is to view it as a regression task where the known ratings for a user are used to predict the missing ratings based on a regression model. This is the approach we investigate in this paper. Although the data can be held in memory in less than 1GB if it is stored in sparse form based on index-value pairs—using the data type `short` for the indices and the data type `byte` for the actual data values (all movie ratings are in the set $1, 2, 3, 4, 5$)—it is a challenging task to build regression models for it. In this paper we present two regression schemes, both based on forward stage-wise additive modeling (FSAM) [6], that are efficient enough to be applicable to this data. The first method uses FSAM in conjunction with an ensemble of simple linear regression models. In the second method, FSAM is used in conjunction with the k-means clustering method.

The paper is structured as follows. In the next section we describe two variants of the FSAM method, one for uni-variate prediction, and one for multi-variate prediction. In Section 3 we present our ensemble of simple linear regression functions, a heuristic base learner for the uni-variate FSAM scheme, and give some results obtained on the Netflix data. In Section 4 we describe how we applied k-means clustering in conjunction with the multi-variate FSAM scheme and present results. Section 5 briefly describes some related work. Section 6 has some conclusions.

2 Regression using forward stage-wise additive modeling

Forward stage-wise additive modeling (FSAM) [6] is a simple technique for fitting an additive model to a given prediction problem. In the case of classification, it is closely related to the well-known boosting methodology. Here we consider the regression version (see Equations 6 and 7 in [6]).

The output of the FSAM scheme is a collection of prediction models, i.e. an ensemble. In the following we call these individual models "base models". The base models are regression models that predict a numeric target based on a given set of independent variables. In this section we assume that we have suitable algorithms for fitting base models to the data. The corresponding algorithms we used for the Netflix problem are discussed in the next two sections.

The FSAM algorithm for additive regression is a greedy algorithm for fitting an ensemble of base models to a given regression problem. Base models are fit in a stage-wise manner, where each model in the sequence is trained on a transformed target consisting of the residual errors that remain from the models built so far. It is assumed that there is an implicit 0th model in the sequence that simply predicts the mean target value observed in the training data. Ensemble predictions are formed by simply adding the predictions of the base models in the sequence formed so far.

Assuming the algorithm used for building the base models minimizes the squared error of the respective residual-based predictions problems, the FSAM algorithm for regression greedily minimizes the squared prediction error of the ensemble as a whole.

The algorithm described so far predicts a single target based on some independent variables. It can be applied to the Netflix data on a per-movie basis, treating each movie individually as a target. However, the Netflix problem is essentially a matrix completion problem, where we would like to predict all missing entries for a user based on the existing entries, and we would like to predict them simultaneously because this would be computationally much more efficient. Fortunately it is straightforward to adapt the FSAM method to model multiple targets simultaneously, assuming the base learner is able to also do so: rather than working based on residuals of a single target variable only, we consider the residuals of all variables simultaneously. In each iteration of the FSAM method, a (sparse) matrix consisting of residual errors is passed to the base learner, which then builds a base model that approximates this matrix. The resulting residuals are then used in the next iteration, and so on. At prediction time, the predictions of the base models are again simply added together. In Section 4 we show how the k-means clustering algorithm can be used as the base learner in this multi-variate scheme.

As with many other learning algorithms, the FSAM method can suffer from overfitting. In practice, on the Netflix data, it is generally the case that the error of the FSAM ensemble decreases at first as more iterations are performed (i.e. more ensemble members are used), but it starts to increase after a certain point. A standard trick to delay the point of overfitting, thus making it possible to perform more iterations and potentially build a more accurate ensemble, is to apply shrinkage to the residuals [6]. In this case, the residuals remaining from the previous FSAM iteration are multiplied by a shrinkage parameter with a value in $(0, 1]$ before being passed to the base learner (see Equation 7 in discussion section of [6]). We will investigate the effect of the shrinkage parameter in our experiments with k-means in Section 4.

3 Ensembles of weighted simple linear regressors

The Netflix data is very high-dimensional: there are 17,770 possible movie ratings. This means it is desirable to apply a base learning algorithm in the FSAM method that is computationally efficient in the number of attributes. Another feature of this data is the large number of missing values: on average, the 480,189 users in this data have rated only about 209 movies. This means it is essential to exploit data redundancy at prediction time, making use of “redundant” attributes (i.e. columns in the matrix that correspond to different movies). This section presents a very simple and fast heuristic base learner that we have used successfully to apply the FSAM method to the Netflix data.

The basic idea is to to build a single simple linear regression model for each of the movies in the data, excluding the movie for which we want to predict ratings, to form an ensemble of (17,770 - 1) predictors. The simple linear regression models can be built very quickly: computing the sufficient statistics for a simple linear regression model requires a single pass through the (sparse) data.

Based on this ensemble of simple linear predictors it is then necessary to form an overall prediction for a new test case—a target movie rating for a particular user. Obviously only those linear regression models can be used to generate this prediction for which ratings for the user concerned are available in the training

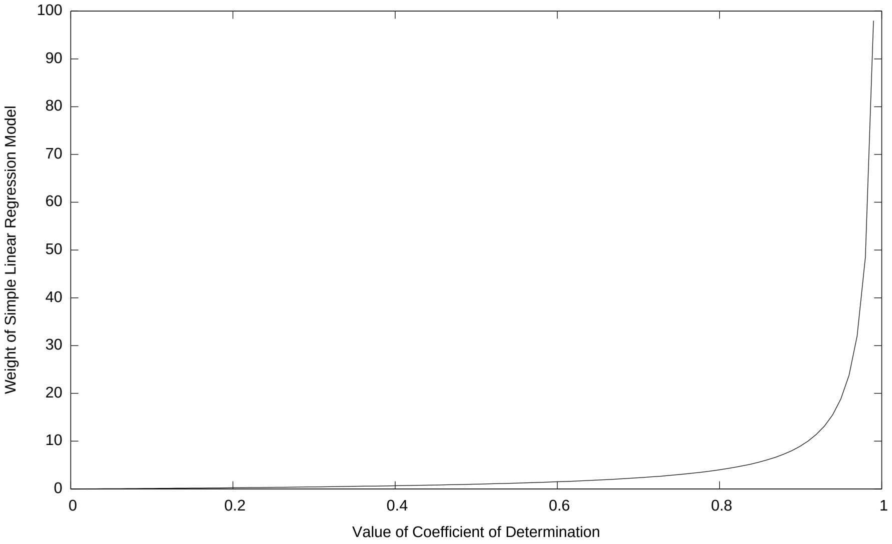

Fig. 1. Function used to compute weight of one simple linear regression model.

data. The most straightforward strategy is to simply average the predictions of the corresponding linear regression models. However, a straight unweighted average does not perform very well. Instead, we found that a simple weighting strategy produced much better results: each linear regression model's prediction is weighted based on the coefficient of determination of the linear model (i.e. the square of the correlation coefficient). It can be computed based on the same sufficient statistics as the linear regression model.

Assuming a value R^2 for the coefficient of determination, we compute the weight of a particular model as as $R^2/(1-R^2)$. This means that more predictive linear regression functions get a higher weight in the ensemble. The weight as a function of R^2 is shown in Figure 1. Using just R^2 as the weight did not appear to give a sufficiently high weight to the most accurate models among the different movie predictors. Hence we adopted this non-linear transformation function.

We also found that it is beneficial to discard simple linear regression models from the ensemble for which we have less than a certain number of movie ratings in the training data (i.e. models that do not have sufficient support in the training data). For the results in this paper we discarded all models that were based on less than 50 training instances.

Moreover, significantly improved performance was obtained by a simple data normalization step that is fairly standard for the Netflix data, namely by "centering" the input data for each user in the following way: for each user, a smoothed mean of the movie ratings for that user was subtracted from each rating for that user, and the maximum rating possible (the value 5) was added to the result. Assuming the user mean was μ_u, and n_u movie ratings were given

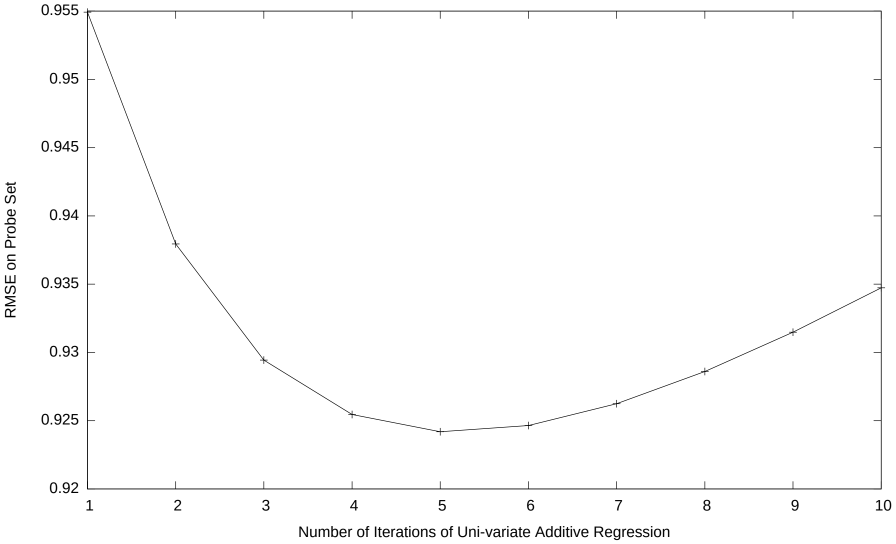

Fig. 2. Root mean squared error on probe set for uni-variate additive regression applied to ensembles of weighted simple linear regression models.

for that user in the training data, the smoothed user mean was computed as: $(1 - 1/(n_u - 1)) \times \mu_u + (1/(n_u - 1)) \times 3$. At prediction time, the smoothed mean for the corresponding user was added to the prediction obtained from the model and the value 5 subtracted from this. Resulting values outside the [1,5] interval were clipped. All the results that are stated in this section are based on data normalized in this fashion.

The resulting ensemble learner gives reasonable performance. It produces a root mean squared error (RMSE) of 0.955 on the Netflix probe set when the probe set is eliminated from the full training set so that an unbiased performance estimate can be obtained.[1] However, we can plug it into the uni-variate FSAM method from the previous section to obtain improved performance. In this case, the ensemble of simple linear regression models is not applied to model the values of the target movie directly; instead, it used to model the residual errors in the target predictions computed in the forward stage-wise additive modeling strategy. Applying this method to predicting the probe data, by building a model for each of the 17,700 possible target movies using 5 iterations of additive modeling, results in an RMSE of 0.924 on the probe data. This is in the same ball park as results obtained using variants of singular value decomposition [10, 9].

Figure 2 shows the RMSE on the probe set as the number of FSAM iterations is increased. These results were obtained without any shrinkage applied to

[1] The probe set specifies a validation set for models built for the Netflix problem and consists of 1,408,395 ratings from the full dataset, leaving 99,072,112 ratings for training.

the residuals in the FSAM method. We can see that after a certain number of iterations performance starts to deteriorate, indicating that overfitting occurs. It is possible to delay the point at which the error starts to increase by performing shrinkage, leading to further small reductions in error. We investigate the effect of shrinkage in more detail for the multi-variate case in the next section.

4 Modeling multiple targets simultaneously with *k*-means

A major drawback of the modeling strategy described in the previous section is that it requires building a separate additive model for each of the possible target movies. Given the large number of possible targets this is a time-consuming process. A much more efficient approach is to model all targets simultaneously by building a multi-variate model, a model that can be used to predict an arbitrary element in the data matrix. In the context of forward stage-wise additive modeling this only requires us to adopt an appropriate multi-variate base learner: as described in Section 2, it is trivial to modify the stage-wise modeling process to work with multiple residuals.

It turns out that there is a well-known modeling technique that slots nicely into this multi-variate version of the FSAM method, namely k-means clustering. It is well-known that k-means clustering finds cluster centers that represent a local minimum of the sum of squared differences to the cluster centers over all training examples. This is exactly what is required from the base learner in the FSAM method: we can simply cluster users according to the residual errors in the movie ratings obtained in the FSAM method (i.e. k-means clustering is applied to the matrix of residuals remaining in a particular FSAM iteration). The cluster centers are used for prediction: a user is assigned to its nearest cluster centroid based on the residuals that remain from the predictions of the previous ensemble members, and missing residuals for that user can be filled in based on the values stored in that centroid.

The Netflix data is sparse, with many missing values, and the basic method needs to be adapted to deal with this. Fortunately, this is simple: missing values are simply skipped in the distance calculation and in the process of computing the mean residuals stored in the cluster centroids. This also makes the process very efficient due to the sparse representation of the data that is used.

We found that basic k-means produced good results when used in this fashion. However, small improvements can be obtained by applying a global smoothing strategy when computing the cluster centroids: instead of using the per-cluster mean of residuals as the corresponding value for the cluster centroid, we smooth it with the global mean of residuals for the corresponding movie, obtained from all the values for the corresponding movie stored in the matrix. For the experimental results given below, we used a weight of $1/(1 + n_m)$ for the global mean, where n_m is the number of observed movie ratings for movie m, and one minus this weight for the local mean, when computing the smoothed cluster centroid.

Note that the idea of applying k-means to the Netflix data is not new (see, e.g, [10, 9]). However, what we propose here is to use k-means as the base learner

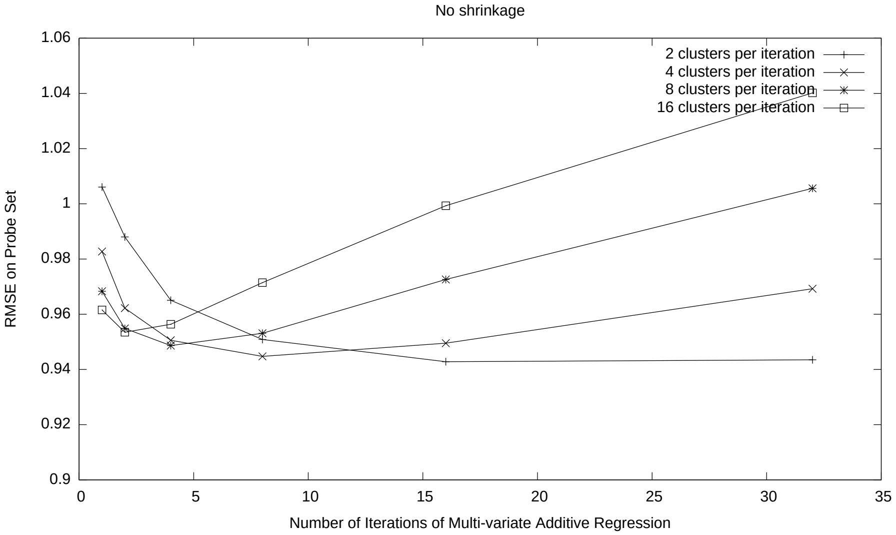

Fig. 3. Root mean squared error on probe set for multi-variate additive regression without shrinkage applied to k-means.

in the FSAM method. The k-means clustering algorithm is used to find centroids of the residuals produced in the FSAM modeling process, and these centroids are also used at prediction time. On the probe set, stand-alone k-means gets an RMSE of 0.962 with $k = 16$ and 0.959 with $k = 32$. We will see that better results can be obtained by using it in conjunction with additive regression.

There are several parameters in the multi-variate FSAM method with k-means: the number of FSAM iterations, the shrinkage value used in the FSAM method, and the number of cluster centers. Another parameter is the number of iterations of the k-means algorithm, but our experiments indicate that this parameter is less important. We fixed it at 40 iterations for the results discussed in the following.

Before presenting the actual RMSE values on the probe set we obtained, let us summarize our findings. We found that it is possible to get good results for different values of k when using k-means in conjunction with the FSAM method. Generally speaking, the larger the value of k, the fewer additive modeling iterations where required to achieve a certain level of performance. However, it was also necessary to adjust the shrinkage value of the additive modeling strategy appropriately. Lower shrinkage values were required for more iterations/cluster centers, to avoid overfitting.

Figure 3 shows the effect of increasing the number of FSAM iterations, for different numbers of clusters (i.e. values of k), keeping the shrinkage parameter fixed at 1 (i.e. no shrinkage). Figure 4 shows the same for a shrinkage value of 0.5, and Figure 5 the results for a shrinkage value of 0.25. The results are consistent with our above summary. It is beneficial to use more iterations of the

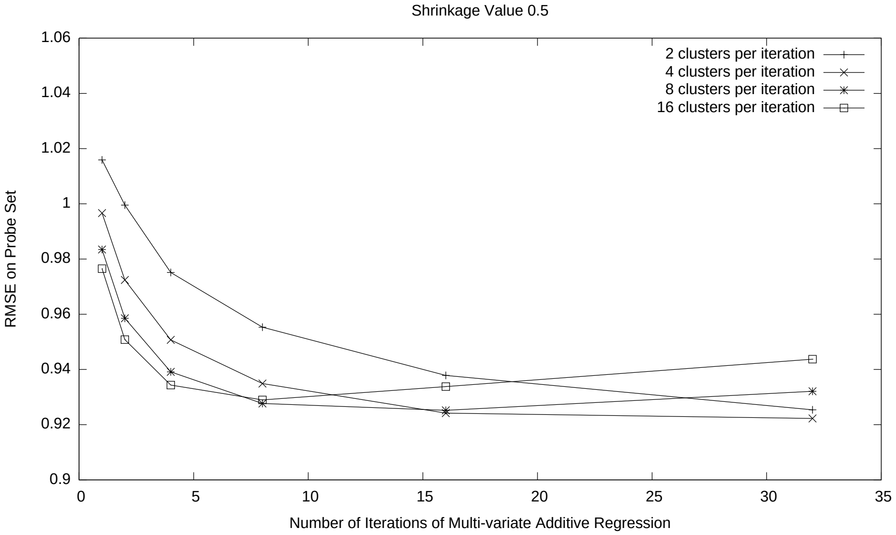

Fig. 4. Root mean squared error on probe set for multi-variate additive regression with shrinkage factor 0.5 applied to k-means.

FSAM method and/or more cluster centers, provided the value of the shrinkage parameter is adjusted appropriately.

In our experiments we obtained an RMSE value of 0.901 by using 4096 iterations of k-means, with k=2, and a shrinkage value of 0.05. Further improvements can be obtained by building multiple additive models using different random number seeds and/or values of k for k-means and averaging their predictions. In this fashion, it is possible to obtain RMSE values below 0.9 on the probe data as well as the actual Netflix test data, which is used in the Netflix competition to evaluate submissions. This appears to be competitive with the best single-paradigm learning schemes that have been evaluated on this data. The best results on the Netflix data so far appear to have been obtained based on large heterogeneous ensembles of diverse learning schemes [1, 10, 9]. Our schemes could be used as part of such ensembles.

We also experimented with a variant of bisection k-means, an efficient method of applying k-means in a recursive fashion to obtain multiple clusters. In this method, k-means is applied recursively with k=2 to build a tree of clusters in a divide-and-conquer fashion. We investigated this method because it makes it possible to apply a more fine-grained smoothing procedure, where a cluster center is smoothed by blending it with the cluster centers occurring in the path from the root node to its corresponding node in the tree. However, when we used this method in conjunction with additive regression, we did not observe significant increases in performance.

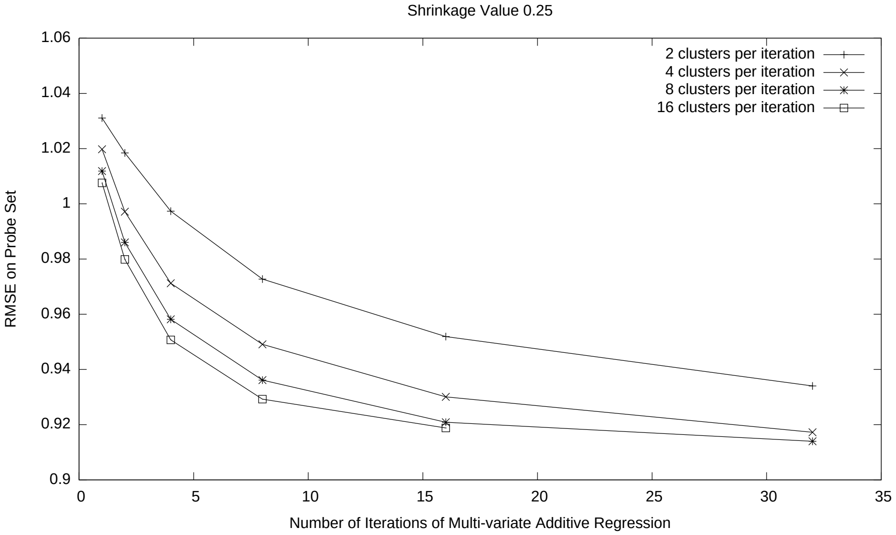

Fig. 5. Root mean squared error on probe set for multi-variate additive regression with shrinkage factor 0.25 applied to k-means.

5 Related work

As mentioned before, k-means has previously been applied to the Netflix data. However, there is also some work on using versions of the FSAM method for this data. Dembczynski et. al. [3] investigate ensembles of decision rules for ordinal classification constructed using FSAM. They present two approaches, one based on AdaBoost [4], and the other based on Friedman's gradient boosting machine [5]. The effectiveness of the methods is evaluated on the Netflix data, but only on a small subset of the total number of movies. This makes it difficult to judge how the performance of these methods compares with others on the full Netflix probe set.

Paterek [9] discusses using a linear regression model constructed for each movie. His method differs from the base learner we propose for our first method in three ways: firstly, a multiple linear regression is built for each movie based on binary vectors, that, for each user, indicate the movies that they have rated. Secondly, the prediction for a given rating is adjusted based on a weight proportional to the number of movies the user in question has rated. Finally, the model parameters are learned using gradient descent, which means that the method is relatively slow.

Nearest neighbor methods for collaborative filtering are discussed in [2]. For discussion of latent factor models, as applied to the Netflix problem, the reader is referred to [7–11]. Homogeneous and heterogeneous ensembles of neighborhood-based methods and latent factor models are discussed in [9–11].

6 Conclusions

This paper has discussed the use of forward stage-wise additive modeling (FSAM) in conjunction with regression schemes for uni-variate and multi-variate prediction on a large-scale collaborative filtering problem, namely the Netflix movie ratings data. Both regression schemes we investigated, ensembles of simple linear regressors for uni-variate prediction and k-means for multi-variate prediction, are fast enough to make FSAM tractable for this application. Results on the Netflix probe set show that both methods achieve good performance. Additive regression with k-means is a particularly attractive scheme because it makes it possible to build a single multi-variate prediction model for the data—effectively a single model that approximates the target matrix and can be used to fill in missing entries in this matrix.

References

1. Robert Bell, Yehunda Koren, and Chris Volinsky. Chasing $1,000,000: How we won the Netflix progress prize. *ASA Statistical and Computing Graphics Newsletter*, 18(2):4–12, 2007.
2. Robert M. Bell and Yehuda Koren. Improved neighborhood-based collaborative filtering. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
3. Krzysztif Denbczynski, Wojciech Kotlowski, and Roman Slowinski. Ordinal classification with decision rules. In *Proceedings of the 3rd International Workshop on Mining Complex Data*, pages 169–181. Springer, 2008.
4. Yoav Freund and Robert Shapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
5. Jerome Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.
6. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and rejoinder by the authors). *Annals of Statistics*, 28(2):337–407, 2000.
7. Miklos Kurucz, Andras A. Benczur, and Karoly Csalogany. Methods for large scale svd with missing values. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
8. Yew Jin Lim and Yee Whye Teh. Variational Bayesian approach to movie rating prediction. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
9. Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
10. Gabor Takacs, Istvan Pilaszy, Bottyan Nemeth, and Domonkos Tikk. On the gravity recommendation system. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
11. Mingrui Wu. Collaborative filtering via ensembles of matrix factorizations. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.