Ensemble Methods for Improving the Performance of Neighborhood-based Collaborative Filtering

Alon Schclar
Deutsche Telekom Laboratories at Ben-Gurion University,
Department of Information Systems Engineering,
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
schclar@bgu.ac.il

Alexander Tsikinovsky
Deutsche Telekom Laboratories at Ben-Gurion University,
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
tsikinov@gmail.com

Lior Rokach
Department of Information Systems Engineering,
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
liorrk@bgu.ac.il

Amnon Meisels
Department of Computer Science,
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
am@cs.bgu.ac.il

Liat Antwarg
Deutsche Telekom Laboratories at Ben-Gurion University,
Department of Information Systems Engineering,
Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
liatant@gmail.com

ABSTRACT

Recommender systems provide consumers with ratings of items. These ratings are based on a set of ratings that were obtained from a wide scope of users. Predicting the ratings can be formulated as a regression problem. Ensemble regression methods are effective tools that improve the results of simple regression algorithms by iteratively applying the simple algorithm to a diverse set of inputs. The present paper describes a simple and effective ensemble regressor for the prediction of missing ratings in recommender systems. The ensemble method is an adaptation of the AdaBoost regression algorithm for recommendation tasks. In all iterations, interpolation weights for all nearest neighbors are simultaneously derived by minimizing the root mean squared error. From iteration to iteration instances that are hard to predict are reinforced by manipulating their weights in the goal function that needs to be minimized. The experimental evaluation demonstrates that the ensemble methodology significantly improves the predictive performance of single neighborhood-based collaborative filtering.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

Ensemble Methods, Collaborative Filtering, Neighborhood based Collaborative Filtering

1. INTRODUCTION

In recent years we witness an ever growing increase in the utilization of recommender systems in various retailing settings since these systems prove to be an effective marketing tool which increases sales. Recommender systems are an integral part of many online stores such as Amazon.com, Buy.com, etc. One of the most famous examples of a recommender system is Netflix [1]. This system contains movie ratings for over $17,000$ movies provided by more than $480,000$ users. A contest whose purpose is to improve the results of the current Netflix recommender system is currently taking place.

The present paper focuses on collaborative filtering (CF) techniques in which customers grade the products they purchased. Given the supplied ratings, the system recommends other products that might be of interest to the users. At the core of the technique lies a prediction algorithm whose purpose is to approximate for every user the ratings of items they have not rated. These predictions are based on items that the user has rated and ratings provided by other users. According to the predicted ratings, the system recommends items which might interest the users. Usually, the system will recommend the items with the highest predicted ratings.

A predominant approach to collaborative filtering is neighborhood based (kNN - k-nearest neighbors). A user-item-preference rating is interpolated from ratings of similar items and/or users. While past kNN methods relate items (or users) by an arbitrary similarity function, modern kNN methods discover the interpolation weights by using a global optimization of a cost function pertaining to all weights simultaneously [6].

The main idea of the ensemble methodology is to combine a set of models, each of which solves the same original task, in order to obtain a better composite global model, with more accurate and reliable estimates or decisions than those produced by using a single model [7]. Ensemble classifiers can be classified into two types: homogeneous and heterogenous. Homogeneous ensembles utilize different versions of the same core model while the heterogenous approach combines models of different types. In the homogeneous approach, it is important to choose a simple and basic (*weak*) model (for example, in case of classification, one should use an inducer that is just a little better than the random one) in order to obtain an effective ensemble. Choosing a *strong* model will produce very little improvement since it does not provide enough diversity - one of the key principles of the ensemble methodology.

Experimental results show that ensembles can produce better results than any single model. Bell et al. [5], for instance, used a combination of 107 different models in their progress-prize winning solution to the Netflix challenge. In order to fuse the results from the ensembles they used a linear regression approach. Different base algorithms for creating the models were employed. Their findings show that it is better to use substantially different approaches than to refine a particular method. In [3], a boosting algorithm that is based on the AdaBoost algorithm, is proposed in order to provide recommendations in the form of *ranking*. The algorithm combines many *weak* rankings - each of them may have only a weak correlation to the target ranking. The output of ranking algorithms is a list of recommendations that are sorted according to their relevance. This is different from the output of our proposed algorithm in which the rating of every item is approximated.

The present paper introduces a homogeneous ensemble algorithm which is a modified version of the AdaBoost.RT ensemble regressor [8]. As a homogeneous ensemble, it incorporates a simple and effective regression algorithm which minimizes the prediction error by solving the gradient of an error cost-function. The AdaBoost algorithm, along with its different versions, is an effective ensemble method that is used to improve the results of a given inducer or regressor where the underlying principal is to reinforce instances that are harder to predict (produce a relatively high prediction error). The amount of reinforcement is determined according to a weight that is assigned to every instance and is iteratively updated. In each iteration, the regressor (or inducer) is applied to the instances and the current weights. The weights of the *problematic* instances are increased. Technically, the same effect is achieved by reducing the weights of the instances that are easy to predict and normalizing all the weights to sum up to 1. Initially, the same weight is assigned to all instances.

The contribution of the proposed algorithm is two-fold: first, it demonstrates how ensemble methods can be utilized in order to improve the performance of a single CF method; second, a novel adaptive data-driven instance-enforcement criterion is introduced into the ensemble regressor.

The rest of the paper is organized as follows: A detailed description of the proposed algorithm is given in Section 2. In Section 3 we provide experimental results to demonstrate the effectiveness of the proposed method. Conclusions are in Section 4.

2. THE PROPOSED ALGORITHM

Formally, the ratings in a recommender system can be represented as a set of m rating triplets $(u, i, r_{ui})_k$, $k = 1, \ldots, m$ where $u \in U$ is a user, $i \in I$ is an item and r_{ui} is the rating that user u gave item i. We denote by $|U|$ the number users and by $|I|$ the number of items. Usually, $m \ll |U| \cdot |I|$ since the number of items, $|I|$, is very high and users rate only the items they tried out. It is rare, if impossible, to find even a small subset of users that tried out all the items and also provided ratings for all of them. To make things worse, even if a user tried out an item, he or she might not provide their rating due to lack of motivation. The prediction algorithm produces a set of predicted ratings $(u, i, r_{ui})_k$, $k = 1, \ldots, |U| \cdot |I|$.

In each iteration of the proposed ensemble regression algorithm, the missing ratings are approximated by solving a minimization problem.

Let $\hat{r}_{ui}$ be the approximation of the rating of the i-th item by the u-th user. We set $\hat{r}_{ui}$ to be

$$\hat{r}_{ui} = b_{ui} + \sum_{\substack{v \in N(u) \\ i \in R(v)}} w_{uv} \left(r_{vi} - b_{vi}\right)$$

where $u = 1, \ldots, U; i = 1, \ldots, I$ and:

- w_{uv} is a weight for the influence of the ratings of user v on user u,
- b_{ui} is an initial approximation of the rating of item i by user u (b_{vi} is similarly defined),
- $N(u)$ is a set of users that are similar to the user u,
- and $R(v)$ is the list of items that user v rated.

$N(u)$ defines a neighborhood for a user u. The neighborhood includes the users whose ratings are similar to those of user u. The similarity is determined according to the Pearson correlation. Specifically, the users that are included in $N(u)$ are those who produce the top ranking (highest) correlations.

The initial approximation b_{ui} is defined as

$$b_{ui} = \mu + b_u + b_i$$

where μ is the average of all the provided ratings by all the users, b_u is the average of the provided ratings of user u and b_i is the average of all the provided rating of item i. Formally, given the indicator function

$$1_{ui} = \begin{cases} 1 & \text{if } r_{ui} \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

these quantities are given by:

$$\mu = \frac{\sum_u \sum_i r_{ui} \cdot 1_{ui}}{\sum_u \sum_i 1_{ui}}, \quad u = 1, \ldots, U;\ i = 1, \ldots, I$$

$$b_u = \frac{\sum_i r_{ui} \cdot 1_{ui}}{\sum_i 1_{ui}} - \mu, \quad u = 1, \ldots, U$$

and

$$b_i = \frac{\sum_u (r_{ui} - (b_u + \mu)) \cdot 1_{ui}}{\sum_u 1_{ui}}, \quad i = 1, \ldots, I$$

We can now formalize the minimization problem. Define the total error of the approximation by the function

$$E(W) = \sum_u \sum_{i \in R(u)} \gamma_{ui} (\hat{r}_{ui} - r_{ui})^2$$

where γ_{ui} is a weight that is assigned to every rating. These weights are determined by the ensemble algorithm that is described in Section 1.

The approximation is obtained by the minimization of $E(W)$ which is found by solving $\nabla E(W) = 0$. We assume that w_{uv} is not necessarily equal to w_{vu} i.e. the ratings of user u do not influence the rating of user v by the same amount as the ratings of user v influence the rating of user u. Accordingly, the gradient of $E(W)$ is given by

$$\begin{aligned} \nabla E(w_{uv'}) &= \textstyle\sum_{i \in R} \Big[2\gamma_{ui} \Big(b_{ui} + \sum_{v \in N(u)} w_{uv} (r_{vi} - b_{vi}) \\ &\quad - r_{ui} \Big) \cdot (r_{v'i} - b_{v'i}) \Big], \quad v' \in N(u) \end{aligned}$$

where $N(u)$ and $R(u)$ were defined above. Thus, for every similarity weight, we obtain a system of $|N(u)|$ equations in $|N(u)|$ variables. Adding the constraint

$$\sum_{v \in N(u)} w_{uv} = 1$$

results in a system of $N(u) + 1$ linear equations in $|N(u)|$ variables.

Before solving any of the obtained systems of equations, one must first incorporate the following adjustments:

1. In case r_{vi} does not exist, it is set to be b_{vi}. This situation can occur although $v \in N(u)$, since $N(u)$ is determined according to *all* items that were rated by users u and v. Thus, v will still be included in $N(u)$ although there are items that were not rated by her but were rated by u.

2. When users u and v have the same rating values and the same free term values, the system of equations has an infinite number of solution where the free variables are a linear combination of the other variables. In this case, one of the solutions is chosen.

3. When user u and v have the same rating values with different free term values, there is no solution to the system. In order to obtain a solution, one of the equations is chosen arbitrarily.

After the predictions are obtained, ratings that are above and below the rating range are truncated to the maximal and minimal ratings, respectively.

2.1 The ensemble regressor

The ratings obtained by the proposed algorithm are enhanced by using ensemble regression. Specifically, we use a modified version of the *AdaBoost.RT* algorithm [8]. The AdaBoost-based algorithms improve the results of a classifier (or regressor) via an iterative process which reinforces instances that are harder to predict (or approximate). The AdaBoost.RT algorithm incorporates a relative-error parameter ϕ in the weight reinforcement criterion. Only instances whose relative errors exceed ϕ are reinforced. Both ϕ and the number of iterations are given as parameters. Our proposed scheme differs from the AdaBoost.RT algorithm by the reinforcement criterion that it uses. Rather than using a relative error ϕ, a deviation factor α is used. In each iteration we calculate the mean and standard deviation of the prediction errors. Only instances whose prediction errors exceed the mean by a given factor α multiplied by the standard deviation are reinforced. Although the algorithm still requires a parameter, the advantage in employing this scheme is two-fold: first, the reinforcement threshold is dynamic, adapting itself at each iteration to the obtained errors; and second, the reinforcement is determined according to the error statistics rather than a predefined threshold.

Both our proposed method and the *AdaBoost.RT* algorithm [8] use the same reinforcement factor for all the *hard-to-predict* instances i.e., their weights are multiplied by the same factor. This is contrary to the *AdaBoost.R2* algorithm [2] where the reinforced weight of each individual instance is determined according to the magnitude of its corresponding prediction error. Technically, the reinforcement effect is achieved by reducing the weight of instances that are easy to approximate (achieve a low prediction error) and normalization of the resulting weights so their sum will be equal to 1.

The proposed ensemble algorithm is described in Algorithm 1. The input to the algorithm is a set of m rating triplets as defined above. The algorithm produces a set of predicted ratings $(u, i, r_{ui})_k$, $k = 1, \ldots, |U| \cdot |I|$ where $|U|$ is the number users and $|I|$ is the number of items.

3. EXPERIMENTAL RESULTS

The proposed algorithm was tested on the *MovieLens* rating database. The database contains 3900 distinct movies and 6040 users. The predictive model is induced for 100 randomly selected users. The training set was constructed from 70% of the ratings of the selected users. The algorithm was used to predict the remaining 30% ratings. For example, if a user rated 20 movies, 14 of them were used for the training and the other 6 were included in the test set. Although, the predictive model refers to only 100 users, the users that were included in $N(v)$ were selected from the entire database.

The size of $N(v)$ was set to 50. The effectiveness of each iteration is measured according to the root mean square error (RMSE). Figure 1 shows the decrease in the RMSE with each iteration demonstrating the improvement that is obtained by using the proposed ensemble algorithm. In ten iterations of the ensemble algorithm, the RMSE decreased by **41.86%** where the most dramatic improvement - 21.03% - was achieved after the first iteration. We compared our results to those obtained by the *SVD++* [4] algorithm.

4. CONCLUSION AND FUTURE WORK

An ensemble regression algorithm for the approximation of rating values in a k-NN recommender system was introduced. The regression algorithm, which is at the heart of every ensemble iteration, minimizes the root mean squared prediction error by directly solving the gradient of a cost function. This is achieved via overdetermined systems of lin-

Algorithm 1 Modified AdaBoost.RT ensemble regressor

```
Input.
1. A set of $m$ rating triplets $(u,i,r_{ui})_k$ , $k = 1,\ldots,m$.
2. The rating prediction algorithm from the beginning of Section 2.
3. Integer $T$ specifying the number of iterations.
4. Error factor $\alpha$ (default $\alpha = 1$).

Initialize.
1. Set iteration number $t = 1$.
2. Set a weight to every triplet according to a uniform distribution $\gamma_k^t = 1/m$.

Iterate. While $t < T$
1. Call the rating prediction algorithm to obtain the predicted ratings $\{\hat{r}_{ui}^t\}$.
2. Calculate the mean, $\bar{\varepsilon}^t$, and the standard deviation, $\sigma^t$, of the errors $\{|\hat{r}_{ui}^t - r_{ui}^t|\}$.
3. Calculate the error rate
       $\nu^t = \left(\sum_{(u,i)\in Q} \gamma_{ui}^t\right)^2$
   where $Q = \{(u,i) : |\hat{r}_{ui}^t - r_{ui}^t| - \bar{\varepsilon}^t > \alpha \cdot \sigma^t\}$.
4. Set the weights of the instances for the next iteration according to
       $\gamma_k^{t+1} = \frac{\gamma_k^t}{N_t} \cdot \begin{cases} \nu^t & \text{if } k \notin Q \\ 1 & \text{otherwise} \end{cases}$
   where $N_t$ is a normalization factor whose value is chosen such that $\{\gamma_k^{t+1}\}$ form a distribution.
5. Set $t = t + 1$

Output.
  • The final predicted rating
       $\hat{r_k} = \frac{\sum_t r_k^t \cdot \log(1/\nu^t)}{\sum_t \log(1/\nu^t)}$
```

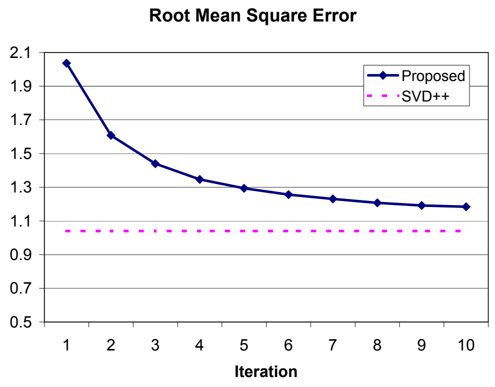

Figure 1: A plot of the root mean square prediction error as a function of the iteration number. An improvement is clearly noticed. The total improvement that is achieved by the ensemble algorithm is 41.86%.

ear equations. Occasionally, these systems cannot be solved due to similar equations whose free terms differ. Currently, this situation is resolved by arbitrarily choosing one of the equations. In some cases, the linear system is underdetermined. At present, the algorithm chooses one of the possible solutions. An improvement to both of these situations is to choose the solution which minimizes the RMSE. Furthermore, the method is going to be extended and tested on larger datasets such as the Netflix [1] dataset.

The proposed scheme achieves good results which render ensemble regressors that are based on a weak collaborative filtering algorithm as a legitimate tool for recommender systems. However, further research should be done in order to extend the proposed approach so that it achieves the results obtained by stronger collaborative filtering algorithms such as those based on SVD matrix-factorization.

Acknowledgements

The authors are grateful to Dr. Yehuda Koren for his invaluable guidelines in the implementation of his SVD++ algorithm.

5. REFERENCES

[1] Netflix, inc. "netflix prize", www.netflixprize.com.

[2] H. Drucker. Improving regressor using boosting. In D. H. Fisher Jr., editor, *Proceedings of the 14th International Conference on Machine Learning*, pages 107–115. Morgan Kaufmann, 1997.

[3] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[4] 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, New York, NY, USA, 2008. ACM.

[5] Y. Koren R. M. Bell and C. Volinsky. The bellkor solution to the netflix prize, 2007.

[6] Y. Koren R. M. Bell and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proc. 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[7] Lior Rokach. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, In Press, Corrected Proof:–, 2009.

[8] D. P. Solomatine and D. L. Shrestha. Adaboost.rt: A boosting algorithm for regression problems. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 1163–1168, 2004.