Improving Simple Collaborative Filtering Models Using Ensemble Methods

Ariel Bar[1], Lior Rokach[1], Guy Shani[1], Bracha Shapira[1], and Alon Schclar[2]

[1] Department of Information Systems Engineering
Ben-Gurion University of the Negev, Beer-Sheva, Israel
{arielba, liorrk, shanigu, bshapira}@bgu.ac.il
[2] School of Computer Science, Academic College of Tel Aviv-Yaffo
P.O.B 8401, Tel Aviv 61083, Israel
alonschc@mta.ac.il

Abstract. In this paper we examine the effect of applying ensemble learning to the performance of collaborative filtering methods. We present several systematic approaches for generating an ensemble of collaborative filtering models based on a single collaborative filtering algorithm (single-model or homogeneous ensemble). We present an adaptation of several popular ensemble techniques in machine learning for the collaborative filtering domain, including bagging, boosting, fusion and randomness injection. We evaluate the proposed approach on several types of collaborative filtering base models: k-NN, matrix factorization and a neighborhood matrix factorization model. Empirical evaluation shows a prediction improvement compared to all base CF algorithms. In particular, we show that the performance of an ensemble of simple (weak) CF models such as k-NN is competitive compared with a single strong CF model (such as matrix factorization) while requiring an order of magnitude less computational cost.

Keywords: Recommendation Systems, Collaborative Filtering, Ensemble Methods.

1 Introduction

Collaborative Filtering is perhaps the most successful and popular method for providing predictions over user preferences, or recommending items. For example, in recent Netflix competitions, CF models were shown to provide the most accurate models. However, many of these methods require a very long training time in order to achieve high performance. Indeed, researchers suggest more and more complex models, with better accuracy, at the cost of higher computational effort.

Ensemble methods suggest that a combination of many simple identical models can achieve a performance of a complex model, at a lower training computation time. Various ensemble methods create a set of varying models using the same basic algorithm automatically, without forcing the user to explicitly learn a single set of model parameters that perform the best. The predictions of the resulting models are com-

adfa, p. 1, 2011.

bined by, e.g., voting among all models. Indeed, ensemble methods have shown in many cases the ability to achieve accuracy competitive with complex models. In this paper we investigate the applicability of a set of ensemble methods to a wide set of CF algorithms. We explain how to adapt CF algorithms to the ensemble framework in some cases, and how to use CF algorithms without any modifications in other cases. We run an extensive set of experiments, varying the parameters of the ensemble. We show that, as in other Machine Learning problems, ensemble methods over simple CF models achieve competitive performance with a single, more complex CF model at a lower cost.

2 Background and Related Work

Collaborative Filtering (CF) [1] is perhaps the most popular and the most effective technique for building recommendation systems. This approach predicts the opinion that the active user will have on items or recommends the "best" items to the active user, by using a scheme based on the active user's previous likings and the opinions of other, like-minded, users. The CF prediction problem is typically formulated as a triplet (U, I, R), where:

- U is a set of M users taking values form $\{u_1, u_2 \ldots u_m\}$.
- I is a set of N items taking values from $\{i_1, i_2 \ldots i_n\}$
- *R* - The ratings matrix, is a collection of historical rating records (each record contains a user id ($u \in U$), an item id ($i \in I$), and the rating that u gave to $i - r_{u,i}$.

A rating measures the preference by user u to item i, where high values mean stronger preferences. One main challenge of CF algorithms is to give an accurate prediction, denoted by $\hat{r}_{u,i}$ to the unknown entries in the ratings matrix, which is typically very sparse. Popular examples of CF methods include k-NN models [1-2] and Matrix Factorization models [3].

Ensemble is a machine learning approach that uses a combination of identical models in order to improve the results obtained by a single model. Unlike hybridization methods [4] in recommender systems that combine different types of recommendation models (e.g. a CF model and a content based model), the base models which construct the ensemble are based on a single learning algorithm.

Most improvements of collaborative filtering models either create more sophisticated models or add new enhancements to known ones. These methods include approaches such matrix factorization [3],[5], enriching models with implicit data[6], enhanced k-NN models [7], applying new similarity measures [8], or applying momentum techniques for gradient decent solvers [5].

In [9] the data sparsity problem of the ratings' matrix was alleviated by imputing the matrix with artificial ratings, prior to building the CF model. Ten different machine learning models were evaluated for the data imputing task, including an ensemble classifier (a fusion of several models). In two different experiments the ensemble approach provided lower MAE (mean absolute error). Note that this ensemble approach is a sort of hybridization method.

The framework presented in [10] describes three matrix factorization techniques, different in their parameters and constraints solving the matrix formation optimization problem. The best results (minimum RMSE – root mean square error) were achieved by an ensemble model which was constructed as a simple average of the three matrix factorization models.

Recommendations of several k-NN models are combined in [11] to improve MAE. The suggested model was a fusion between the User-Based CF approach and Item-Based CF approach. In addition the paper suggests lazy Bagging learning approach for computing the user-user, or item-item similarities.

In [12] a modified version of the AdaBoost.RT ensemble regressor (AdaBoost [13] variant designed for regression tasks) was shown to improve the RMSE measure of a neighborhood matrix factorization model. The authors demonstrate that adding more regressors to the ensemble reduces the RMSE (the best results were achieved with 10 models in the ensemble).

A heterogeneous ensemble model which blends five state-of-the-art CF methods was proposed in [14]. The hybrid model was superior to each of the base models. The parameters of the base methods were chosen manually.

The main contribution of this paper is a systematic framework for applying ensemble methods to CF methods. We employ automatic methods for generating an ensemble of collaborative filtering models based on a single collaborative filtering algorithm (homogeneous ensemble). We demonstrate the effectiveness of this framework by applying several ensemble methods to various base CF methods. In particular, we show that the performance of an ensemble of simple (weak) CF models such as k-NN is competitive compared with a single strong CF model (such as matrix factorization) while requiring an order of magnitude less computational cost.

3 Ensemble Framework

The proposed framework consists of two main components: (a) the ensemble method; and (b) the base CF algorithm. We investigate four common ensemble methods: Bagging, Boosting (a variant of AdaBoost), Fusion and Randomness Injection. These methods were chosen due to their improved accuracy when applied to classification problems, and the diversity in their mechanisms.

The Bagging and AdaBoost ensembles require the base algorithm to handle datasets in which samples may appear several times, or datasets where weights are assigned to the samples (equivalent conditions). Most of the base CF algorithms assume that each rating appears only once, and that all ratings have the same weight. In order to enable application of Bagging and Boosting, we modify the base CF algorithms to handle recurring and weighted samples. We evaluate four different base (modified) CF algorithms: k-NN User-User Similarity; k-NN Item-Item Similarity; Matrix Factorization (three variants of this algorithm) and Factorized Neighborhood. The first three algorithms are simpler, having a relatively low accuracy and rapid training time, while the last two are more complex, having better performance and higher training cost.

4 Ensemble Methods For CF

4.1 Bagging

The Bagging approach (Fig.1) [15] generates k different bootstrap samples (with replacement) of the original dataset where each sample is used to construct a different CF prediction model. Each bootstrap sample (line 2) is in the size of the original rating data set. The base prediction algorithm is applied to each bootstrap sample (line 3) producing k different prediction models. The ensemble model is a simple average over all the base ones (line 5).

Input:
- T – Training dataset of ratings <*U,I,R*>
- K – ensemble size.
- *BaseCF*– the Base CF prediction algorithm.

Output: *Bagging ensemble*

Method:

1. **for** i= 1 to K do:
2. Create a random bootstrap sample T_i, by sampling T with replacemen
3. Apply the *BaseCF* to T_i and construct M_i.
4. **end for**

The prediction rule of the model is:

$$\hat{r_{ui}} = \sum^{K} r_{u_i}^{M_i} / K$$

Fig. 1. Bagging algorithm for CF

4.2 Boosting

AdaBoost [15] is perhaps one of the most popular boosting algorithms in machine learning. In this approach, weights are assigned to each rating tuple, while an iterative process constructs a series of K models. After model M^i is learned, the weights are updated to allow the subsequent model, M_{i+1}, focus on the tuples that were poorly predicted by M_i. The ensemble model combines the predictions of each individual model via a weighted average according the accuracy of each model.

In this work we evaluated several variants of the AdaBoost.RT [16] algorithm. Initial experiments with the original algorithm resulted with poor accuracy models; for all evaluated configurations, the ensemble model either had a negligible accuracy improvement or even an overall accuracy decrease compared to the original base model; Thus, we replaced the original relative error function with a pure absolute one as presented in Fig.2 (line 6 in the pseudo code). This modification resulted with improvement of the original algorithm. As suggested in the original work, we initialize δ the demarcating threshold criteria to be the AE (the model error) of the original dataset. During the calibration process of the algorithm we evaluated different values for "n" (line 8 in the pseudo code), controlling the distribution function. The best results

were achieved when we set n=1. We noticed that both the original and modified algorithms were highly unstable with n=2 or 3, as the accuracy of the final ensemble model decreased substantially.

Input:

- T – Training dataset of ratings <*U,I,R*>
- K – the ensemble size.
- *BaseCF*– the Base CF prediction algorithm
- δ – Threshold (0 <δ<*the rating score range*) for demarcating correct and incorrect predictions

Output: *AdaBoost.RT model*

Method:

1. Assign iteration number *t=1*
2. Assign initial distribution for each tuple in *R*: $D_t(r_{ui}) = 1/|R|$
3. **while** $t \leq K$ Do
4. Apply *BaseCF* to *T* with distribution D_t, and construct the model M_t.
5. **for each** rating $r_{ui} \in R$
6. calculate $AE_t(r_{ui}) = | \hat{r}_{ui}^{Mt} - r_{ui} |$
7. calculate error rate of iteration *t*:
$$\varepsilon_t = \sum_{ui:AE_t(r_{ui})>\delta} D_t(r_{ui})$$
8. Set $\beta_t = \varepsilon_t^n$
9. Update distribution D_{t+1} as: //Z_t = normalization factor
$$D_{t+1}(r_{ui}) = \frac{D_t(r_{ui})}{Z_t} \times \begin{cases} \beta_t & if\ AE_t(r_{ui}) \leq \delta \\ 1 & otherwise \end{cases}$$
10. Set t= t+1
11. **end while**

The prediction rule of the model is:
$$\hat{r}_{ui} = \sum_{i=1}^{K} \log(\frac{1}{\beta_i}) \cdot \hat{r}_{ui}^{Mi} / \sum_{i=1}^{K} \log(\frac{1}{\beta_i})$$

Fig. 2. AdaBoost.RT algorithm for CF

4.3 Fusion

A straightforward way to construct an ensemble is to take a specific prediction algorithm, and use it several times on the same dataset, but each time with different initial parameters [17]. This process constructs different models, which can later be combined together by e.g. averaging. For the k-NN algorithms, we applied the following three fusion schemas:

1. Fusion by similarity metric - we combined the predictions of a two k-NN with different similarity measures (Pearson and Cosine.)
2. Fusion by CF perspective - we combined the predictions of the User-User k-NN model, and the Item-Item k-NN model.

3. k-NN Fusion by CF perspective & similarity metric - combination of the two previous fusion schemes (total of four models in the ensemble).

For matrix factorization, we applied fusion to models which were constructed using different vector sizes of the latent factors.

4.4 Randomness Injection

All ensemble methods described so far in this section are generic in the sense that they are not limited to a specific CF prediction algorithm. Thus one of their parameters is BaseCF- the base CF prediction algorithm, which is used to construct the base models in the ensemble.

A different approach to create an ensemble is to take a base algorithm and modify it such that it will create various sub models and combine their results. A popular way to achieve this is by introducing randomness to the basic learning schema. By doing so, it is possible to run the algorithm several times, and receive a different model each time. These models can then be joined to provide a combined prediction. In this work the randomness to the CF algorithms was injected as follows:

Random k-NN - Instead of selecting the top k nearest neighbors (users or items) for the prediction rule, we randomly select any k users/items from the top 2*k nearest neighbors. We can repeat this process K times (the ensemble size) to get K different predictions, and then use a simple average on them for the final one.

The MF algorithms are naturally randomized, since in the initialization process of the learning phase, we assign small random numbers to the latent factors. If we simply repeat this process K times (the ensemble size), each time with random initial values, we will receive K different MF models. These models can then be combined into an ensemble by a simple average.

5 Modified CF Algorithms

Some ensemble methods require that the base prediction algorithm can handle datasets with reoccurring or weighted samples. Accordingly, we had to modify CF algorithms which assume that each rating appears once, and that all ratings weights are equal. The first step in our modification was to update the original CF prediction problem from Section 2, by adding a new element W to the problem formalization. W is a vector of weights whose size is equal to the number of ratings, where $w_{u,i}$ indicates the relative distribution of $r_{u,i}$. It is important to notice that when all weights are equal, all modified algorithms coincide with the original ones.

5.1 Modified k-NN Algorithms

The main modification in these algorithms is to include the rating's weights into the similarity measures. For the user-user k-NN prediction, we suggest using the modified Pearson correlation coefficient, and the modified cosine-based similarity measures as described in Eqs.1 and 2 respectively, where S(u,v) is the set of items that both users

u and v rated, r_u is the weighted average rating of user u, and w_{uvj} is the maximum between w_{uj} and w_{vj}. We use the same modified measures for item-item similarity, updating the required indices.

$$pearson(u,v) = \frac{\sum_{j \in S(u,v)} w_{uvj}(r_{u,j} - r_u)(r_{v,j} - r_v)}{\sqrt{\sum_{j \in S(u,v)} w_{uvj}(r_{u,j} - r_u)^2 \sum_{j \in S(u,v)} w_{uvj}(r_{v,j} - r_v)^2}} \quad (1)$$

$$\cos ine(u,v) = \cos(\vec{u},\vec{v}) = \frac{\vec{u} \bullet \vec{v}}{||\vec{u}||_2 \times ||\vec{v}||_2} = \frac{\sum_{j \in S(u,v)} w_{uvj} r_{u,j} r_{v,j}}{\sqrt{\sum_{j \in S(u,v)} w_{uvj} r_{u,j}^2} \sqrt{\sum_{j \in S(u,v)} w_{uvj} r_{v,j}^2}} \quad (2)$$

5.2 Modified MF Algorithms

In this work we modified the ISMF, RISMF and BRISMF algorithms from [5] to handle weighted datasets. The modified algorithm continues to the minimize SSE (Sum of Square Errors), while applying new gradient steps as presented in Eq.3, taking into consideration the associated weight for each rating.

$$p'_{uk} = p_{uk} + \eta \cdot w_{ui} \cdot (e_{ui} \cdot q_{ki} - \lambda \cdot p_{uk}) \quad (3)$$

$$q'_{ki} = q_{ki} + \eta \cdot w_{ui} \cdot (e_{ui} \cdot p_{uk} - \lambda \cdot q_{ki})$$

In a similar way, we modify the Factorized Neighborhood Model (FNM) [7] to handle weighted datasets. We chose this model due to the high accuracy of its predications. The new gradient steps are presented in Eq. 4.

$$\begin{aligned} q_i &= q_i + \eta \cdot w_{ui}(e'_{ui} \cdot p_u - \lambda \cdot q_i) \\ b_u &= b_u + \eta \cdot w_{ui}(e'_{ui} - \lambda \cdot b_u) \\ b_i &= b_i + \eta \cdot w_{ui}(e'_{ui} - \lambda \cdot b_i) \\ x_i &= x_i + \eta \cdot w_{ui} \cdot (|R(u)|^{-\alpha} \cdot (r_{ui} - b_{ui}) \cdot sum\_error - \lambda \cdot x_i) \\ y_i &= y_i + \eta \cdot w_{ui} \cdot (|N(u)|^{-\alpha} \cdot sum\_error - \lambda \cdot y_i) \end{aligned} \quad (4)$$

6 EVALUATION

6.1 Experimental Setup

The evaluation of the algorithms described in sections 4 and 5 was mainly based on the 100K MovieLens dataset. We used RMSE for measuring accuracy over 5 different random 80:20 on the dataset. We compared the following configurations: all k-NN models were evaluated by applying the modified Pearson and Cosine similarity measures and 3 k-NN sizes (5, 10, 20). The three matrix factorization algorithms from section 5.2, were evaluated using different sizes of latent factors (3, 4, 5, 10, 20, 30, 40, 50); The Factorized Neighborhood Mode algorithm (FNM) was evaluated using

different latent factor sizes: 3, 4, 5, 10, 20 and 30, all other parameters of the MF algorithms were consistent with the original work. For each configuration we evaluated its original RMSE (baseline) without any ensemble enhancement. We apply all ensemble methods from Section 4 to each configuration with different ensemble sizes and compare the results to the baseline: The ensemble size of Bagging ranged from 5 to 20 for the k-NN algorithms, and from 5 to 50 for the MF algorithms; The ensemble size of AdaBoost.RT ranged from 1 to 10; The ensemble size of Fusion was either 2 or 4 for the k-NN algorithm, and ranged between 5 and 10 for the MF algorithms; Finally, the ensemble size of Randomness Injection was set from 1 to 10. We now report various results and insights from these experiments.

6.2 Accuracy Results

Due to space restrictions, we are unable to report all possible RMSE results. We therefore limit Table 1 to the best configuration of each method. For example, from all k-NN User-User ensemble configurations using Bagging, the ensemble over k=20 produced the best results and is hence reported in the table.

We organized the base CF model according to their relative "strength", where simple/less accurate models appear on the left, and more advanced/complex/accurate appear on the right. The final row of the table indicates the improvement percentages of the best ensemble model compared to the baseline model.

Table 1. ML (100K ratings) Accuracy Results (RMSE)

CF / Ensemble	k-NN-User	k-NN-Item	ISMF	RISMF	BRISMF	FNM
Baseline	0.9535	0.9526	0.9434	0.9407	0.9268	0.9231
Bagging	0.9495 (20)	0.9464 (20)	***0.9173 (50)***	***0.9152 (50)***	***0.9170 (50)***	0.9333
AdaBoost.RT	0.9410 (10)	0.9459 (10)	0.9397 (10)	0.9415	0.9332	0.9367
Fusion	***0.9383 (4)***	***0.9383S (4)***	0.9411 (10)	0.9383 (10)	0.9241 (10)	0.9158 (10)
Random	0.9462 (10)	0.9437 (10)	0.9407 (10)	0.9381 (10)	0.9237 (10)	**0.9153 (10)**
Improvement	1.57%	1.47%	2.76%	2.66%	0.97%	0.87%

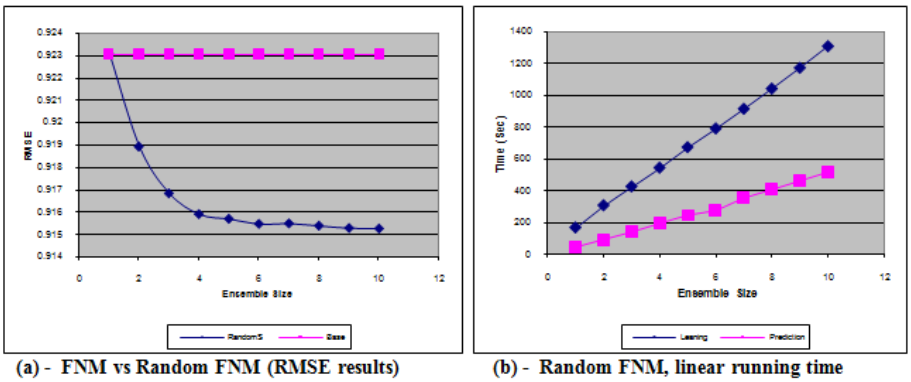
We use the following notations in the table: ensemble enhancements which improved with statistical significance the RMSE measure over the baseline accuracy are presented with the ensemble size (in parentheses). The best model in each column is displayed in bold-face font. Ensemble models of relatively weak algorithms which improve the RMSE to a level of more advanced models are displayed in italic font. We check for statistical significance using One-Way ANOVA with repeated measures (applying the Greenhouse-Geisser test) with confidence level $\alpha = 0.05$, followed by a simple paired t-test, with confidence level $\alpha = 0.05$.

Our results indicate the following: We were able to significantly improve the baseline results of every base CF model type in our work, by at least two different ensemble approaches; The improvement level was between 0.87% and 2.76%. These improvements may seem modest, but lowering the RMSE is a difficult problem, and every reduction in RMSE is difficult to achieve. The improvement level depends on

the base CF models - more complex models are more difficult to improve. This agrees with the idea that ensemble should be applied to boost the performance of weak CF models, not to improve complex models. The Fusion and Random Injection ensemble methods were able to improve the accuracy of all base CF models; Bagging failed to improve FNM, and AdaBoost failed to improve RISMF, BRISMF and FNM, however, these ensemble approaches may achieve better results than other ensembles, when applied to other base CF algorithms. The performances of the suggested boosting approaches on the matrix factorization models require exploring additional boosting methods such as the Stochastic Gradient Boosting [18] or designing special ones for this task. In the spirit of the "No Free Lunch" theorem, none of the evaluated ensemble method was optimal for all given scenarios. Consequently, one should look for the (base model, ensemble) pair that achieves the best results for the dataset at hand.

6.3 The Effect of the Ensemble Size

Table 1 show that if the ensemble method improves the accuracy of the basic model, then the ensemble model that achieved the best result is the one with the highest number of members. Consequently, the strategy in this case is to use as many ensemble members as possible provided that the improvement is significant, and feasible with the amount of computation resources. Fig.3 demonstrates this idea by using Randomness injection on FNM. Adding more members to the ensemble may be practical, as the complexity of all the ensemble methods grows linearly in the number of ensemble members. Fig.3a also demonstrates that the accuracy improves with the ensemble sizes , the more members of the ensemble, the higher the accuracy, up to a certain limit, after which the improvement is marginal.(e.g.: adding the first member to the ensemble improved the accuracy by 0.43%, while the last member contributed only 0.02% improvement). This observation was stable in all the experiments of our studied ensemble methods, leading us to define the maximum ensemble sizes as described in section 6.1, as in all evaluated models the last ensemble member contributed up to 0.07% RMSE improvement which is considered insignificant.



(a) - FNM vs Random FNM (RMSE results) (b) - Random FNM, linear running time

Fig. 3. The effect of the ensemble size in Random FNM

6.4 Computational Cost and Accuracy Tradeoff

As described in sub-section 6.2 in several scenarios an ensemble of relatively weak models achieved better accuracy than a single stronger model. Fig.4 present the RMSE obtained by various methods as function of the computation cost (training time - presented in log scale). The graph shows the following results: An ensemble of the k-NN-User method achieves competitive performance with two MF methods (ISMF and RISMF) at an order of magnitude less computational cost (4 seconds instead of 24-26). An ensemble of MF methods (ISMF and RSIMF) achieves a competitive performance with a BRISMF method at a much lower computational cost (170 seconds instead of 490).

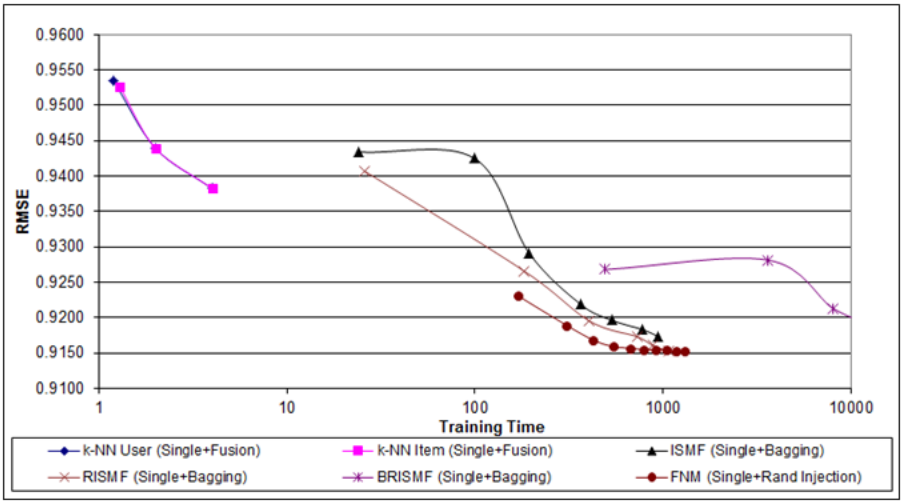


Fig. 4. Computational cost VS. RMSE

6.5 Additional Accuracy Results

We now present results for the larger MovieLens dataset with 1 million ratings. Due to time limitations it was not feasible to test all methods. Therefore, for each of base CF models, we evaluated only the two ensemble models which produced the best results on the MovieLens 100K dataset. Table 2 summarizes the RMSE results of the experiments with the same notations as in the previous table. We applied the same configurations as described in sub-section 6.2 except for the maximum ensemble size that was set to 30 in the Bagging experiments with MF algorithms. The accuracy results in this experiment are consistent with the ones in previous sections.

Table 2. ML (1M ratings) Accuracy Results (RMSE)

BaseCF	Ensemble Model	RMSE-MLB (Ensemble Size)		BaseCF	Ensemble Model	RMSE-MLB (Ensemble Size)
KNN-User	*Base (None)*	0.9302		*RISMF*	*Base (None)*	0.8712
	Fusion	**0.8972 (4)**			*Bagging*	***0.8480 (30)***
	AdaBoost.RT	0.9116 (10)			*Random*	0.8673 (10)
KNN-Item	*Base (None)*	0.9029		*BRISMF*	*Base (None)*	0.8620
	Fusion	0.8972 (4)			*Bagging*	**0.8519 (30)**
	Random	**0.8954 (10)**			*Random*	0.8570 (10)
ISMF	*Base (None)*	0.8812		*FNM*	*Base (None)*	0.8654
	Bagging	***0.8523 (30)***			*Fusion*	0.8469 (10)
	Random	0.8759 (10)			*Random*	**0.8465 (10)**

7 CONCLUSIONS

In this work we presented a novel systematic framework for applying ensemble methods to collaborative filtering models. Our framework used four popular ensemble techniques (Bagging, Boosting, Fusion and Randomness Injection) which were adapted to solve the collaborative filtering based rating prediction task. Typical collaborative filtering algorithms neither handle datasets with reoccurring samples, nor weighted samples. We thus modify the original base collaborative filtering algorithms to handle such settings.

Empirical evaluation shows an RMSE improvement by applying the suggested ensemble methods to the base CF algorithms. These improvements may increase the accuracy of relatively weak models to the level of more advanced ones. We found that in most cases it is preferable to add more base models to the ensemble, as we obtain a more accurate model compared to the combined model. Since all our ensemble methods have a linear running time and space complexity with respect to the ensemble size, it may be feasible to add more models to the ensemble as long as the improvement level is significant. These encouraging results indicate that ensemble methods can be used to enhance collaborative filtering algorithms. The boosting approach suggested in this paper is preliminary and requires further research. In the future we plan to evaluate our suggestions on other datasets and also on other problems, such as the recommendation of items.

A key issue that needs further investigation is how to find a data-driven criterion for choosing the optimal (ensemble, base model) pair for a given dataset. Other issues that need to be addressed are: application of other boosting/ensemble methods, and evaluation of additional collaborative filtering models.

8 REFRENCES

1. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. "Analysis of Recommendation Algorithms for ECommerce". Proc. of the 2nd ACM conference on E-commerce, 158 – 167, 2000.

2. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. "Item-based collaborative filtering recommendation algorithm". International World Wide Web Conference Proceedings of the 10th international conference on World Wide Web,pp. 285–295, 2001.
3. Y. Koren. "Factorization Meets the Neighborhood: a Multifaceted Factor in the Neighbors: Scalable and Accurate Collaborative Filtering". Proc. 14th ACM Int. Conference on Knowledge Discovery and Data Mining (KDD'08), ACM press, 2008.
4. BURKE, R. "Hybrid recommender systems: Survey and Experiments. User Model. User-Adapt. Interact. 12, 4, 331–370, 2002.
5. Takacs, I Pilaszy, B. Nemeth, D. Tikk. "Scalable Collaborative Filtering Approaches for Large Recommender Systems". JMLR 10, pages 623-656, 2009.
6. R. Bell and Y. Koren, "Lessons from the Netflix Prize Challenge", SIGKDD Explorations 9, 2007.
7. R. Bell and Y. Koren, "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights", IEEE International Conference on Data Mining (ICDM'07), pp. 43–52, 2007.
8. L. Candillier, F. Meyer, F. Fessant. "Designing Specific Weighted Similarity Measures to Improve Collaborative Filtering Systems". IEEE International Conference on Data Mining, 2008.
9. X. Su, T.M Khoshgoftaar, X. Zhu, and R. Greiner. "Imputation-boosted collaborative filtering using machine learning classifiers". Symposium on Applied Computing, Proceedings of the 2008 ACM symposium on Applied computing, 2008.
10. M. Wu "Collaborative Filtering via Ensembles of Matrix Factorizations". Proceedings of KDD Cup and Workshop, 2007.
11. J.S Lee, J.-S., & S. Olafsson. "Two-way cooperative prediction for collaborative filtering recommendations". Expert Systems with Applications, 2008.
12. A. Schclar, A. Meisels, A. Gershman, L. Rokach, A. Tsikinovsky. "Ensemble Methods for Improving the Performance of Neighborhood-based Collaborative Filtering". Proceedings of ACM RecSys 2009.
13. Y. Freund & R.E Schapire. "Experiments with a New Boosting Algorithm". Machine Learning: Proceedings of the Thirteenth International Conference, 1996.
14. M. Jahrer, A. Töscher, R. Legenstein, "Combining predictions for accurate recommender systems", Proc. 16th ACM SIGKDD, pp. 693-702, 2010.
15. L. Breiman, "Bagging Predictors". Machine Learning 24, pp. 123-140, 1996.
16. D. Shrestha and D. Solomatine, "Experiments with AdaBoost.RT, an improved boosting scheme for regression", Neural computation, Vol. 18, 2006.
17. K.J. Cherkauer. "Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks", inProc. AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms, Portland, OR, AAAI Press, Menlo Park, CA, pp.15-21, 1996.
18. Friedman, Jerome H. "Stochastic gradient boosting." Computational Statistics & Data Analysis 38.4 (2002): 367-378.