Hybrid Weighting Schemes For Collaborative Filtering

Afshin Moin, Claudia-Lavinia Ignat

Hybrid Weighting Schemes For Collaborative Filtering

Afshin Moin and Claudia Ignat

Inria de Nancy Grand Est

Abstract. Neighborhood based algorithms are one of the most common approaches to Collaborative Filtering (CF). The core element of these algorithms is similarity computation between items or users. It is reasonable to assume that some ratings of a user bear more information than others. Weighting the ratings proportional to their importance is known as feature weighting. Nevertheless in practice, none of the existing weighting schemes results in significant improvement to the quality of recommendations. In this paper, we suggest a new weighting scheme based on Matrix Factorization (MF). In our scheme, the importance of each rating is estimated by comparing the coordinates of users (items) taken from a latent feature space computed through Matrix Factorization (MF). Moreover, we review the effect of a large number of weighting schemes on item based and user based algorithms. The effect of various influential parameters is studied running extensive simulations on two versions of the Movielens dataset. We will show that, unlike the existing weighting schemes, ours can improve the performance of CF algorithms. Furthermore, their cascading capitalizes on each other's improvement.

1 Introduction

Recommender systems (RS) have become increasingly popular in online shops, multimedia and social networks. They are proved to be important tools for helping users overcome the problem of product overload. RS find content personalized to the taste and needs of every individual. They also provide valuable data for service providers, which is in turn analyzed to improve the quality of service. Collaborative Filtering (CF) [7, 9, 4, 14, 15, 10, 8] is the most common strategy for designing recommender systems. CF is based on the assumption that users like items being appreciated by other taste wise users. CF shows more satisfactory results than other alternative strategies like the content based approach [11].

Two prevalent techniques for CF are the neighborhood based approach and Matrix Factorization (MF). The neighborhood based approach computes the similarity between users or items. In turn, it can be user based or item based. The user based approach detects a neighborhood of similar users for each user. On the contrary, the item based approach forms a neighborhood of similar items around each item. Once the neighborhood is formed, its ratings are used to compute the recommendations. Matrix factorization leverages the whole ratings at the same time instead of relying on local similarity based neighborhoods. More specifically, it computes a low rank estimation of the rating matrix which serves to assign coordinates to users and items in a space whose dimensionality is much smaller than the number of users or items. The position of users and items in the space is consequently used to compute the recommendations.

The similarity computation between users or items is the central phase of neighborhood based algorithms. Pearson correlation, cosine and adjusted cosine are the most common measures used for similarity computation in the literature. In their original version, every common rating between two users/items contributes equally to the resulting similarity value. This can decrease the exactitude of the result because some ratings may reveal more information than others about a user's taste. For example in the context of movie recommendation, famous movies receive high ratings from the majority of users. As a result, knowing that two users have rated them similarly does not provide enough evidence for judging them as tastewise.

To overcome this problem, it is possible to escalate or attenuate the weight of each rating proportionally to its estimated importance. However, due to the high level of uncertainty in CF and numerous parameters that may influence the score a user gives to an item, it is very difficult to find a weighting scheme that can correctly estimate which ratings are more important than others. In fact, most suggested approaches degrade the precision of the recommendations. In this paper, we suggest a weighting scheme based on MF computing the weights in function of the item (user) for which the recommendation is computed. This scheme infers the weights by comparing the distance of the corresponding item to the target item in an embedding latent feature space computed through MF. The effect of this scheme is studied besides a large number of other weighting schemes. Although it increases the complexity, its performance is far than many of the existing approaches. Moreover it is merged with other proper schemes, the value of the improvement becomes significant. To compare the effect of different feature weighting methods, we run extensive experiments on two versions of the Movielens dataset [3]. This comparative study is of importance on its own as it compares the effect of many weighting schemes in terms of different parameters (ex. size of the neighborhood) on the big and more reliable datasets which are now publicly available.

2 Background

Assume a system with M users and N items, that is, $u \in \{1, 2, ..., M\}$ and $i \in \{1, 2, ..., N\}$, where the rating of user u for item i is denoted by r_{ui}. The set of ratings can be represented by a *rating matrix* $R \in \mathbb{R}^{M \times N}$, where the entroy corresponding to the uth row and the ith column contains r_{ui} provided u has rated i and is missing otherwise. Each row of the matrix R_{u*} is called the rating vector of user u and each column R_{*i} the rating vector of item i. The goal of the recommender algorithm is to predict the missing entries of the rating matrix. The predicted rating of user u for item i is denoted by $\hat{r}_{ui}$. To test the performance of an algorithm, the dataset is split into two parts: training set and test set. Predictions are made using the ratings in the training set, and results are compared against the real ratings in the test set. We compare precision of different algorithms using Root Mean Square Error defined as $\mathrm{RMSE} = \sqrt{\frac{\sum_{r_{ui} \in T} (\hat{r}_{ui} - r_{ui})^2}{|T|}}$, where T denotes the set of test ratings.

2.1 Neighborhood-based Recommender Algorithm

The neighborhood-based approach can be either user-based or item-based. The user-based algorithm [4] uses the *rows* of the rating matrix to compute the similarity between *users*. Pearson correlation is the most common similarity measure between two users. It is defined as:

$$Corr_{uv} = \frac{\sum_{i \in I_u \cap I_v}(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v}(r_{ui} - \bar{r}_u)^2}\sqrt{\sum_{i \in I_u \cap I_v}(r_{vi} - \bar{r}_v)^2}}$$

To account for the number of items both users have rated, correlation is often multiplied by η_{uv} known as *significance weighting* [4]. Its role is to avoid identifying two users as similar due to a very small number of common ratings. In this paper, we use a logarithmic function for significance weighting: $\eta_{uv} = \log(|I_u \cap I_v| + 1)$. The similarity between two users u and v is then:

$$s_{uv} = \eta_{uv} Corr_{uv}. \tag{1}$$

Once similarities are computed, the ratings of the most similar users to u are passed to an aggregation function to compute her missing ratings:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N(u,i)} s_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v \in N(u,i)} s_{uv}},$$

where $N(u, i)$ is the set of the k most similar users to u having rated i.

The item-based algorithm relies on the columns of the ratings matrix to compute the similarity between *items*. Based on our experiments, *adjusted cosine* leads to better results than Pearson correlation in the item-based approach. It is defined as:

$$ACos_{ij} = \frac{\sum_{u \in U_i \cap U_j}(r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \in U_i \cap U_j}(r_{ui} - \bar{r}_u)^2}\sqrt{\sum_{u \in U_i \cap U_j}(r_{ui} - \bar{r}_u)^2}}$$

The similarity between i and j is denoted by:

$$z_{ij} = \eta_{ij} ACos_{ij}. \tag{2}$$

The predictions are then made as:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{j \in N(i,u)} z_{ij}(r_{uj} - \bar{r}_u)}{\sum_{j \in N(i,u)} z_{ij}}$$

$N(i, u)$ is the set of k most similar items to i that u has previously rated.

2.2 Feature Weighting Schemes for Collaborative Filtering

To discriminate the contribution of ratings to the computation of similarity according to their importance, each rating can be weighted proportional to some weight. Specifically, the Weighted Pearson Correlation is defined as:

$$WCorr_{uv} = \frac{\sum_{i \in I_u \cap I_v} \omega_i(r_{ui} - \bar{r}_u) \cdot \omega_i(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} \omega_i^2(r_{ui} - \bar{r}_u)^2}\sqrt{\sum_{i \in I_u \cap I_v} \omega_i^2(r_{vi} - \bar{r}_v)^2}}$$

The weighted version of the item based approach can be obtained in a similar way. The only nuance is that ω_u should be applied to $ACos_{ij}$ instead of ω_i. The rest of the computations remain similar to the unweighted approaches.

Few works have previously suggested a number of weighting schemes for CF. These techniques are mainly borrowed from information retrieval and rely on statistical properties of data. In [2], Inverse User Frequency (IUF) is applied to the user-based approach, that is, $\omega_i = \log \frac{|U_i|}{|U|}$. The idea is that popular items are rated rather by every one. Therefore, they bear less information. Another technique suggested in [4] is to weight ratings by their variance, i.e. $\omega_i = \frac{var_i - var_{min}}{var_{max}}$, where $var_i = \frac{\sum_{u \in U_i} (r_{ui} - \bar{r}_i)^2}{|U_i|}$. An item with a high variance *may* indicate users have different taste about that. Nevertheless, it is not necessarily true as some items may be difficult for users to rate. [19] uses the Entropy of the items as the weighting scheme. The Entropy of item i is defined as $H(i) = -\sum_{k \in K} p(r_i = k) \log p(r_i = k)$, where K is the range of ratings. In the same work, *mutual entropy* between each item and the item for which the recommendation is computed has also been used as weighting scheme. The mutual entropy between i and j is defined as $I(i,j) = H(i) + H(j) - H(i,j)$, where $H(i,j)$ is the joint entrpy of i and j. This type of weighting increases the complexity as the similarity between two users (items) must be recalculated for every item (user). Based on the results reported in the above works, none of the mentioned schemes can lead to significant improvement of the precision of the recommendations. Most of them even considerably degrade the performance of the algorithm.

3 Hybrid Weighting Schemes

Weighting schemes like IUF, variance and entropy are computed using the *global* statistical patterns of the dataset. They all have in common that their value is always the same for two users and does not depend on the item for which the recommendation is computed. This shortcoming limits their performance because two users may have the same opinion about one type of item but disagree about another type. For example, assume that John and Andy watch comedy and drama movies. They both like all types of dramas and rate high a good movie of this genre. Therefore, their rating behavior is similar for dramas. However, they do not show similar rating behavior about comedies. John likes black comedy (comedy about disturbing subjects like war) but dislikes blue comedies (comedy based on sexism, racism and homophobic view); Andy dislikes the former and likes the latter. In this context, both dramas and comedies are found in the profiles of the two users. However, they are similar w.r.t. dramas, but dissimilar w.r.t. comedies. If portions of their profiles corresponding to these two parts are equal, Pearson similarity between them will be zero as their similarity on dramas neutralizes their dissimilarity on comedies. This problem can be alleviated by computing the similarity between two users in function of the item for which the recommendation is computed. In this case the weighting term and the similarity are function of the active item a, that is, ω_{ia} and $s_{uv}(a)$ replace ω_i and s_{uv} in the respective equations. Indeed, ω_{ia} indicate how much the ratings for item i are informative *for computing the ratings of item* a; An item may be found helpful for predicting the ratings of some items while irrelevant for predictions of some others. In the above example, if the rating to be computed is

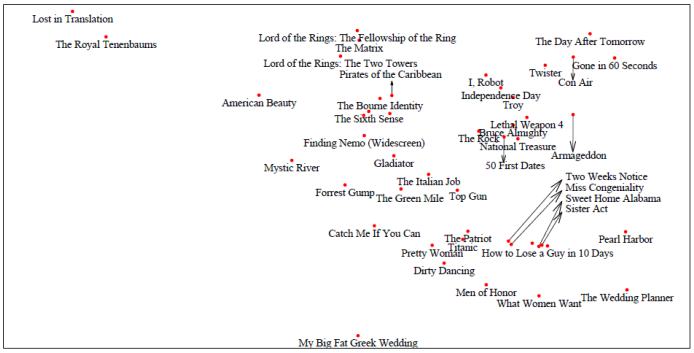


Fig. 1. The top rated movies of Movielens embedded in a latent feature space.

for a drama, the similarity is positive while if it is for a comedy, the outcome would be negative. As will be seen later, ω_{ia} leads to more enhancement than ω_i. It is expectable because ω_{ia} considers more elegant dependencies between ratings. The side effect is that it increases the complexity. M^2N similarities must be computed instead of M^2. Provided item clustering techniques are applied, the complexity can be decreased to M^2C, where C is the number of item clusters. Th weighting schemes that we present in the sequel are from the second type. The other weighting scheme of this type previously suggested in the literature is mutual entropy [19]. We only present the weighting schemes for the user based approach. Their application to the item-based approach is pretty similar: ω_i is replaced by ω_u, and ω_{ia} by ω_{ua}, where a would be the active *user* in the item based approach.

3.1 A Weighting Scheme Through Matrix Factorization

A way to estimate the relevance between two items is to rely on MF. Matrix factorization projects both users and items in a *latent feature space* by factorizing the ratings matrix into two low rank matrices containing user and item coordinates. The factorization is usually done through Singular Value Decomposition (SVD). To compute SVD, all elements of the rating matrix must be known. This is not the case in CF as the majority of the ratings are missing. To bypass this problem, it is a common practice to compute the latent features by minimizing a cost function. The minimization is regularized to avoid overfitting the existing ratings.

$$min \sum_{p*,q*,r*\in R} (r_{ui} - p_u \cdot q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2),$$

where $p_u = (x_u(1), \cdots, x_u(l))$ and $q_i = (y_i(1), \cdots, y_i(l))$ are the latent features of user u and item i in an embedding space of l dimensions. More information can be found in [9, 7] about matrix factorization in CF.

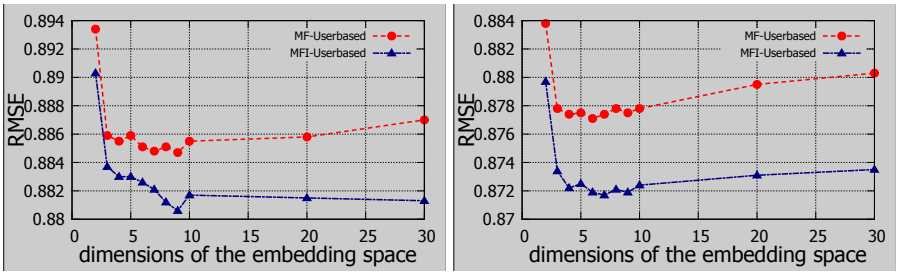
Each neighborhood based approach and MF have their advantages and drawbacks. The former is easy to understand and its recommendations are better explainable to the

users. On the other hand, matrix factorization is less complex and often slightly more precise. Though, explaining the origin of the recommendations to the users is more challenging than in the neighborhood based approach. Namely, it is difficult to explain to the users how the coordinates have been computed and what the logic behind them is. Meanwhile, the superiority of one method over the other, even in terms of one criterion like precision, depends on the underlying dataset. No approach can outperform the other in all scenarios. While MF is a successful recommender algorithm on its own thanks to its multiple advantages, our motivation for its using is pretty different. MF clusters similar items and users close to each other in the embedding space. This property has been noted in passing in some works [9]. Some others like [6] go further and use this property for visualization purposes. Figure 1 shows movies with the largest number of ratings in the Movielens dataset. With a little knowledge about the movies we can observe continuous consistency of movie genre on the map. For example, the two episodes of *Lord of The Rings* are close to each other. In the same way, very similar artsy movies *Lost in Translation* and *Royal Tenenbaums* are on the top left side while *The day after tomorrow* and *Twister* on top right side of the map are both about natural disasters. Specifically, movies on the top left of the map are mostly artsy movies, while those on the top right are actions and adventures. These two groups smoothly join each other in the center just before meeting drama, romance and comedy movies on the bottom. In this paper, we take advantage of the clustering property of MF to estimate the relevance between two items (users). We weight each rating with the inverse Euclidean distance of the corresponding item from the active item:

$$\omega_{ia} = \frac{1}{\sqrt{\|q_i - q_a\|}} \tag{3}$$

Therefore, ratings of items with very different content similarity from the active item have less weight in the computation of user-user similarities. In the experiments, we represent this approach by MF-Userbased and MF-Itembased for the user based and item based approach respectively. The performance of MF weighting scheme depends on the precision of the applied MF approach. The performance of these approaches improves with the number of dimensions of the embedding space up to a threshold where data starts to overfit in the space. Figure 2 shows how the performance of the weighted user based scheme changes with respect to the number of dimensions of the latent feature space.

We also examined a number of embedding algorithms other than the presented SVD-like approach. In particular, we adopted the Euclidean embedding algorithm presented in [6]. This algorithm projects users and items in the embedding space such that the predictions exclusively depend on the Euclidean distance between users and items. We also tried a more exact version of the presented SVD-like algorithm where some base line predictors are added to improve the precision, i.e. $\hat{r}_{ui} = \mu + b_u + b_i + p_u q_i$. Irrespective of the type of the embedding algorithm, items with similar content are put close to each other in terms of *Euclidean distance*. They all lead to almost the same amount of improvement to the precision of the weighted user based algorithm.



(a) neighborhood size (k) = 50 (b) neighborhood size (k) = 20

Fig. 2. Effect of the number of dimensions of the embedding space on RMSE of MF type weighting schemes for Movielens1m.

3.2 Item Weighting with Correlation

Another way of estimating ω_{ia} is to use the correlation s_{ia} between each item and the active item. The rational behind using correlation for feature weighting is slightly different from its direct application to similarity estimation. We use correlation to understand how much the ratings of an item are statistically dependent on the other. Larger values of correlation show that one rating can be estimated with more certainty from the other. Therefore, only the absolute value of similarity (dissimilarity) is important, but not its sign. Note that we use the correlation s_{ij} given in Equation (1) between two items and not adjusted cosine z_{ij} in Equation (2). It is shown in previous works [19] that in CF, continuous weighting has better performance than feature selection (omitting some features). In other words, no user feedback should be totally ignored. Sometimes the correlation between i and a is zero, the corresponding rating will be ignored. To avoid this issue, we sum the correlation with a constant. Hence, the item correlation weighting is defined as:

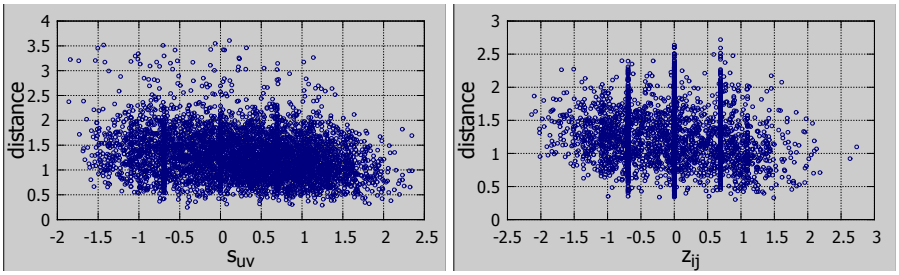
$$\omega_{ia} = (0.5 + |Corr_{ij}|)(1 + \eta_{ia}). \tag{4}$$

A rather similar approach has been applied in [1] without using any significance weighting. Based on our experiments however, $1 + \eta_{ia}$ is necessary to obtain satisfactory results. Unlike traditional user based or item based approach where each use either rows or the columns of the rating matrix, the hybrid approach uses both rows and columns of the rating matrix. Due to this optimized use of data, it can compute more exact recommendations. This approach is denoted by I-Userbased (Item weighting user based approach) and U-Itembased (User weighting item based approach) in the experiments.

3.3 Hybrid Weighting Schemes

Although correlation weighting and MF weighting both estimate the relevance of an item to the active item, they seek this same goal in very different scales. The former captures *local* similarities between items while the latter clusters items and users by compromising their positions is a latent feature space such that a *global* cost is minimized. Investigating the statistical similarity between users (items) and their distance in

the embedding space reveals that these two values are almost uncorrelated. Figure 3a shows s_{uv} versus the distance between u and v in an SVD-like embedding space. Figure 3b shows z_{ij} against distance between i and j in the embedding space. Both graphs are plot for a randomly chosen subset of 100 users and items taken from the Movielens1m dataset. Figure 3b show Cosine against the Euclidean distance in the same type if embedding. It is seen that there is no correlation between the similarity and the distance, that is, given the distance between two users (items) no conclusion can be drawn about their similarity (and vice versa). This basic difference between the behavior of these two weighting schemes enables them to complement each other. This claim is validated in the next section. In the experiments, we will represent this hybrid scheme as MFI-Userbased and MFU-Itembased. It is indeed a hybrid approach merging the item based and user based approach with matrix factorization.



(a) s_{uv} vs. distance (b) z_{ij} vs. distance

Fig. 3. similarity versus distance in the latent feature space.

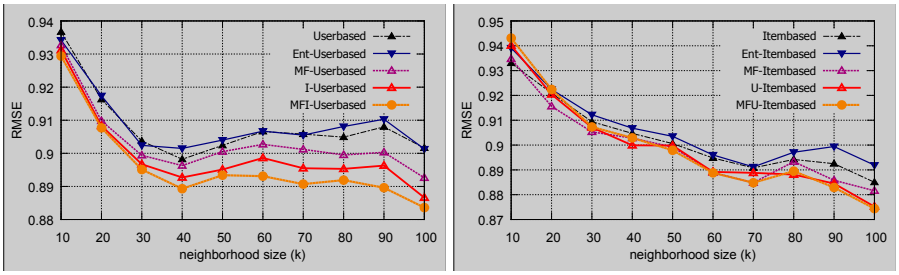
4 Experiments And Results

In this section we report the performance of different weighting schemes on two versions of the Movielens dataset [3]. The properties of these datasets are given in Table 1. The training set contains 95% of the dataset while the remaining 5% form the test set. Each user in the Movielens datasets has at least 20 ratings. We partitioned the profile of each user has at least 1 movie in its test set. IUF, variance and mutual entropy did not enhance the results in any of our experiments. For more clarity, we do not represent them in the figures, but show the most important results in Table 2. A complete report of the results is available in the technical report [anonymous submission].

Entropy was the only low complexity weighing feature that in very few cases lead to slight improvements. We represent the entropy based algorithms with Ent-Userbased and Ent-Itembased. In the same way, variance, IUF and mutual entropy weighting schemes are represented by Var-Userbased, IUF-UserBased and MutEnt-Userbased for the user based, and with Var-Itembased IIF-Itembased and MutEnt-Itembased for the item based approach. By slight engineering of the entropy weighting scheme suggested

Dataset	users	items	ratings	density%
MovieLens100K	943	1682	100000	6.3
MovieLens1M	6040	3883	1000209	4.24

Table 1. Properties of the datasets



(a) Movielens100k-userbased

(b) Movielens100k-itembased

(c) Movielens1m-userbased

(d) Movielens1m-itembased

Fig. 4. RMSE vs. neighborhood size (k)

in [19], we obtained better results. Namely, we add a small constant to avoid excessive ignorance of items whose entropy is very low. Moreover, we observed that using entropy square results in better performance. To prevent high values of entropy due to half stars, we round all the ratings to their ceiling before the computation of entropy. In our experiments we used $\omega_i = (H(i) + 0.2)^2$. Variance and mutual entropy are implemented exactly as in [19] and [4].

Figure 4 shows RMSE in terms of the size of the neighborhood. In MF weighting schemes of Movielens100k, the dimension of the embedding space is 6. In those of Movielens1m, it is set to 7. I-Userbased approach can lead to an improvement slightly superior to that of MF-Userbased. Though, MF weighting scheme is the only one that in no case, even for the item based approach, leads to considerable degradation of the precision. It is then can be used with some level of certainty in all scenarios. Fortunately, their hybrid outperforms each of them individually. This is a good sign that they can improve the performance of each other. It is also interesting that mixing them with entropy, i.e. EntMFI-Userbased, lowers even further the error. Comparing Figure 4a with Figure 4c, it is seen that scheme weighting has more effect on the bigger dataset. Moreover,

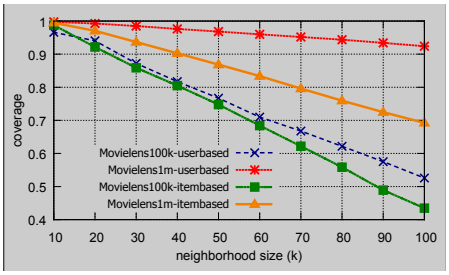


Fig. 5. Coverage of user based and item based schemes.

it leads to more improvement when applied on larger neighborhoods. It is expectable as once a larger number of ratings are properly weighted, the results are supposed to improve further. A recommender algorithm must be able to predict an acceptable fraction of the missing ratings in the dataset. This fraction is known as the *coverage*, and is defined as: $\text{coverage} = \frac{|I_p|}{|T|}$, where $|I_p|$ is the number of missing ratings in the test set for which the algorithm is capable of making a prediction. Figure 5 shows the coverage of user based and item based schemes. Feature weighting does not change the coverage of the algorithm. Table 2 shows the exact RMSE of all weighting schemes for Movielens1m. We choose the size of the neighborhood such that the coverage remains more than 95% (see Figure 5). This corresponds to 20 for the item based and 60 for the user based approach. The precision drops from 0.8829 for the user based approach with no weighting to 0.8673 for EntMFI-Userbased. Considering the limited range of achievable RMSE in collaborative filtering, this is considered as significant gain in the precision.

We also experimented the effect of weighting schemes in different levels of sparsity by changing the ratio of the training set to the test set. Feature weighting was only successful on high levels of density. When the data is sparse, computed weights are prone to a high level of uncertainty. Therefore, they contain more noise than useful data. Moreover, we studied the effect of scheme weighting based on the number of ratings in the user and item profile. Specifically, users and items were each partitioned into three disjoint subsets with small, average and large number of ratings in their profile. Each group of users (items) contained approximately one third of the total ratings. Weighting schemes were experimented for 9 mixtures of user and item groups. For Movielens100k, weighting schemes were unsuccessful when both items and users had few ratings. Apart from this special case, the each weighting scheme had almost the same amount of improvement/degradation for all mixtures of users and items.

5 Related Work

Few works have previously studied the effect of weighting schemes on the neighborhood based CF [16, 12]. Baltrunas and Ricci [1] compare a number of weighting and feature selection methods with each other. Some works like [17, 11] infer the weights

Userbased 0.8829	Ent-Userbased o.8814	IUF-Userbased 0.9022	Var-Userbased 0.8833	MutEnt-Userbased 0.8849
I-Userbased 0.8753	MF-Userbased 0.8768	MFI-Userbased 0.8715	EntMFI-Userbased **0.8673**	
Itembased 0.8879	Ent-Itembased 0.8888	IIF-Itembased 0.8957	Var-Itembased 0.8895	MutEnt-Itembased 0.8867
U-Itembased 0.8962	MF-Itembased 0.8902	MFU-Itembased 0.8986	EntMFU-Itembased 0.8991	

Table 2. RMSE of different weighting schemes for Movielens1m. The neighborhood size is 20 and 60 for the item based and user based approach respectively.

using the items meta data. However, content information is an extra source of information that sometimes companies try to avoid as it costs time and money. An automatic weighting scheme is suggested in [5]. The authors argument that IUF and variance are predefined functions that do not consider the whole structure of a dataset. Hence, it is not sure that their application always lead to improvement of results. They suggest an approach based on maximum likelihood to estimate how significant the ratings of each item are. Their approach computes a global weight for each item by training the algorithm over the whole ratings. Said et al. [13] study the effect the logarithmic IUF and a linear IUF in terms of the number of user ratings. Users are divided into three subsets of cold start, post cold start and power users based on the number of their ratings. The authors show that weighting schemes work better for post cold start users and for cases where the rating scale is compact, say from 1 to 5. [18] reports that continuous weighting methods outperform feature selection in contexts where some features are useful but less important than others. This is indeed the case in CF; Some ratings are more important than others but no user feedback should be totally ignored. In this paper, our study is limited to weighting methods.

6 Conclusion

We suggest feature weighting schemes for improving the precision of neighborhood based collaborative filtering algorithms. We observed that feature weighting based on correlation and matrix factorization are both effective on the user based approach. Moreover, they can improve the performance of each other. Although, they increase the complexity of computations, their mixture achieves considerable improvement over the user based scheme with no weighting. Each type of weighting scheme accounts for different effects in the dataset. Item independent weightings like IUF, variance and entropy leverage the whole ratings of an item to estimate a global weight for it. Although they are less complex than our suggested methods, they are not effective to improve the precision. On the other hand, item dependent weighting schemes, i.e. mutual entropy, correlation weighting and MF weighting, estimate the importance of each rating in function of the item for which the prediction is made. From among these three approaches mutual entropy could not improve the precision of the recommendations for the Movielens dataset. Furthermore, the only scheme that either improves precision

or leaves it almost intact is the MF weighting scheme. We also observed that feature weighting schemes work better for larger datasets and larger sizes of neighborhood.

References

1. L. Baltrunas and F. Ricci. Dynamic item weighting and selection for collaborative filtering. In *Web Mining 2.0 Workshop, ECML-PKDD*, 2007.
2. J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52, 1998.
3. U. o. M. GroupLens. *MovieLens Datasets*, 2013. http://presspot.cs.umn.edu/datasets/movielens/.
4. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR 1999*, pages 230–237, 1999.
5. R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *SIGIR*, pages 337–344, 2004.
6. M. Khoshneshin and W. N. Street. Collaborative filtering via euclidean embedding. In *ACM conference on Recommender systems*, 2010.
7. Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ACM SIGKDD*, pages 426–434, 2008.
8. Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, pages 89–97, 2010.
9. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, pages 30–37, 2009.
10. G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, pages 76–80, 2003.
11. P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Artificial intelligence*, pages 187–192, 2002.
12. D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: a hybrid memory- and model-based approach. In *Uncertainty in artificial intelligence*, pages 473–480, 2000.
13. A. Said, B. J. Jain, and S. Albayrak. Analyzing weighting schemes in collaborative filtering: cold start, post cold start and power users. In *ACM Symposium on Applied Computing*, pages 2035–2040, 2012.
14. R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *International conference on Machine learning*, pages 791–798, 2007.
15. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, pages 285–295, 2001.
16. X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, pages 4:2–4:2, 2009.
17. P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Feature-weighted user model for recommender systems. In *User Modeling*, pages 97–106, 2007.
18. D. Wettschereck, D. W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.
19. K. Yu, X. Xu, M. Ester, and H.-P. Kriegel. Feature weighting and instance selection for collaborative filtering: An information-theoretic approach. *Knowl. Inf. Syst.*, pages 201–224, 2003.