A non-IID Framework for Collaborative Filtering with Restricted Boltzmann Machines

Kostadin Georgiev KGEORGIEV@VMWARE.COM
VMware Bulgaria EOOD, 16A G.M. Dimitrov Blvd, 1797 Sofia, Bulgaria

Preslav Nakov PNAKOV@QF.ORG.QA
Qatar Computing Research Institute, Qatar Foundation, Tornado Tower 10th floor, PO box 5825, Doha, Qatar

Abstract

We propose a framework for collaborative filtering based on Restricted Boltzmann Machines (RBM), which extends previous RBM-based approaches in several important directions. First, while previous RBM research has focused on modeling the correlation between item ratings, we model both user-user and item-item correlations in a unified hybrid non-IID framework. We further use real values in the visible layer as opposed to multinomial variables, thus taking advantage of the natural order between user-item ratings. Finally, we explore the potential of combining the original training data with data generated by the RBM-based model itself in a bootstrapping fashion. The evaluation on two MovieLens datasets (with 100K and 1M user-item ratings, respectively), shows that our RBM model rivals the best previously-proposed approaches.

1. Introduction

In recent years, we have seen steady growth in the number of products and services that have been made available to consumers online. While this growth has created many opportunities for potential customers, it has also made it practically impossible for someone to manually examine the full list of online offerings.

The scale at which products and services are offered has severely limited the utility of human-based solutions such as asking a friend for advice or getting expert opinion from a review or a discussion forum.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

Collaborative Filtering (CF) offers an automated solution by making recommendations based on scores assigned by users with similar interests. The assumption is that if two users have rated a set of items in a similar way in the past, it would be reasonable to expect them to rate new items likewise as well.

Collaborative Filtering works with user ratings that are typically represented as an $N \times M$ integer *user-item matrix*, where each row contains the ratings of one user on M items, while each column represents the ratings for a single item by the N users. In practice, most of the entries in this user-item matrix are zeroes, which indicates the absence of a rating. Hence, CF systems usually have to deal with very large and highly sparse datasets, which is a considerable challenge. It makes it harder for the system to generate accurate recommendations as generally there are no enough commonly-rated items in order to compare user-user preferences efficiently.

To deal with this problem, CF systems typically apply dimensionality reduction techniques (Sarwar et al., 2000b), so that the rows in the matrix, or the user vectors, are projected in a low-dimensional space. Unlike the original vectors of all ratings by a given user, which are very sparse, the low-dimensional vectors live in a reduced space with increased density, which both alleviates data sparseness and offers an opportunity to discover important latent item-item relationships.

The classic way to implement CF, known as *user-based CF*, models the similarity between the different users, and then derives predictions for the unknown ratings of a given target user based on the ratings given by the users it is most similar to in his/her known ratings.

Alternatively, one can take an item-oriented approach to collaborative filtering: use the same methodology, but apply it on items as opposed to users.

By analogy, this technique is called *item-based CF*, and has been reported to have certain advantages over user-based CF (Sarwar et al., 2001). It models item-item similarities and makes predictions for a given user's unknown ratings based on the ratings for items that are similar to the ones the user has already rated.

Naturally, the prediction quality of user-based algorithms drops when there is no enough information to assess the similarities between the users, while item-based algorithms underperform when the same holds for items. To alleviate these issues, below we will combine user-based and item-based CF.

Overall, the introduction of model-based CF and various dimensionality-reduction techniques has allowed CF systems to handle very large datasets, which is crucial for their practical applicability. One promising approach in this respect is the use of Restricted Boltzmann Machines (RBMs) (Smolensky, 1986; Freund & Haussler, 1994; Hinton, 2002), stochastic neural networks, which learn to infer lower-dimensional representations automatically. It has been experimentally shown that RBM-based models are a very competitive way to implement CF (Salakhutdinov et al., 2007): they can scale to hundreds of millions of user-item ratings, while yielding state-of-the-art results on standard benchmark datasets such as those of Netflix.

Below we extend the original RBM-based framework as applied to CF in several important directions. Unlike previous RBM research, which has modeled the correlation between item ratings only, we model both user-user and item-item correlations in a unified non-IID framework, training a single RBM model to make rating predictions for all users in the dataset. Moreover, we work with real numbers as opposed to categorical variables, which allows us to take the natural order between ratings into account, e.g., we consider predicting a rating of 4 when the actual rating is 5 to be better than predicting a rating of 2. This direct modeling of the rating values on the visible layer improves the overall performance of the model. Finally, we explore the potential of combining the original training data with data generated by the RBM-based model itself in a bootstrapping fashion. The evaluation of our extended RBM models on two MovieLens datasets, of 100K and 1M user-item ratings, respectively, shows that they rival the best previously-proposed approaches.

The remainder of the paper is organized as follows: Section 2 presents the related work, Section 3 introduces the general RBM-based framework and our extensions, Section 4 describes our experiments, Section 5 discusses the results, and Section 6 concludes with directions for future work.

2. Related Work

An early extensive comparison of user-based and item-based CF algorithms is presented in (Sarwar et al., 2001), where the item-based algorithms were reported to work better. It was also noted that the item neighborhood, i.e., the set of items that are similar to each other, remains fairly static as new ratings are included, which allows to precompute the item-item similarities, and thus achieve better performance in terms of test-time speed as compared to user-based algorithms.

Successful CF algorithms often involve low-rank approximations of the user-item matrix, e.g., using singular value decomposition (Billsus & Pazzani, 1998; Sarwar et al., 2000a; 2002), or principal component analysis (Goldberg et al., 2001; Kim & Yum, 2005; Kozma et al., 2009). Related model-based approaches include statistical latent models (Hofmann, 2004), uncertain graphs (Taranto et al., 2012) and autoassociative networks (Vozalis et al., 2010).

Recently, matrix factorization techniques that model both users and items in a joint latent factor space have gained popularity for CF (Salakhutdinov & Mnih, 2008; Koren et al., 2009; Lawrence & Urtasun, 2009). A similar probabilistic latent model that represents users and items simultaneously is described in (Langseth & Nielsen, 2012); our model is related, but simpler and based on RBM.

RBMs were first applied to CF in (Salakhutdinov et al., 2007), where a separate RBM was trained for each user. All RBMs had the same number of binary hidden units, but each RBM had visible softmax units for the items rated by the target user only. However, all weights and biases were shared among the RBMs, so that if two users had rated the same item, their RBMs would use the same weights between the visible unit for that item and the hidden units. The user ratings were modeled using multinomial random variables. In contrast, our non-IID framework models both user-user and item-item interactions in a single RBM; we further use real-value visible units, which capture the natural order between the ratings.

Finally, Truyen et al. (2009) explored joint modeling of users and items for CF, but inside an *unrestricted* version of Boltzmann Machines (BMs). Their network topology, visible units and rating prediction schemes are quite different from ours and rely on more complex parametrization and preprocessing, such as correlation computation and neighborhood formation to determine the connectivity at their visible layer. Moreover, BMs are not as scalable as RBMs, which makes our model preferable: it yields comparable quality.

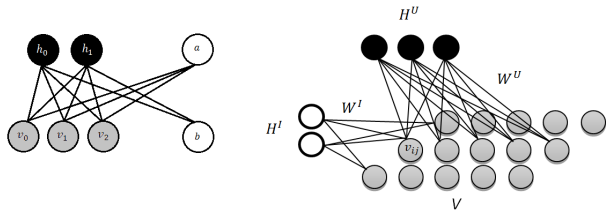

Figure 1. The user-based RBM model (left) vs. our user-item-based RBM (right). Each visible unit represents a numerical user rating. In the left model it is connected to one hidden layer, while in the right it is linked to two independent hidden layers: one modeling correlations between items and another one modeling correlations between users.

3. Method

Below we introduce the general RBM-based framework and our extensions and modifications thereof.

3.1. RBM for CF

An RBM uses an $N \times M$ *user-item matrix* with integer ratings between 1 and K, and 0 when a rating is missing. It has a linear visible layer V consisting of M real-valued units $\{v_0, v_1, \ldots, v_M\}$, which is fully connected to a binary hidden layer H consisting of F hidden units $\{h_0, h_1, \ldots, h_F\}$. All connection weights are symmetric ($w_{ij} = w_{ji}$) and there are no connections within a layer. There are also some special units: a bias a_i for the visible unit v_i, and a bias b_j for the hidden unit h_j. See the left of Figure 1 for an example.

The probabilistic activation function of the hidden layer conditioned on the visible layer is given by the following equation: $P(h_j = 1|v) = L(b_j + \sum_{i=1}^{M} w_{ij} v_i)$, where $L(x)$ is the logistic function. Traditionally, RBMs have modeled real-valued data using Gaussian visible units (Hinton & Salakhutdinov, 2006): $P(v_i|h) = \mathcal{N}(a_i + \sum_{j=1}^{F} w_{ij} h_j, \sigma_i^2)$. However, in our experiments, we achieved better prediction quality when using noise-free reconstructions, i.e., when the value of a visible unit is equal to the sum of its total input from the hidden units plus its bias.

We treat each user as a single training case for the same RBM, which learns to model the joint distribution of all user vectors of ratings. Each of the hidden units models the presence or absence of a particular feature for a given user. The symmetric connections between a hidden unit and all visible units, i.e., the user's ratings, learn to jointly model the correlations between the ratings assigned to the different items. The RBM model learns to represent each user with a set of F binary features, and for each configuration of features, it can generate ratings for all M items.

Naturally, we can neither expect that all N users have rated all M items nor that they have rated the same subset of items. Furthermore, most of the M ratings by a given user will be missing. In order to allow our RBM to make predictions for missing ratings, we make a simple change to the learning procedure.

In general, during the learning process of an RBM, it is trained to approximate the observed data distribution with the distribution that is eventually produced by the machine when it reaches the state of thermal equilibrium. So, normally if some rating is missing for a given user, i.e., is set to 0, the RBM will always strive to reconstruct the same value for it, i.e., 0; this is not what we want. We deal with this problem by changing the weight update function to ignore any terms that are connected to visible units that were originally set to 0. Of course, these visible units are ignored from all computations in both the positive and negative phases. For the rest, we update the weights as follows:

$$\Delta w_{ij} = \varepsilon(< v_i h_j >^+ - < v_i h_j >^-), v_i^0 > 0 \quad (1)$$

$$\Delta w_{ij} = 0, v_i^0 = 0 \quad (2)$$

This setting is logically equivalent to the multiple RBMs with shared connections described in (Salakhutdinov et al., 2007), but it serves as a better foundation for the introduction of our hybrid RBM model. In the above formulas, v_i^0 denotes the original value of v_i, i.e., the rating for item i as given in the user-item matrix; Δw_{ij} is the update for the weight between the visible unit v_i and the hidden unit h_j; ε is the learning rate[1]; $< v_i h_j >^+$ denotes the average of the product $v_i h_j$ sampled from the positive phase, when a data vector from the training set is clamped on the visible units; $< v_i h_j >^-$ denotes the average of the product $v_i h_j$ sampled from the negative phase, when the network runs freely. Note that we can avoid computing the latter by following the gradient of the *Contrastive Divergence* objective function instead of the log-likelihood, as described in (Salakhutdinov et al., 2007).

To make predictions after the RBM model has been trained in this way, we simply clamp a given user's ratings on the visible layer. Then, we let the features in the hidden layer be activated via the weights of their connections to the visible layer. Finally, we compute the new values of the visible layer given the state of the hidden layer and the weights of the connections to it. The new values of the visible layer are the actual rating predictions. The values could simply be rounded if only integer values are allowed for the ratings.

[1] In the interest of clarity of presentation, we have intentionally omitted some other factors from the equation such as the weight decay and momentum.

3.2. Hybrid RBM for CF

Given that the above-described RBM model represents users on both its visible and its hidden layers, it could be classified as a user-based CF model. However, we can easily design an alternative item-based version of it. The only difference would be that now the RBM will have a linear visible layer V consisting of N real-valued nodes $\{v_0, v_1, \ldots, v_N\}$, where v_i would accept the rating assigned to a given item by user i.

More importantly, we can combine the two RBM models into a single unified model that takes into account both item-item and user-user correlations. In order to do that, we consider an RBM model where the whole user-item matrix is treated as a single training example. Let v_{ij} be the visible unit that corresponds to the rating of user i on item j. The unit v_{ij} is connected to two independent hidden layers – one user-based and another item-based. Assuming noise-free reconstruction, its value is given by the following equation:

$$v_{ij} = \frac{1}{2}\left[a_i^U + \sum_{p=1}^{F^U} w_{ip}^U h_{ip}^U + a_j^I + \sum_{q=1}^{F^I} w_{jq}^I h_{jq}^I\right] \quad (3)$$

Here the upper index U denotes that the corresponding term is related to the user-based hidden layer, while the upper index I stands for relation to the item-based hidden layer. The size of the user-based hidden layer is F^U, while the size of the item-based hidden layer is F^I. It can be easily seen that in order to compute a single rating v_{ij}, the entire i-th row as well as the entire j-th column has to be available. Hence, in order to compute all ratings for a given user, the entire user-item matrix must be available. This is computationally more intensive than the user/item-based RBM model where only the row/column for the target user/item is needed. Still, it is possible to reduce this complexity by precomputing the sums in the equation above. This model is shown on the right of Figure 1.

This hybrid RBM model could be also represented by two standalone user-based and item-based RBMs, after making some simple changes in their learning schemes. Consider batch learning where the batch equals the entire user-item matrix. Then, after each learning step, an RBM generates a new prediction matrix according to its current internal state. The learning rule changes the weights so that the prediction matrix approximates the original user-item matrix for all ratings that were not missing originally. So, in order to train the hybrid model, we need to change the learning rule, so that when the new weight updates are being computed, we substitute each RBM prediction matrix with the average sum of the two individual RBM prediction matrices.

Hence, the training time of the hybrid model equals roughly the sum of the training times of the two standalone models. This learning procedure corresponds to contrastive divergence for the proposed hybrid RBM model. It is very important that we take the average, as opposed to the direct sum of the two matrices, so that we let each individual RBM, and its feature detectors, model the real rating values as opposed to parts of sums. We have experimentally confirmed the significance of this: on the small MovieLens dataset, using a simple sum yields MAE of just 0.875, while using the average yields MAE of 0.690 (which is the best result we report in this paper).

For comparison, we also try our hybrid user-item approach on RBM with a multinomial visible layer, as described in (Salakhutdinov et al., 2007). In order for this model to be efficient, equation (3) has to be computed as the total input for each softmax unit. Note that each softmax visible unit is connected to each unit of the two hidden layers via symmetric connections.

3.3. RBM-boosted CF

It is possible to boost the final prediction quality by combining the predictions of an RBM model with those of a standard neighborhood-based model. Consider an item-based approach where the weight w_{ij} is computed as a combination of the few available original ratings and the full set of ratings predicted by an RBM model using *Pearson correlation*:

$$w_{ij} = \frac{\sum_{u=1}^{N}(r'_{ui} - \overline{r}_i)(r'_{uj} - \overline{r}_j)}{\sqrt{\sum_{u=1}^{N}(r'_{ui} - \overline{r}_i)^2}\sqrt{\sum_{u=1}^{N}(r'_{uj} - \overline{r}_j)^2}} \quad (4)$$

Here r'_{ui} is the rating of user u for item i as predicted by the RBM model, and $\overline{r}_i$ is the average rating for item i. Unlike in previous research, we compute the averages from the original ratings only, excluding the ratings predicted by the RBM model. The final prediction for the rating of user u on item i can be computed using weighted average:

$$P_{ui} = \overline{r}_i + \frac{\sum_{j=1}^{M}(r_{uj} - \overline{r}_j)w_{ij}}{\sum_{j=1}^{M}|w_{ij}|} \quad (5)$$

An individual user-based or item-based RBM has very good performance in terms of speed. By combining an RBM model with a neighborhood-based model, we can retain much of the speed efficiency since the item averages and item-item weights (correlations) could be precomputed. Moreover, as noted in (Sarwar et al., 2001), an item's neighborhood is fairly static, and thus we can retain only a small portion of the most similar items to be used for the actual ratings prediction step.

4. Experiments and Evaluation

Below, we first describe our test datasets and our evaluation measure; then, we present our evaluation setup and we report our experimental results.

4.1. Datasets

We evaluated the above-described RBM-based models on two *MovieLens* datasets[2], which are commonly used for evaluating collaborative filtering algorithms.

The first dataset (MovieLens 100k) consists of 100,000 ratings for 1,682 movies assigned by 943 users, while the second one (MovieLens 1M) contains one million ratings for 3,952 movies by 6,040 users. Each rating is an integer between 1 (worst) and 5 (best).[3] The ratings are highly sparse: around 93.7% of them are missing from the former and 95.8% from the latter dataset. For both datasets, we follow previous researchers and perform 5-fold cross-validation using the provided five disjoint 80%:20% training:testing data splits.

4.2. Evaluation Measure

For evaluation, we use *Mean Absolute Error* (MAE), which measures the deviation of the predicted ratings from their true values as specified by the users. For each pair of a true user-specified rating u_i and a predicted rating p_i, the absolute difference $|u_i - p_i|$ is calculated between them as the error for that pair. Then MAE is computed as the average error over all L pairs:

$$MAE = \frac{\sum_i^L |u_i - p_i|}{L} \tag{6}$$

4.3. Experimental Setup

We evaluated three RBM-based models: **U-RBM**, a user-based RBM, **I-RBM**, an item-based RBM, and **UI-RBM**, a joint user-item-based RBM.

We further experimented with the above-described neighborhood-based algorithm, implemented to model the similarities between items as opposed to users, and further boosted with rating predictions generated by an item-based RBM model. Below we refer to this hybrid algorithm as **I-RBM+INB**.

We trained all models in batch mode using mean-field update after a full pass through the data. We used Contrastive Divergence learning with one step of Gibbs sampling, which reduces training time considerably.

[2] `http://www.grouplens.org/node/73`

[3] The dataset provides some further information, e.g., demographic, which has typically been ignored in previous research; we did the same in order to make our results directly comparable to those in related work.

We trained all models for several epochs. For the weight updates, we used a learning rate of 0.05/training-size, which is a good default value, where the training size equals the number of users for the user-based RBM or the number of items for the item-based RBM. We also used a momentum of 0.6, again a good default value, and a weight decay with regularization parameter $\lambda = 0.0002$; we did not try to optimize the values of these parameters. We sampled the initial weights from a zero mean normal distribution with a standard deviation of 0.1.

We trained the hybrid RBM model (UI-RBM) by adapting the training procedures of the standalone item-based and user-based RBMs, so that after each pass through the training data and before calculating the new weight updates, each RBM "thinks" that it has generated the average of the values generated by the two RBMs. The final rating predictions are also obtained as an average of those of the two RBMs.

4.4. Results

We report results on both MovieLens 100k and 1M.

4.4.1. MovieLens 100k

We compared the prediction quality of our models for different sizes of the hidden layers and for different numbers of training epochs. The left side of Figure 2 shows the dependency of MAE on the number of units in the hidden layer when the models are trained for 200 epochs. We can see that the user-based model (U-RBM) works best with 50 hidden units, while for the item-based model (I-RBM), it is best to use about 80 hidden units. In contrast, the hybrid, item-based RBM and neighborhood-based model (I-RBM+INB), is not affected much by the number of units in the RBM hidden layer; this is a very attractive property.

Next, we studied the dependency of MAE on the number of epochs in training. The experiments with U-RBM ($F = 50$), I-RBM ($F = 80$), I-RBM+INB ($F = 130$) and UI-RBM ($F^U = 40$, $F^I = 40$) are shown on the right side of Figure 2. We can see that U-RBM achieves optimal performance after 150 epochs and stays relatively stable afterwards; I-RBM performs a bit worse when trained for less than 200 epochs and then slightly outperforms U-RBM. We can further see that UI-RBM significantly outperforms the standalone models U-RBM and I-RBM. It is also better than the I-RBM+INB model, but not by much.

We made an experiment averaging the results of U-RBM and I-RBM, and this also led to significant improvement (MAE = 0.74) over the standalone models.

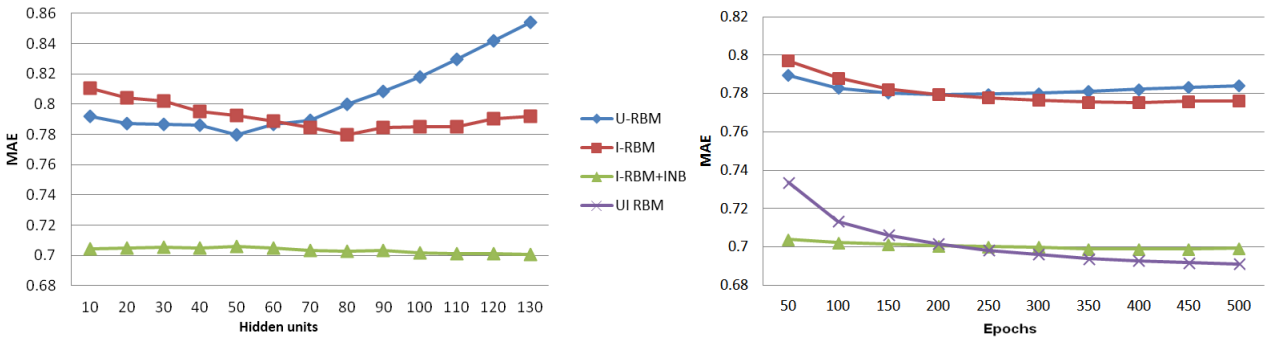

Figure 2. MAE on 100k: user-based RBM (U-RBM), item-based RBM (I-RBM), neighborhood-based boosted with data from I-RBM (I-RBM+INB) and hybrid RBM model (UI-RBM), $F^U = 40$, $F^I = 40$ (right side only). On the left, the x-axis is hidden nodes, for 200 epochs. On the right, the x-axis is epochs, for the optimal hidden size from the left.

We can attribute this to the two models having different kinds of errors. However, as we see on the right side of Figure 2, the unified model UI-RBM yields even better results, which indicates that modeling user-based and item-based correlations is a better idea.

Next, we compared the prediction quality achieved by our models to that of previous work that used the MovieLens 100k dataset. These include not only RBMs, but also collaborative filtering approaches based on singular value decomposition (SVD), principal component analysis (PCA), and nearest neighbors.

Table 1. Comparison (on 100k) of the prediction quality of various CF models and our RBM-based models (in bold).

CF Model	MAE
SVD PCA (Vozalis et al., 2010)	0.793
H-NLPCA (Vozalis et al., 2010)	0.784
U-RBM	**0.779**
I-RBM	**0.775**
SVD (Sarwar et al., 2002)	0.733
Item-based CF (Sarwar et al., 2001)	0.726
Iter PCA + K-means (Kim & Yum, 2005)	0.712
Iter PCA + RRC (Kim & Yum, 2005)	0.700
I-RBM+INB	**0.699**
UI-RBM	**0.690**
Latent CF (Langseth & Nielsen, 2012)	0.685

The results are shown in Table 1. We can see that our two baselines, U-RBM and I-RBM, perform better than two strong models from the literature: the autoencoder network and the SVD- and the PCA-based approaches of Vozalis et al. (2010); the difference in MAE is statistically significant according to the two-tailed Pearson's χ^2 test. Also, the results of our I-RBM+INB model are comparable to the results of the improved but more complex PCA-based models combined with clustering methods (Kim & Yum, 2005).

The improvement of our hybrid UI-RBM over the PCA-based algorithms of Kim & Yum (2005) is also statistically significant. Our final result gets very close to the state-of-the-art result of Langseth & Nielsen (2012): with a difference of only half a point of MAE.

4.4.2. MovieLens 1M

Next, we evaluated the performance of our RBM-based models on the larger MovieLens 1M dataset. We further compared the use of real-valued visible layers vs. multinomial ones, e.g., the latter were used in (Salakhutdinov et al., 2007). The results are summarized on Table 2 along with the best previously-published results on this dataset.

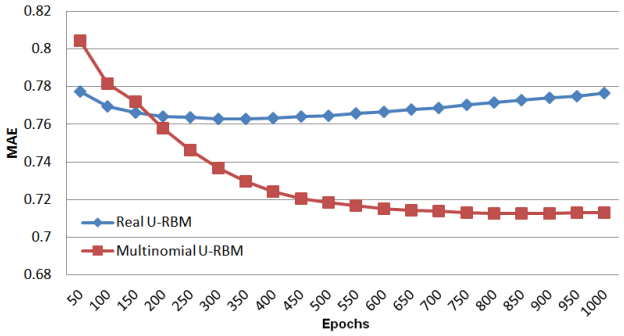

Figure 3. MAE on 1M: a user-based RBM with a real-valued visible layer (Real U-RBM) vs. a user-based RBM with multinomial visible layer (Multinomial U-RBM).

Figure 3 shows the dependency of MAE on the number of epochs for U-RBM when using a real-valued vs. a multinomial visible layer. We used $F = 150$ hidden units for the former and $F = 50$ for the latter. We determined these values experimentally, as we did for the 100k dataset; however, due to the lack of space, this time we skip the detailed results.

Figure 3 shows that the real-valued model initially works better, but then falls behind the multinomial model as the training continues. Hence, while the real-valued model is simpler and has fewer parameters to learn, it is also more sensitive to overfitting and thus it should not be preferred to the multinomial model.

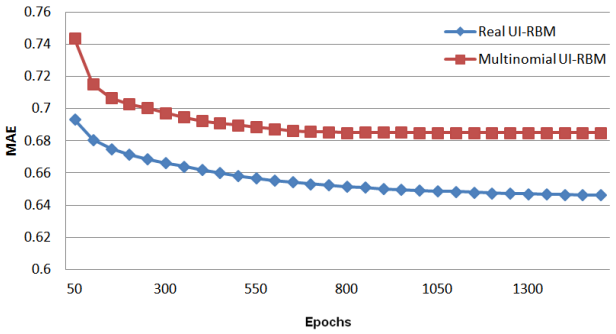

Figure 4. MAE on 1M for two hybrid RBM models: with a real-valued visible layer, $F^U = 70$, $F^I = 70$ and with a multinomial visible layer, $F^U = 50$, $F^I = 50$.

However, Figure 4, which compares the prediction quality of the hybrid model when applied to the real-valued and to the multinomial visible layers, shows that the situation is different when users and items are modeled jointly. Here, the real-valued model yields consistently better results compared to the multinomial one and is ultimately significantly better than it. Therefore, we can conclude that modeling user-item ratings directly using a linear real-valued visible layer is preferable over the multinomial alternative as it yields better predictions while also being simpler topologically and computationally, and while requiring less parameters to learn.

Finally, Table 2 shows a comparison of the prediction quality of all models we experimented with to previously-published results on the MovieLens 1M dataset.[4] Note, that all RBM-based models include RBM in their names, while all BM-based models include BM;[5] also, Multinomial U-RBM denotes our implementation of the categorical RBM model described by (Salakhutdinov et al., 2007). We can see that the results for our real-valued U-RBM and I-RBM are comparable, and the latter is slightly better. The rest of our RBM-based models are better than the best results reported in (Taranto et al., 2012).

[4]The compared methods on the two datasets are different because we compare against published results, and the best results for the two datasets are different.

[5]In order to save space, we abbreviate the original model names given in the literature by substituting USER-ITEM with UI. We further append BM to model names to denote that they are based on an *unrestricted* Boltzmann Machine.

Table 2. Comparison (on 1M) of the prediction quality of various CF models and our RBM-based models (in bold). The values for (Truyen et al., 2009) are approximate and derived from the graphs in their work.

CF Model	MAE
Real U-RBM	**0.762**
Real I-RBM	**0.761**
L8 (Taranto et al., 2012)	0.720
Multinomial U-RBM	0.711
Multinomial I-RBM	0.710
Multinomial UI-RBM	**0.685**
GAUSS-UI-BM (Truyen et al., 2009)	0.675
Real I-RBM+INB	**0.669**
ORD-UI-BM (Truyen et al., 2009)	0.657
Real UI-RBM	**0.645**
ORD-UI-BM-CORR (Truyen et al., 2009)	0.640

Also, both our real-valued hybrid RBM model (Real UI-RBM) and our neighbourhood-based RBM model (I-RBM+INB) perform better than the more complex joint user-item Gaussian BM-based model (GAUSS-UI-BM) described by Truyen et al. (2009). The visible nodes in the latter are connected not only to the hidden layers but also to other nodes in the visible layer. This introduces additional parameters to be learned, which, combined with the different types of parameters used, complicates and slows down learning (this holds for all BM-based models in Table 2), leading to worse model performance as compared to ours.

Our real-valued hybrid RBM model further outperforms the ordinal joint user-item BM-based model (ORD-UI-BM) proposed by Truyen et al. (2009). Finally, we can see in the last row of Table 2 that the additional preprocessing, which includes user-user and item-item correlation computation followed by neighborhood formation for each visible node in the input layer, helps the ORD-UI-BM-CORR model (Truyen et al., 2009) to achieve state-of-the-art performance. Yet, our real-valued hybrid RBM-based model is preferable as it requires no additional preprocessing, has simpler parametrization, and still achieves a very similar performance (worse by 0.005 MAE only).

5. Discussion

There are several important observations we can make. First, the relatively simple RBM models with linear visible nodes, such as U-RBM and I-RBM, can yield prediction quality that outperforms more complex approaches based on autoencoder networks, SVD, PCA, and correlation-based neighborhood formation.

Training our models is speed-efficient, which allows us to handle potentially very large datasets. Moreover, the evaluation on a small and on a ten times larger datasets has demonstrated that the prediction quality remains consistent and improves when the amount of training data increases by an order of magnitude; this is a good indication for potential practical applicability. The user-based RBM achieves its optimal MAE with fewer parameters to learn, $F = 50$, while the item-based one achieves its best results for $F = 80$ on MovieLens 100k dataset. While the latter yields slightly better results than the former, the difference is not statistically significant. Also, the user-based model is more susceptible to overfitting as the number of hidden nodes (user features) grows, which is not the case for the item-based RBM model and especially for its combination with Pearson correlation. The prediction quality of the latter remains stable for different numbers of item features, which means that it can handle a variety of dataset scales without the need to retune its parameters. While averaging the results of both models yields significant improvements, unifying them in a joint RBM model works even better.

We have further shown that using a real-valued visible layer that directly models the user-item ratings is not only more efficient and simpler, but also yields ultimately better prediction quality compared to using a multinomial binary visible layer. This holds for all but the standalone user- or item-based RBM models, where the multinomial model achieves better results after a certain number of training epochs.

Our final results are comparable to the best published results for CF algorithms on the two MovieLens datasets. The training/prediction time of our best model equals the sum of the training/prediction times of two standalone RBM models. However, in order to predict the ratings for a single user, the whole user-item matrix needs to be clamped on the RBM's visible layer. This could be optimized by precomputing the activation signal to each hidden node.

Furthermore, using the predictions generated by the standalone item-based RBM in the computation of the item-item correlations with Pearson correlation yields significant improvements for the neighborhood-based algorithm. The final results are slightly worse than those of the hybrid user-item-based RBM model.

Finally, the performance of the ratings prediction process could be optimized by precomputing the item neighborhoods as noted in (Sarwar et al., 2001). However, computing the correlations using the predictions generated by the hybrid RBM model does not lead to further improvements.

6. Conclusion and Future Work

We have introduced and experimentally evaluated extensions of the original RBM-based framework for collaborative filtering in several important directions. While the original RBM framework only uses user-based modeling, we model both user-based and item-based correlation in a unified framework. In particular, we have proposed a non-IID hybrid RBM model that uses two types of hidden features: the first type models dependencies between the ratings of different items, while the second type models user-user dependencies.

Our evaluation on two MovieLens datasets of different sizes has shown that this model yields results that rival the prediction quality of the best previously-proposed CF algorithms, which are also more complex. We have further proposed a combination of our standalone item-based RBM model with a standard neighborhood-based algorithm that relies on Pearson correlation. We experimentally confirmed that this combination also yields comparable results, while maintaining high speed.

Moreover, we used linear (as opposed to sigmoid) visible nodes that model the numerical rating values and their predictions directly. Unlike the original binary visible nodes, the linear ones allow the model to take into account the natural order between real-valued ratings. This means that when the actual score is 5, our model considers lower penalty when predicting a score of 4 than if the prediction was 2. Also, the direct modeling of the rating numbers leads to improved prediction and overall performance of the model. In order to examine the effects of this simplification on the prediction quality, we also have implemented alternative variants of our models that use multinomial visible layers. The evaluation on both datasets we experimented with shows that using real-valued input nodes yields significant improvement.

In future work, we are considering a number of interesting extensions of the proposed framework. One way to extend the hybrid RBM model is to stack an additional hidden layer on top of the independent item-based and user-based feature detectors. This new layer could potentially model higher-order correlations between users and items. Another possibility for extension is to incorporate various content-based features such as user's demographic information, item's categorizations and others (Pazzani, 1999; Melville et al., 2002). Finally, the prediction quality of the combined RBM and the neighborhood-based approach could be potentially improved by better tuning the sizes of the items neighborhood.

References

Billsus, Daniel and Pazzani, Michael. Learning collaborative information filters. In *Proceedings of ICML*, pp. 46–54, San Francisco, CA, USA, 1998.

Freund, Yoav and Haussler, David. Unsupervised learning of distributions on binary vectors using two layer networks. Technical report, University of California at Santa Cruz, Santa Cruz, CA, USA, 1994.

Goldberg, Ken, Roeder, Theresa, Gupta, Dhruv, and Perkins, Chris. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, 4(2):133–151, 2001.

Hinton, Geoffrey. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14 (8):1771–1800, 2002.

Hinton, Geoffrey and Salakhutdinov, Ruslan. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Hofmann, Thomas. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1): 89–115, 2004.

Kim, Dohyun and Yum, Bong-Jin. Collaborative filtering based on iterative principal component analysis. *Expert Syst. Appl.*, 28(4):823–830, 2005.

Koren, Yehuda, Bell, Robert, and Volinsky, Chris. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Kozma, Laszlo, Raiko, Tapani, and Ilin, Alexander. Binary principal component analysis in the Netflix collaborative filtering task. In *Proceedings of the IEEE Workshop on Machine Learning for Signal Processing*, pp. 1–6, Grenoble, France, 2009.

Langseth, Helge and Nielsen, Thomas Dyhre. A latent model for collaborative filtering. *Int. J. Approx. Reasoning*, 53(4):447–466, 2012.

Lawrence, Neil D. and Urtasun, Raquel. Non-linear matrix factorization with Gaussian processes. In *Proceedings of ICML*, pp. 601–608, Montreal, Quebec, Canada, 2009.

Melville, Prem, Mooney, Raymod, and Nagarajan, Ramadass. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the national conference on Artificial Intelligence*, pp. 187–192, Edmonton, Alberta, Canada, 2002.

Pazzani, Michael. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, 1999.

Salakhutdinov, Ruslan and Mnih, Andriy. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of ICML*, pp. 880–887, Helsinki, Finland, 2008.

Salakhutdinov, Ruslan, Mnih, Andriy, and Hinton, Geoffrey. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of ICML*, pp. 791–798, Corvalis, OR, USA, 2007.

Sarwar, Badrul, Karypis, George, Konstan, Joseph, and Riedl, John. Application of dimensionality reduction in recommender system – a case study. In *Proceedings of the ACM WebKDD workshop*, Boston, MA, USA, 2000a.

Sarwar, Badrul, Karypis, George, Konstan, Joseph, and Riedl, John. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the ACM conference on Electronic commerce*, pp. 158–167, Minneapolis, MN, USA, 2000b.

Sarwar, Badrul, Karypis, George, Konstan, Joseph, and Riedl, John. Item-based collaborative filtering recommendation algorithms. In *Proceedings of WWW*, pp. 285–295, Hong Kong, 2001.

Sarwar, Badrul, Karypis, George, Konstan, Joseph, and Riedl, John. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Proceedings of the International Conference on Computer and Information Science*, pp. 27–28, 2002.

Smolensky, Paul. Information processing in dynamical systems: foundations of harmony theory. In Rumelhart, David and McClelland, James (eds.), *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pp. 194–281. MIT Press, Cambridge, MA, USA, 1986.

Taranto, Claudio, Mauro, Nicola Di, and Esposito, Floriana. Uncertain graphs meet collaborative filtering. In Amati, Giambattista, Carpineto, Claudio, and Semeraro, Giovanni (eds.), *Proceedings of the Italian Information Retrieval Workshop*, pp. 89–100, Bari, Italy, 2012.

Truyen, Tran The, Phung, Dinh, and Venkatesh, Svetha. Ordinal Boltzmann machines for collaborative filtering. In *Proceedings of UAI*, pp. 548–556, Montreal, Quebec, Canada, 2009.

Vozalis, Manolis, Markos, Angelos, and Margaritis, Konstantinos. Collaborative filtering through SVD-based and hierarchical nonlinear PCA. In *Proceedings of ICANN*, pp. 395–400, Thessaloniki, Greece, 2010.