A Neural Autoregressive Framework for Collaborative Filtering

Abstract—**Restricted Boltzmann Machine (RBM) is a two layer undirected graph model that capable to represent complex distributions. Recent research has shown RBM-based approach has comparable performance with, even performs better than previous models on many collaborative filtering (CF) tasks. However, the intractable inference makes the training of RBM sophisticated, which prevents it from practical application. The Neural Autoregressive Distribution Estimator (NADE) is inspired by RBM, but it provides tractable distribution using an autoregressive approach. We describe a novel neural framework for collaborative filtering called NACF, which is extended from NADE, providing comparable performance with previous models but an easier training procedure than RBM. We apply the autoregressive approach of NADE to CF tasks by extending it in many ways. First, we model the user-item ratings with different visible units, among which the linear output units perform best. We propose the dual reversed ordering approach to relieve the data sparsity and bias caused by single random ordering. Further, we show that it is easy to combine with other information to improve performance, which is promising for practical application. Finally, we show the NACF model can be easily trained compared to RBM-based models, and the item-based NACF yields better performance than corresponding RBM on two MovieLens datasets.**

I. Introduction

With the rapid growth of numerous information and services on the Internet, people are feeling troubled when selecting products or services that really comply with their preferences. And more often, even browsing the potential item list should be time consuming. In recent years, recommendation system not only gives people easier access to news, movies, music or other products or services on the Internet, but also brings more and more business opportunities for companies.

One of the most common recommendation problem is user-item rating prediction, it involves $N \times M$ user-item matrix, N and M are the number of users and items respectively. The j-th entry of the i-th row of the matrix represents the rating of item j by user i. And the rating often scales from 1 to 5, and zero means the entry of rating is unknown. The matrix often is very sparse because user only access a small part of the whole item set in real scenarios. The task is about to fill in the empty entries in the matrix, that is, predicting the rating of an unseen item for a user, and we can use the estimated ratings to recommend items to the users.

Collaborative Filtering is a simple but effective approach for this problem. Memory-based models [1] use the user-item ratings to calculate similarity of the users or items, then using the weighted sum of all the similar users or items to make rating predictions. Memory-based methods are effective and easy to implement, but it also has some drawbacks. Because of the sparsity problem there's little common user or item, this makes the calculated similarity becomes unreliable [2]. And it also has limited scalability for large datasets.

While memory-based method often suffered from scalability and efficiency problems, recently factor models, based on the intuition that the ratings are decided by user or item latent factors, has been shown to have good performance and more efficient in implementation. These models use the raw ratings to train a model with many latent factors, these latent factors can be viewed as user or item features. Through the latent factor learned from existing ratings, we can calculate the missing ratings with the features. Dimension reduction methods such as Singular Value Decomposition (SVD) [3], [4], principle component analysis(PCA) **??** can deal with the scalability problem and giving good rating predictions despite the sparsity problem.

Restricted Boltzmann Machine (RBM) is often used as a generative model, to model the distribution of the input vector. Recent research also bring it into the problem of CF tasks [5], [6]. It has been shown to be have good performance as Matrix-Factorization based models like SVD, even perform slightly better. Because the error by RBM-based CF model and SVD are different, combining the result can also improve the predicted ratings. Combining the results of RBM and Probabilistic Matrix Factorization (PMF) reduces the RMSE 7% than baseline on Netflix competition dataset [7]. Since the RBM is undirected graphic model, the inference of RBM is intractable, we often resort to approximation method like Gibbs sampling when training it. Recently, the CD-k training method have shown to yield good result for RBM training [8]. But with the number training epoch increases, the k value have to be many times larger than early training stage, which makes the whole training still very time consuming. And it is often not easy to decide when to increase k.

Another neural latent factor model, Neural Autoregressive Density Estimator (NADE) [9], which is extended from RBM, providing tractable distribution estimation using an autoregressive approach. Because it is a feed-forward neural network, common tricks for neural network training, even second order optimization methods [10], [11], can be easily applied to it. It is shown to have comparable performance in density estimation task, and also be successfully used to model the topics of documents [12].

In this paper, we introduce a novel Neural Autoregressive framework for Collaborative Filtering (NACF) to the task of user-item rating prediction, and evaluate it in different ways. First, we model the user-item ratings with different visible units, finding the linear output units perform best. We also experimented how the orderings affects the performance, and propose the dual reversed ordering approach to overcome external bias caused by data sparsity and single ordering. We show that it is easy to combine with extra attributes of users or items to improve the performance, and many extensions of NACF are promising for real-world applications. Finally, we show the NACF model can be easily trained compared to RBM-based models, and the item-based NACF yielding better performance than corresponding RBM on two MovieLens datasets.

The rest of the paper is organized as follows: Section II presents preliminary knowledge for our model, Section III illustrates the NACF model and how we extend it to adapt to CF tasks, Section IV describes our experiments, Section V discuss the experiment results and future extensions and Section VI concludes this paper.

II. Background

A. Restricted Boltzmann Machine

RBM is a undirected graphical model, a bipartite whose distribution is energy based. The energy function of binary hidden and visible units is

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{h}^T \boldsymbol{W} \boldsymbol{v} - \boldsymbol{b}^T \boldsymbol{h} - \boldsymbol{c}^T \boldsymbol{v} \tag{1}$$

The marginal distribution of visible units is given by

$$p(\boldsymbol{v}) = \frac{\Sigma_{\boldsymbol{h}} exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\Sigma_{\boldsymbol{v}', \boldsymbol{h}} exp(-E(\boldsymbol{v}', \boldsymbol{h}))} \tag{2}$$

$\boldsymbol{W}$ is the weight matrix, $\boldsymbol{v}$, $\boldsymbol{h}$ represent the visible input vector and hidden vector respectively. $\boldsymbol{b}$ is the bias for visible units, $\boldsymbol{c}$ is the bias for the hidden units. Given visible units the probability of hidden unit h_j is activated:

$$p(h_j = 1|\boldsymbol{v}) = sigm(c_j + \boldsymbol{W}_{:j}^T \boldsymbol{v}) \tag{3}$$

Respectively, given hidden units the i-th visible unit is activated:

$$p(v_i = 1|\boldsymbol{h}) = sigm(b_i + \boldsymbol{W}_{i:} \boldsymbol{h}) \tag{4}$$

The activation function is $sigm(x = 1/(1 + exp(-x))$. The visible units can also be modeled by softmax units or Gaussian units, which the training and inference algorithm can be easily extended.

For parameter learning, we can get the gradient for W from

$$\frac{\partial \log p(\boldsymbol{v})}{\partial W_{ij}} = < v_i h_j >_{data} - < v_i h_j >_{model} \tag{5}$$

where $< v_i h_j >_{data}$ represents the expectation of observed visible unit v_i and hidden unit h_j are on simultaneously, the value of h_j is get from Eq.3. And $< v_i h_j >_{model}$ represents the model driven expectation of reproduced v_i and h_j, which can not be calculated directly because of the undirected connections. It has to seek to slow MCMC sampling method to get the approximation of the expectation.

Recently an algorithm called Contrastive Divergence (CD), which actually use the truncated k step Gibbs sampling instead, to approximate the gradient, yields good results for RBM training [8], [13]. However, in order to get better results, the value of k has to increase to get more accurate gradients as the training stage increases. And it is often not easy to decide when to increase k to get a better distribution estimators.

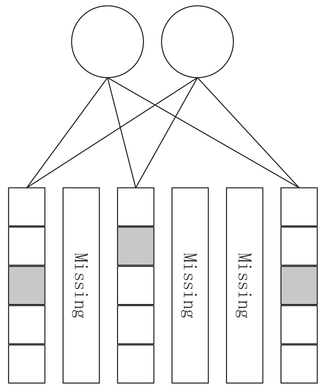


Fig. 1. Multinomial RBM for collaborative filtering

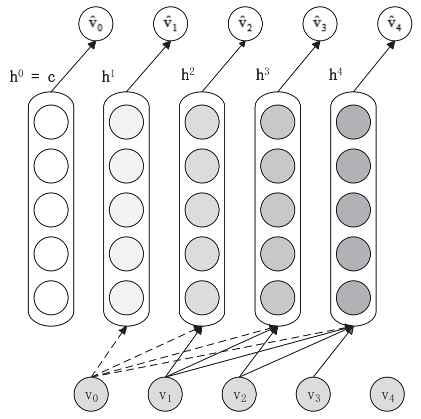


Fig. 2. NADE model is an autoregressive feed forward neural network with binary stochastic units. The dashed line indicates the weight from v_0 to the corresponding hidden state is the same for any h^j where $j > i$. And this applies to all other visible units, except for the last visible unit because it is the right most.

B. RBM for Collaborative Filtering

Given $M \times F$ user-item matrix, N is the number of users, M is the number of items, each rating entry is a integer value from 1 to K.

As shown in Fig.1, the corresponding RBM has M visible K-way softmax units, F binary hidden units, and weight matrix $\boldsymbol{W}$ size is $M \times F \times K$. Each row of the matrix, which is the ratings for all the items of a particular user, is a training case of the RBM. For different users, the missing rating entries tend to be different. In the training procedure, we just ignore the corresponding entries of $\boldsymbol{W}$, which means the structure of the RBM varies between different training cases, yet sharing the same weight matrix.

In the inference procedure, we using non-empty item ratings to get the hidden values $\boldsymbol{h}$ (see Eq.(3)), then through Eq.(4) to impute the missing item ratings with the output or expectation of the visible softmax units.

C. The Neural Autoregressive Distribution Estimator

NADE is a generative model extended from RBM by factoring the marginal distribution of the visible units $p(\boldsymbol{v})$

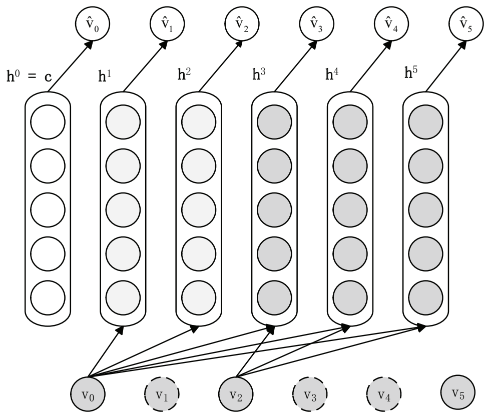


Fig. 3. Figure for NACF model. For a user-based NACF, the visible units are the rating for all items of a user. The visible units with dashed border like v_1, v_3 and v_4 represent the empty entries of item rating. Because there's no contribution from v_1, v_3 and v_4, we have $\boldsymbol{h}^1 = \boldsymbol{h}^2$, $\boldsymbol{h}^3 = \boldsymbol{h}^4 = \boldsymbol{h}^5$.

to conditional distribution

$$\begin{aligned} p(\boldsymbol{v}) &= \prod_{i=1}^{D} p(v_i|\boldsymbol{v}_{<i}) \\ &= \prod_{i=1}^{D} \frac{p(v_i, \boldsymbol{v}_{<i})}{p(\boldsymbol{v}_{<i})} \\ &= \prod_{i=1}^{D} \frac{\Sigma_{\boldsymbol{v}_{>i}} \Sigma_{\boldsymbol{h}} exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\Sigma_{\boldsymbol{v}_{j \geq i}} \Sigma_{\boldsymbol{h}} exp(-E(\boldsymbol{v}, \boldsymbol{h}))} \end{aligned} \tag{6}$$

D is the dimension of the visible input vector. From Eq.(2) we can derive the $p(v_i = 1|\boldsymbol{v}_{<i})$ is also intractable because of its partition function. However, it can be obtained by finding $q(v_i, \boldsymbol{v}_{>i}, \boldsymbol{h}|\boldsymbol{v}_{<i})$ to approximate the true conditional $p(v_i, \boldsymbol{v}_{>i}, \boldsymbol{h}|\boldsymbol{v}_{<i})$ by minimizing the KL-divergence (see [9]). The conditional distribution is computed in an autoregressive feed forward approach:

$$\boldsymbol{h}^i(\boldsymbol{v}_{<i}) = sigm(\boldsymbol{c} + \boldsymbol{W}^T_{:<i} \boldsymbol{v}_{<i}) \tag{7}$$

$$p(v_i|\boldsymbol{v}_{<i}) = sigm(\boldsymbol{b} + \boldsymbol{V}_{<i:} \boldsymbol{h}^i(\boldsymbol{v}_{<i})) \tag{8}$$

Like previous definition for RBM, $\boldsymbol{W}$ is the weight matrix from visible units to hidden units, $\boldsymbol{c}$ and $\boldsymbol{b}$ are the bias of hidden and visible units respectively. $\boldsymbol{V}$ is the weight matrix form hidden units to visible units. This is different from RBM, the untied weight matrix $\boldsymbol{V}$ can lead to better distribution estimator [9] .

$\boldsymbol{v}_{<i}$ is the visible units left to v_i, $\boldsymbol{W}^T_{:<i}$ and $\boldsymbol{V}_{<i:}$ is the corresponding weight matrix for $\boldsymbol{v}_{<i}$. $\boldsymbol{h}^i(\boldsymbol{v}_{<i})$ denotes the hidden activation contributed by visible units left to v_i, and its superscript i denotes that it is only for calculating the conditional distribution $p(v_i|\boldsymbol{v}_{<i})$.

As shown in Fig.2, $\hat{v}_i$ is the expectations for $p(v_i = 1)$, which is calculated from the contributions from $v_{<i}$. For the first visible unit, the expectation is given only by the bias $\boldsymbol{c}$. Thus, the neural network can be trained by standard back-propagation algorithm to minimize the cross-entropy error.

III. THE MODEL

A. *The Neural Autoregressive framework for Collaborative Filtering*

Now we bring in the NACF model. As the RBM model for CF, we want it to ignore the missing ratings for a particular user in the training process. However, when come across missing ratings (here we assuming it to be binary), the conditional distribution chain as the Eq.(6) are broken. From Fig.2 we can see that, for a full visible vector, each visible unit contribute to the activation of the hidden units for the ones that right to it. As shown in Eq.(8) the conditional distribution is directly calculated from $\boldsymbol{h}^i(\boldsymbol{v}_{<i})$, so we can just ignore the contribution of missing rating entries when calculating $\boldsymbol{h}^i(\boldsymbol{v}_{<i})$ using Eq.(7).

As shown in Fig.3, if there are continuous missing entries between two visible units, the activation of hidden units remains the same. That's to say, if a rating for a visible unit v_i is missing, the corresponding hidden status for calculating $\boldsymbol{h}^i(\boldsymbol{v}_{<i})$ is the same as $\boldsymbol{h}^{i-1}(\boldsymbol{v}_{<i-1})$. However, it would be a problem to predict some empty ratings when they are the left most in the specific ordering. It means that there's no one to contribute to their corresponding hidden states, and consequently the predictions is calculated only with its bias. This problem becomes severe when most entries of the input vector $\boldsymbol{v}$ are empty. We show how to address this problem later in Section III-C.

B. *Modeling rating values*

We demonstrate that a basic NACF model can be easily applied to modeling the binary ratings. By taking a row or a column of user-item rating matrix as one training case, we can get our user-based or item-based NACF. The user-item ratings in the real world are often from 1 to K. RBM model use softmax visible units to model the user-item ratings, and the training complexity is linear to the number of non-empty ratings in the user-item matrix. However, for NACF, 1-of-K representation like softmax will expand the weight matrix by K times. And because of its autoregressive approach to model the conditional distribution, the training complexity also increases to K times. So we find other approaches to bypass the problem.

We first try to normalize each entry of user-item ratings to zero mean, standard deviation, just like RBM with Gaussian visible units [14]. This approach convert the integer ratings to valid probability values, which can be easily fed into NACF models with binary stochastic visible units. We denote this as Gaussian-NACF, and its training and inference procedure is the same as basic NACF model.

We also extend the NACF to linear visible units instead of binary stochastic units, and this brings many advantages. First, with the linear units, it is easy to adapt to any possible rating values, keeping the number of parameters to $O(HD)$ instead of $O(HDK)$ for RBM. Second, the linear visible units are also naturally more fit to rating values than 1-of-K representation [6]. For example, if a true rating value is 4, the 1-of-K multinomial representation might take rating value 1 and 3 as the same, while linear real-value representation will take 3 as a better predication, because it is more close to 4 than rating value 1.

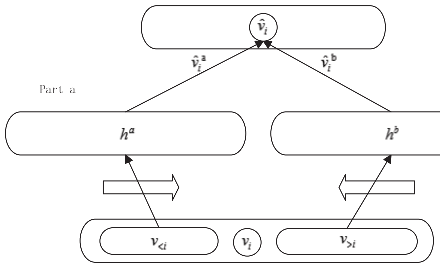


Fig. 4. Dual reversed ordering NACF (Dual-NACF) is composed by two NACF with reversed orderings denoted as part a and part b. The white hollow arrows represents the autoregressive directions, and the two orderings are exactly reversed.

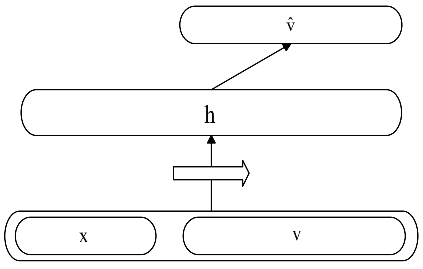


Fig. 5. The whole visible layer of Conditional NACF is composed by two parts. The first part is $\boldsymbol{x}$, represents the conditions like user profile for a user-based NACF. The second part is the ratings just like basic NACF. The output $\hat{\boldsymbol{v}}$ is calculated from the total contribution of two parts.

If we want to use linear units, we also need to change the error function from cross entropy to mean square error

$$E(\hat{\boldsymbol{v}}) = \frac{1}{D}\Sigma_{i=1}^{D}(\hat{v}_i - v_i)^2 \tag{9}$$

C. Multiple orderings to overcome sparsity

The conditional distribution chain of NACF implicitly introduce some particular ordering, and this assumption brings extra bias of the data. When the input vector are sparse, the problem becomes more severe. As explained before in Section III-A, some visible units may have few of other units left to them, and the corresponding hidden states may have enough information to predict the ratings precisely.

Random orderings of input units have different biases, and combine these results of the models may reduce the bias of the final rating predictions. The intuition is that, we can reduce the bias by combining all possible orderings. However, the number of all possible orderings is $O(D!)$, which means it is computationally impossible for NACF of considerable input size.

[15] introduced a method to generate NADE ensembles, but it is still time consuming for training all the parameters. At the inference stage, it still has $O(D!)$ time complexity to get the output. Moreover, our experiments shows that, the improvement of predicted ratings is not significant by simple averaging scheme. Instead we introduce a new ensemble method for NACF. It brings more improvement than simply averaging models of different orderings, while just double the training time of basic NACF with single ordering.

As shown in Fig.4, the output of dual-ordering NACF model is computed by the two NACF components. The two parts have different hidden units and different weight matrix, but they share the input data and the final output is calculated by averaging of the two outputs. The two components have reversed orderings. The intuition is that, for each visible unit v_i, the predicted probability (or just output value for linear unit) is determined by the contribution from all other visible units instead of just of the left ones. By using reversed orders, each visible unit can 'see' the contribution from exactly all visible units except itself, which might have better predictions than single random ordering model.

The way to calculate the output is defined as:

$$\hat{v}_i = (\hat{v}_i^a + \hat{v}_i^b)/2 \tag{10}$$

The $\hat{v}_i^a$ and $\hat{v}_i^b$is the output value of visible unit i in part a and part b respectively. When updating the weight matrix in part a or part b, we use $\hat{v}_i$ instead of $\hat{v}_i^a$ or $\hat{v}_i^b$ to calculate error. Our experiments show this approach is much better than simply averaging the output of several different orderings.

D. Combining other information

For real world recommendation problems, there's also some intrinsic properties or information for users or items other than the ratings on the items. For example, user's age, points of interest, category of products or genre of movies, may be useful for predicting the user-item ratings.

Because autoregressive connections of NADE provide a natural conditional distribution format, it give us convenience to combine any other information that reveals the property of the user or item.

As shown in Fig.5, the vector $\boldsymbol{x}$ is the other properties of a user or item. The value of hidden units of user-based or item-based NACF can be viewed as user or item features, one part is contributed by $\boldsymbol{x}$, the other is contributed by $\boldsymbol{v}$. We denote this model to be Conditional NACF, and the training and inference can be easily extended from basic NACF model.

In the training process of NACF, for a visible unit v_i, the gradients of corresponding weight $\boldsymbol{W}_{:<i}^T$ and $\boldsymbol{V}_{<i:}$ comes from the error from the visible units that are right to v_i. Note that for conditional NACF, we only need to give the predicted value of $\hat{\boldsymbol{v}}$, the $\hat{\boldsymbol{x}}$ is not required, which means the contribution from $\boldsymbol{x}_{<i}$ to x_i is not important. Thus for a conditional visible unit x_i, we don't need to incorporate the contribution from $\boldsymbol{x}_{>i}$ to calculate the corresponding weights.

The conditional NACF model gives us a way to address the problem of cold start. Assume that for a user-based collaborative filtering system, a new user just registered and few actions are done on the item set, the predicted ratings tend to be average ratings of the items, which is usually not reliable for personalized recommendation. However, for conditional NACF model, we can use the user profile or point of interests as conditions to give better rating predictions. And this also can be applied to item-based conditional NACF models if useful item properties are provided.

IV. Experiments

A. Datasets

We evaluate the NACF and its variations on two MovieLens datasets[1] with different size. All the user-item ratings of both two datasets are from 1 to 5. The MovieLens-100k dataset has 100,000 user-item ratings for 1,682 items by 943 users. The MovieLens-1M dataset has 1,000,000 ratings by 6,040 movies for 3,952 items. There are also some other attributes or infomation of user and item. For each user, the age, sex, occupation and zipcode is given. And for each movie, the movie name, url on IMDB and its genre are also provided. Both datasets are highly sparse, the rating density is 6.30% and 4.19% for MovieLens-100k and MovieLens-1M respectively. As previous research usually did, we use 5-fold cross validation training/testing datasets to evaluate the performance. The training/testing rate is 80%/20%, and these 5 testsets are disjoint with each other.

B. Evaluation Metric

We use *Root Mean Square Error* (RMSE) and *Mean Absolute Error* (MAE) as evaluation metrics for the 1 to 5 score prediction. These two metric are popular for collaborative filtering methods on rating prediction problems. They are designed to evaluate the overall difference between predicted score and testset real score. The RMSE metric is defined as:

$$RMSE = \sqrt{\frac{\Sigma_{i,j}(r_{ij} - \hat{r}_{ij})^2}{N_t}} \tag{11}$$

And the MAE is defined as:

$$MAE = \frac{\Sigma_{i,j}|r_{ij} - \hat{r}_{ij}|}{N_t} \tag{12}$$

where r_{ij} and $\hat{r}_{ij}$ are the true rating and rating predicted by model for item j by user i respectively. N_t is the number of user-item pairs exists in the testset. The empty entries in the testset usually are not counted to calcuate the RMSE or MAE.

C. Experiment Setup

Without tuning too much parameters, we fix the learning rate of NACF model to 0.1, with a decreasing parameter γ set to 0.0005, weight decay parameter λ fixed to 0.001. The number of visible units and hidden units varies from 10 to 50 regard to different datasets and NACF implementations. The approach to decrease the learning rate for each epoch is defined as:

$$lr_i = \frac{lr_0}{1 + \gamma \cdot i} \tag{13}$$

where lr_i is the learning rate for epoch i, lr_0 is the initial learning rate, and γ is the decreasing constant. All the NACF model are trained with simple Stochastic Gradient Descent (SGD) scheme, which means we update the weight for every training case. We denote the user-based NACF and item-based NACF to be U-NACF and I-NACF respectively.

In this part, we mainly intend to:

- Explore how NACF performs with different visible units i.e. Gaussian, linear and sigmoid visible units.
- Inspect how the ordering affects the performance of NACF.
- Evaluate the performance of Conditional-NACF compared to basic NACF.
- Compare NACF model with previous existing CF models, especially RBM-based models on MovieLens dataset.
- Show how NACF and previous existing CF models performs with different number of factors.

[1]http://www.grouplens.org/node/73

D. Different visible units

NADE is a distribution estimator designed to use stochastic binary visible and hidden units. The input value of NADE is a vector of binary values, by using the sigmoid nonlinearity, NACF can be directly applied to binary ratings.

However, since the ratings are usually integer values like 1 to 5 in MovieLens dataset, or probably some real valued ratings, we have to seek to other methods to model the ratings. One naive method is just simply scale the rating down in range $[0, 1]$, which is what we did for in the experiment of sigmoid nonlinearity visible units [2].

Another way is to use Guassian visible units to model the real-valued ratings. The Guassian visible units are actually linear units with independent Gaussian noise [14]. For user-based NACF, we calculate the mean and standard deviation of a column, that is, a specific item rating from all the users. For item-based NACF, we calculate the mean and standard deviation of a row. We normalize the ratings with corresponding mean and standard deviation to inputs with mean $\mu = 0$, with standard deviation $\delta = 1$ in the experiment. Because the data is very sparse, some rows or columns may be empty or only one value. This will make the standard deviation of the corresponding item or user ratings are zero. We reset these zero standard deviation to a default small value like 0.0000001 to avoid dividing the ratings by zero.

We also experimented NACF linear visible unit, so that it can apply to real-valued ratings. Instead of minimizing the cross entropy error, the corresponding error function changed to be mean square error.

The RMSE and MAE on MovieLens-100k of different visible units are as Fig.6 shows. All the results is given by NACF trained on 80% ratings and test on 20% data. We can see that, the decrease speed of RMSE and MAE of NACF with linear visible unit is the fastest. For stochastic binary units, which is denoted as 'sigmoid', converge slower than linear unit and Gaussian unit. The Gaussian visible units, initially converge faster than linear units, but later its RMSE and MAE becomes higher. This is mainly because the different loss function defined the Gaussian NACF and RMSE or MAE. And another reason is that, the Gaussian units are more sensible to learning rate than binary units or linear units [14]. Note that for experiments after this section, our result all is given by NACF with linear visible units.

[2]We scale the ratings by multiplying 0.15, thus the 1-to-5 ratings will fit in the values in [0,1]. Before we calculate RMSE and MAE, we recover the ratings to the original scale, so our result is compatible with previous work.

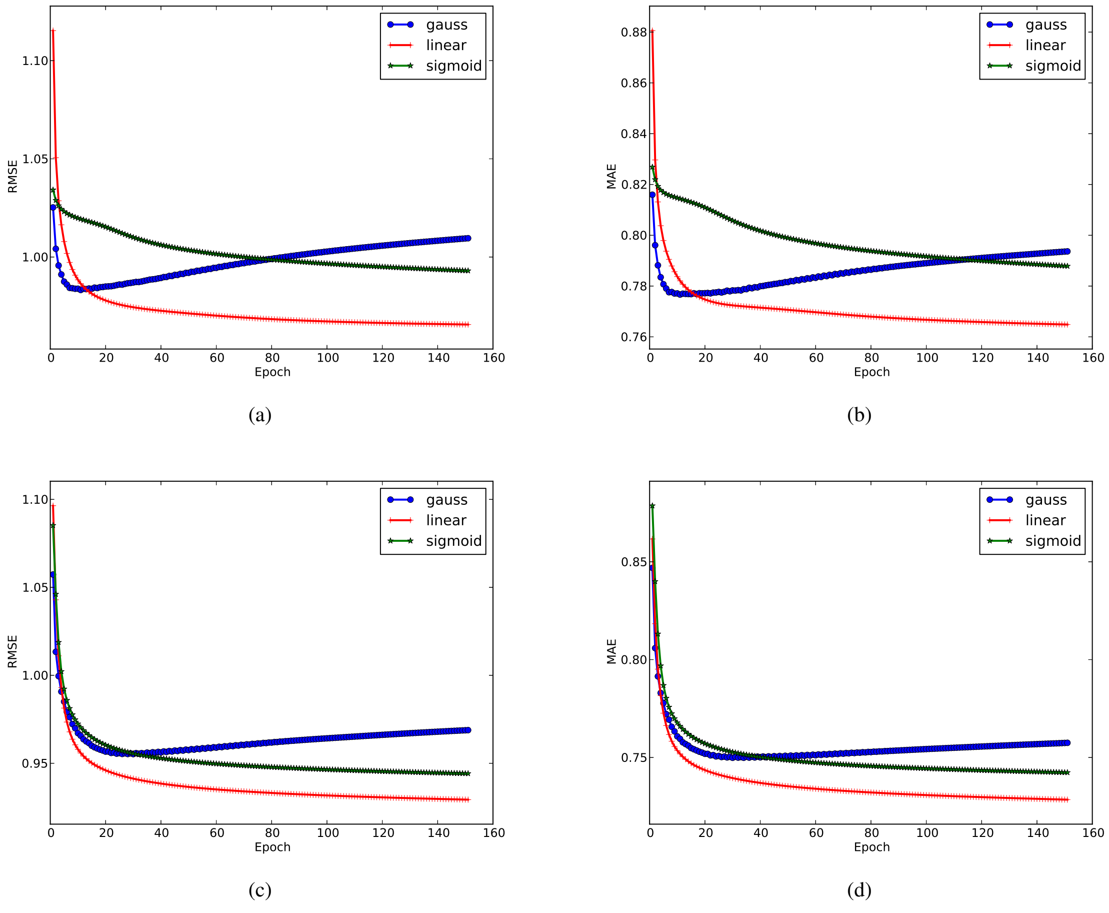


Fig. 6. The results for NACF using different visible units on MovieLens-100k dataset. (a) and (b) is the test RMSE and MAE for user-based NACF; (c) and (d) is the test RMSE and MAE curve for item-based NACF.

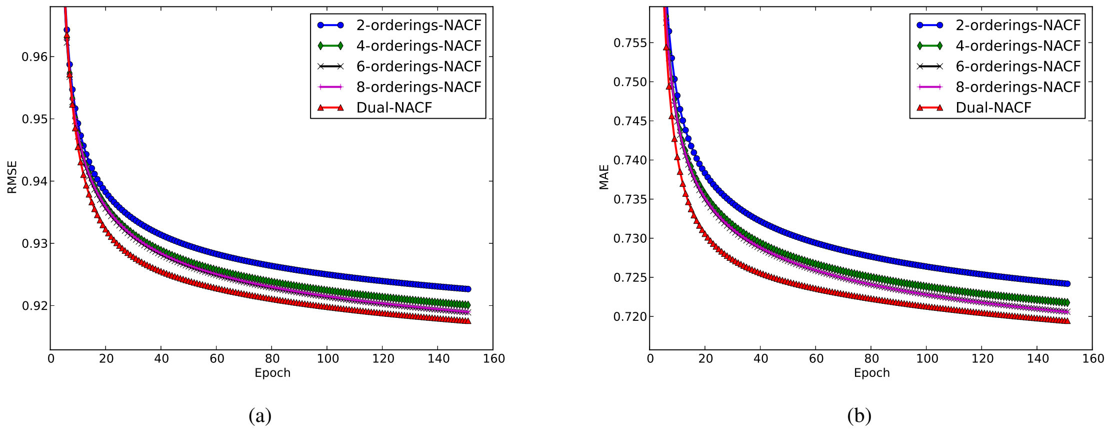
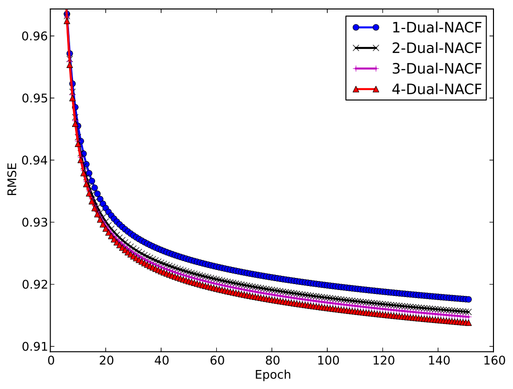
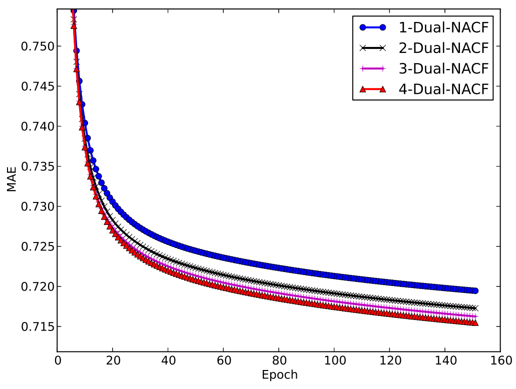


Fig. 7. Evaluation of NACF with different number of orderings and dual reversed NACF on MovieLens-100k. (a) the test RMSE curve; (b) the test MAE curve.



(a)



(b)

Fig. 8. Evaluation of averaging Dual-NACF with different number of orderings on MovieLens-100k. (a) the test RMSE curve; (b) the test MAE curve.

TABLE I. NACF AND C-NACF PERFORMANCE ON MOVIELENS-100K DATASET.

Metric	UC-NACF	U-NACF	IC-NACF	I-NACF
RMSE	**0.9690**	0.9694	**0.9198**	0.9251
MAE	0.7720	**0.7665**	**0.7226**	0.7267

E. *Multiple orderings to overcome sparsity*

Averaging the predictions of NACF with multiple orderings can improve the quality of predicted ratings. We test NACF with different orderings, and evaluate the dual reversed ordering NACF (denoted as Dual-NACF) on the MovieLens-100k.

As shown in Fig 7, the RMSE and MAE of NACF with 2, 4 and 6 orderings decreases with incorporating more orderings. However, we can see that the RMSE and MAE for 6 orderings and 8 orderings are almost the same. This means simply averaging the results from more NACF with different orderings does not help.

Our proposed Dual-NACF outperforms simple averaging approach significantly. Moreover, the training time complexity of Dual-NACF is just twice of training a single ordering NACF, while averaging approach may cost much more time for the same performance. In practice applications, we can also average the result from different Dual-NACF. This brings better rating prediction, because it combines models from different hypothesis.

The result of averaging different Dual-NACF is shown in Fig 8. We can see that, the result of averaging 2 Dual-NACF is significantly better than single Dual-NACF, and the 4 Dual-NACF performs best.

F. *Combining other information*

The MovieLens dataset also provides extra information of the users and movies, here we evaluate the performance of the conditional user-based and item-based NACF.

For each user, the dataset provide the age, sex and occupation of the user. As described in Section III-D, the x input vector is composed of some user information for U-NACF. We scale the age by 150, and sex is set to 1 for male and 0 for female. The occupation of the user is modeled by a 1-of-K style binary vector $\boldsymbol{x}^o =< x_0^o, x_1^o, ..., x_{k-1}^o >$, the entry x_i set to one if the user is occupied in the i-th occupation.

For each movie, the dataset provides movie genre. The x binary input vector here is $\boldsymbol{x}^g =< x_0^g, x_1^g, ..., x_{l-1}^g >$, and there may be multiple entries in $\boldsymbol{x}^g$ to be 1.

The training procedure of the conditional NACF is as described in Section III-D. All the parameters are set to be the same for UC-NACF, U-NACF, IC-NACF and I-NACF. The number of hidden units are all set to be 10. The ordering of rating visible units for UC-NACF and U-NACF are the same, which is likewise for IC-NACF and I-NACF. Only the number of visible units is different, because the extra condition vector and the number of users and items are different.

On the MovieLens-100k dataset, we do a 5-fold cross validation for the four models. We can see from Table I that, for user-based models, UC-NACF have minor RMSE decrease than U-NACF, but the MAE is slightly higher than U-NACF. For item-based models, IC-NACF outperforms I-NACF on MAE and RMSE significantly. From the result, we can see that movie genre seems to be more helpful for rating prediction than user information like age, sex and occupation.

G. *Comparison with previous existing models*

Here we compare the performance with previous collaborative filtering models on MovieLens-100k and MovieLens-1M. We evaluate the Dual-NACF and the single ordering NACF with linear visible units without combining any other information. The result of RBM-based models comes from [6], while the PMF and NACF results are evaluated with common 5-fold cross validation.

For the MovieLens-100k dataset, we set the number of factors to 10. The U-RBM, I-RBM, I-RBM+INB and UI-RBM are from RBM based methods [6]. I-RBM+INB is a neighborhood based method that utilizing the latent features of item-based RBM. The Latent-CF [16] performs best, and UI-RBM also have comparable performance on MovieLens-100k dataset. Because UI-RBM is a unified model, it performs better than U-RBM and I-RBM. Our NACF models, despite

TABLE II. MAE OF CF MODELS ON MOVIELENS-100K DATASET, AMONG WHICH OUR NACF MODELS IS IN BOLD.

CF Model	MAE
PMF	0.729
U-RBM	0.779
I-RBM	0.775
U-NACF	**0.772**
I-NACF	**0.727**
U-Dual-NACF	**0.750**
I-Dual-NACF	**0.715**
I-RBM+INB	0.699
UI-RBM	0.690
Latent CF	0.685

TABLE III. MAE OF CF MODELS ON MOVIELENS-1M DATASET, AMONG WHICH OUR NACF MODELS IS IN BOLD.

CF Model	MAE
PMF	0.689
Real U-RBM	0.762
Real I-RBM	0.761
Multinomial U-RBM	0.711
Multinomial I-RBM	0.710
U-NACF	**0.733**
I-NACF	**0.693**
U-Dual-NACF	**0.725**
I-Dual-NACF	**0.686**
Real I-RBM+INB	0.669
Real UI-RBM	0.645

its single random ordering, performs better than corresponding user based or item based RBM model. And the Dual-NACF performs better than corresponding NACF model.

For the MovieLens-1M dataset, the number of factors for our NACF based models is set to be 50. We can see from Table III that, our I-NACF model performs better than both Real I-RBM and Multinomial I-RBM, U-NACF outperforms Real U-RBM, but the MAE is slightly higher than Multinomial U-RBM. Dual-NACF still performs better than corresponding NACF models. Other neighborhood models like Real I-RBM+INB, or models like UI-RBM or UI-BM based on full matrix, performs better than single user-based or item-based models.

H. Sensitivity for different number of factors

We compared NACF-based models with RBM-based and PMF on the two MovieLens datasets with different number of factors. As shown in Fig.9, among the four models, RBM model has the highest RMSE and MAE, and the result are affected greatly by the number of factors. All the NACF model here are item-based. On the MovieLens-100k dataset, the NACF and PMF have similar RMSE and MAE, and the results are all stable with different number of factors. The Dual-NACF performs best on MovieLens-100k datasets. On the MovieLens-1M dataset, Dual-NACF has comparable RMSE with PMF, and the MAE of Dual-NACF is the lowest. We can see the NACD-based models, especially Dual-NACF model are stable to different number of factors.

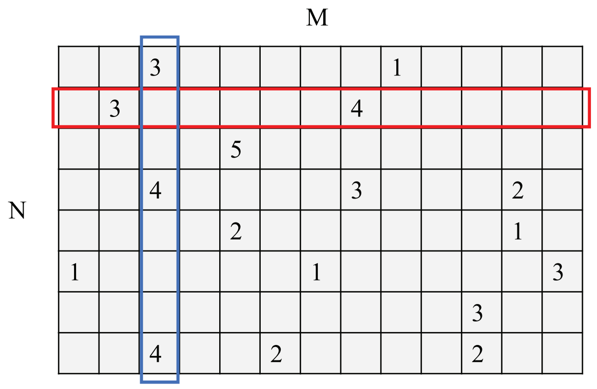


Fig. 10. An example user-item rating matrix. The red box stands for a user-based training case; the blue box stands for an item-based training case.

V. DISCUSSION

A. Difference of performance for U-NACF and I-NACF

One important observation from above experiment is that, item-based NACF or Dual-NACF outperforms the user-based NACF or Dual-NACF significantly, while the corresponding user-based and item-based RBM model have similar performance.

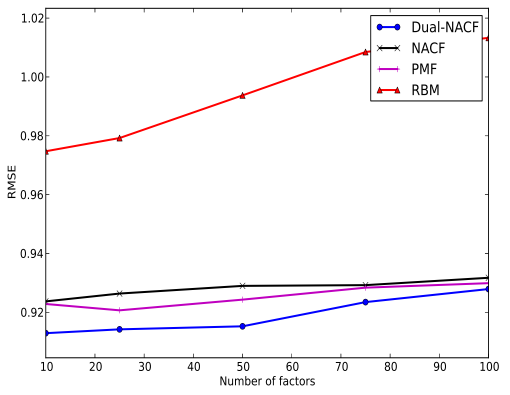
The reason is mainly because the shape of user-item rating matrix. For the MovieLens-100k dataset, number of users $N = 945$, number of items $M = 1682$, we have $M/N = 1.77$. For the MovieLens-1M dataset, $N = 3952$ and $M = 6040$, we have $M/N = 1.53$. In practice applications, the number of users often larger than the number of items often, i.e. $M/N > 1$.

How does M/N affects the performance of U-NACF and I-NACF? We illustrate our explanation in Fig. 10. We can see that, a user-based training case contains more entries than item-based training case. A U-NACF has more visible units than an I-NACF, which means the model is more complex, because of larger weight matrix $\boldsymbol{W}$ and $\boldsymbol{V}$ if number of hidden units is the same. Moreover, the number of training case for user-based and item-based model is N and M respectively, and we have $N < M$, so the user-based approach suffers from sparsity more than item-based approach. This is of vital importance to the difference of performance for single ordering U-NACF and I-NACF.

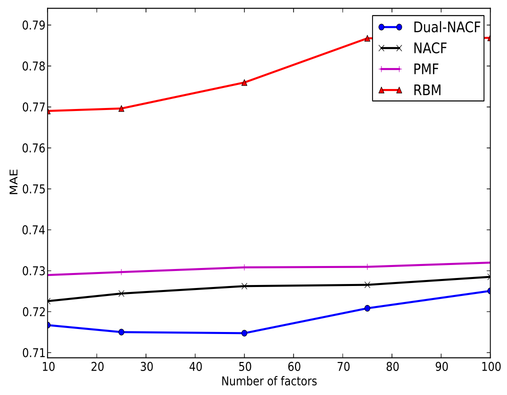
In practice, we prefer I-NACF to U-NACF. As the above experiments shows, the I-NACF outperforms the corresponding RBM-based models. And combining the I-NACF with neighborhood method will give better rating predictions.

B. Training scheme of NACF

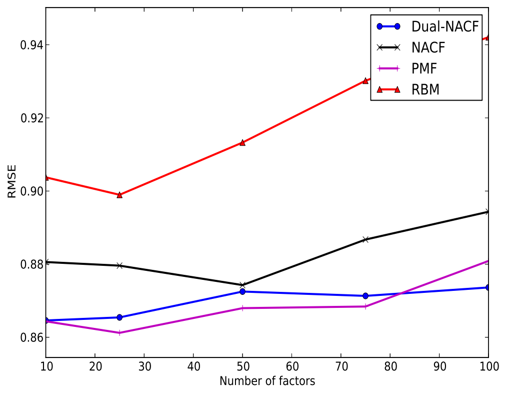
In this paper, we choose simple per-case SGD training scheme for NACF models. We can also use mini-batch SGD



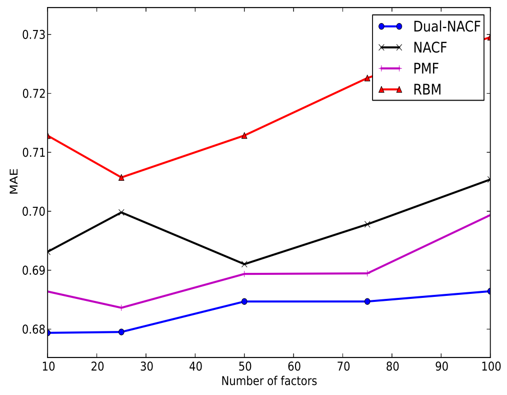
(a)



(b)



(c)



(d)

Fig. 9. The results for CF models with different number of factors on MovieLens-100k dataset. (a) and (b) is the test RMSE and MAE of CF models on MovieLens-100k; (c) and (d) is the test RMSE and MAE of CF models on MovieLens-1M.

for larger datasets to reduce weight update times. RBM based models often seek to CD-k training approach, which is actually based on a truncated Gibbs sampling. CD-1 is fast, but if we want to get more accurate results, we have to increase the number of steps of sampling. And it is also not easy to decide when to increase k. In contrast, because the NACF is actually a feedforward neural network, the parallel version of NACF training algorithm can be easier to get. Recent research on parallel optimization algorithms on neural networks especially deep neural networks are more popular [17]–[19], and second order optimization method can also be applied to accelerate the speed of NACF to converge [10], [11].

C. Model Extensions

1) Considering content and social information: For real world collaborative filtering tasks, many useful information of users and items are available beside the user-item ratings. Utilizing these user or item features often improves the user-item rating predictions of collaborative filtering systems.

For example, for the scientific articles, the topic of the document make significant difference for the user to make decision to cite it or add it to favorite list. Although everyone has his or her own preference, it is often hard for us to decide which to choose among the numerous products or services on the Internet. However, if one of your friends recommends a movie to you, you are more probable to watch it than just knowing the genre or only seeing a brief introduction of the movie. Thus, the social network can also provide meaningful information for recommendation. Recent research often combine graphic models with Matrix Factorization methods to incorporate the content or social information [20], [21]. These models share the user or item feature space with other information by using an additive approach, that is, the feature vector value are contributed by both the collaborative information and social or content information. But the proportion of two parts is decided by different datasets, leading to a parameter tuning problem, which is painful for large datasets.

Our NACF framework can be easily extended to incorporate the user or item information. As shown in Fig. 5, the x vector can also be word inputs from item contents, or social matrix rows. The contribution from x can be interpretable representation such as topics or user social preference, and the contribution from ratings can be viewed as offsets to these topics or preference. This is a little different from Section III-D, which just utilizing the extra user or item information without considering interpretability. Moreover, by treating it as an autoregressive feedforward neural network, the training procedure is simple and straight forward, which may save us from rigid parameter tuning on coefficients of contribution from ratings or other information.

2) Unified user-based and item-based model : The user-based or item-based RBM approach can be combined together to improve the rating prediction results. Likewise, the NACF framework can be directly extended by using this approach. However, it still need further experimental studies to see the performance of training a unified model with averaged the outputs. As we discussed before, unlike the RBM-based models, the item-based NACF outperforms user-based NACF significantly. This might cause the unified NACF model perform worse than item-based NACF. Moreover, the training procedure have to change from SGD to full batch update, which can be an obstacle for large-scale applications. Once the model is unified, combining both the item and user information is easy and straight forward, which is quite meaningful in real world applications.

VI. Conclusion

In this paper, we introduced a novel neural autoregressive model for collaborative filtering, and evaluated the model in different ways. We studied how is the performance of different visible units to model the user-item ratings, finding the linear visible units yielding best RMSE or MAE and converge fastest.

Furthermore, because the NACF models are based on some random orderings, which may introduce extra bias, we studied how does the ordering affects the result RMSE or MAE on testset. Experiments show simply averaging the output of NACF with multiple orderings can not improve the predicted ratings. We propose to take the average outputs of NACF with exactly reversed orderings to learn the gradients and update the weight matrix, which outperforms the simple averaging approach significantly.

We also illustrated the natural conditional distribution form of NACF model provides convenience for us to introduce extra user or item attributes. And we discussed future extensions to incorporate content or social network via the conditional NACF form.

Finally, we showed the NACF model can be easily trained compared to RBM-based models, and the item-based NACF yielding better performance than corresponding RBM on two MovieLens datasets.

In future work, we are going to adapt the extensions of NACF to more datasets with heterogeneous kinds of information, and compare with other exists models on them. Meanwhile, implementing more efficient CPU or GPU parallel training algorithms for NACF is also a consideration.

References

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[2] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[3] D. Billsus and M. J. Pazzani, “Learning collaborative information filters.” in *ICML*, vol. 98, 1998, pp. 46–54.

[4] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[5] R. Salakhutdinov, A. Mnih, and G. Hinton, “Restricted boltzmann machines for collaborative filtering,” in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 791–798.

[6] K. Georgiev and P. Nakov, “A non-iid framework for collaborative filtering with restricted boltzmann machines,” in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1148–1156.

[7] R. Salakhutdinov and A. Mnih, “Probabilistic matrix factorization.” in *NIPS*, vol. 1, no. 1, 2007, pp. 2–1.

[8] G. E. Hinton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” *Science*, vol. 313, pp. 504–507, 2006.

[9] H. Larochelle and I. Murray, “The neural autoregressive distribution estimator,” *Journal of Machine Learning Research*, vol. 15, pp. 29–37, 2011.

[10] D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization,” *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[11] J. Martens, “Deep learning via Hessian-free optimization,” in *International Conference on Machine Learning*, 2010, pp. 735–742.

[12] H. Larochelle and S. Lauly, “A neural autoregressive topic model.” in *NIPS*, 2012, pp. 2717–2725.

[13] Y. Bengio, “Learning deep architectures for ai,” *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[14] G. Hinton, “A practical guide to training restricted boltzmann machines,” *Momentum*, vol. 9, no. 1, p. 926, 2010.

[15] B. Uria, I. Murray, and H. Larochelle, “A deep and tractable density estimator,” *arXiv preprint arXiv:1310.1757*, 2013.

[16] H. Langseth and T. D. Nielsen, “A latent model for collaborative filtering,” *Int. J. Approx. Reasoning*, vol. 53, no. 4, pp. 447–466, 2012.

[17] M. Zinkevich, M. Weimer, A. Smola, and L. Li, “Parallelized stochastic gradient descent,” in *Neural Information Processing Systems*, 2010, pp. 2595–2603.

[18] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, “Large scale distributed deep networks,” in *NIPS*, 2012.

[19] S. Scanzio, S. Cumani, R. Gemello, F. Mana, and P. Laface, “Parallel implementation of artificial neural network training,” in *International Conference on Acoustics, Speech, and Signal Processing*, 2010, pp. 4902–4905.

[20] S. Purushotham, Y. Liu, and C.-C. J. Kuo, “Collaborative topic regression with social matrix factorization for recommendation systems,” *arXiv preprint arXiv:1206.4684*, 2012.

[21] M. Jamali and M. Ester, “A matrix factorization technique with trust propagation for recommendation in social networks,” in *Conference on Recommender Systems*, 2010, pp. 135–142.