Neural Autoregressive Collaborative Filtering for Implicit Feedback

Yin Zheng
Hulu LLC.
Beijing, China, 100084
yin.zheng@hulu.com

Cailiang Liu
Hulu LLC.
Beijing, China, 100084
cailiang@hulu.com

Bangsheng Tang
Hulu LLC.
Beijing, China, 100084
bangsheng@hulu.com

Hanning Zhou
Hulu LLC.
Beijing, China, 100084
eric.zhou@hulu.com

ABSTRACT

This paper proposes *implicit* CF-NADE, a neural autoregressive model for collaborative filtering tasks using implicit feedback(e.g. click/watch/browse behaviors). We first convert a user's implicit feedback into a "like" vector and a confidence vector, and then model the probability of the "like" vector, weighted by the confidence vector. The training objective of implicit CF-NADE is to maximize a weighted negative log-likelihood. We test the performance of implicit CF-NADE on a dataset collected from a popular digital TV streaming service. More specifically, in the experiments, we describe how to convert watch counts into implicit "relative rating", and feed into implicit CF-NADE. Then we compare the performance of implicit CF-NADE model with the popular implicit matrix factorization approach. Experimental results show that implicit CF-NADE significantly outperforms the baseline.

CCS Concepts

•Information systems → Collaborative filtering; •Computing methodologies → Neural networks; Learning latent representations;

Keywords

collaborative filtering; implicit feedback; deep learning; neural network

1. INTRODUCTION

Modern online systems rely heavily on recommender systems to help users identify items they might be interested in, from a usually massive catalog, and therefore provide personalized experience. The most popular and successful technique of building a recommender system is *collaborative filtering* (CF) [4], which predicts user preferences by analyzing past user behaviors and establishing relevance between items and also between users. There are two major types of inputs to a CF-based recommender system: 1) *explicit feedback*, e.g. 5-star ratings, likes/dislikes; and 2) *implicit feedback* e.g. watch/search/browse/purchase behaviors. Explicit feedback accurately reflects a user's preference over an item, and thus is most convenient to use. Techniques designed for explicit feedback, such as restricted Boltzmann machine (RBM) CF [18], matrix factorization [14, 17, 9], neural network matrix factorization[3], and recently developed *neural autoregressive distribution estimator for CF tasks* (CF-NADE) [24] have been highly successful in predicting explicit user preferences, which, to the best of our knowledge, is the state-of-the-art on MovieLens 1M, MovieLens 10M [5] and Netflix datasets [2].

In real-world applications, only a small fraction of users actively provide explicit feedback, which restricts the application of aforementioned methods. On the other hand, implicit feedback is abundant, as long as the user interacts with the online system. Hence, building recommender system using collaborative filtering on implicit feedback has attracted increasing attention. One major characteristic of implicit feedback is that there is only *positive* feedback, in that one can only tell whether a user *has* engaged with an item for how many times. Consider the size of the catalog, the number of items a user has engaged with is tiny. So implicit feedback is inherently *unbalanced* and *sparse*. Also, a user has not engaged with an item does not necessarily mean that he/she dislikes the item or the item is irrelevant, and in most cases it is because the user is unaware of the item. Therefore in literature, collaborative filtering using implicit feedback is sometimes referred to as *one-class collaborative filtering* (OCCF)[16].

A natural way of building recommender system using collaborative filtering for implicit feedback is to interpret implicit feedback as explicit feedback in a proper way and apply existing successful algorithms for explicit feedback, such as [7, 16]. In this paper, we describe a generalized CF-NADE for implicit feedback, which is referred to as *implicit* CF-NADE. Specifically, we first introduce the original CF-NADE model [24] for explicit feedback briefly in Section 3. Then, we focus on describing implicit CF-NADE in Section 4. We compare implicit CF-NADE with Implicit Matrix Factorization (IMF) approach [7], and show the performance comparison in Section 5.

2. RELATED WORK

Many previous works on recommender system using implicit feedback are based on matrix factorization. [7] proposes to employ matrix factorization where implicit feedback is treated as bi-

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

nary preferences and weighted according to the number of engagements. The work of [7] is quite popular and has been included into popular software libraries like MLlib of Spark [12]. [16] also formulates the problem as weighted matrix factorization, and proposes to use negative sampling to mitigate the unbalancedness problem. With a similar weighting strategy, inner products are replaced by logistic functions in the probabilistic model called logistic matrix factorization [8]. Besides weighting, values can be imputed for unobserved examples to indicate possible feedback. This method and its combination with weighting are discussed in [23]. In [11], multiple implicit feedback sources are considered, either by treating each source separately and combining with a linear model, or collectively embedding all feedback sources into a collective collaborative filtering model. Alternatively, SLIM [15] formulates the CF for implicit feedback as a convex optimization problem, which is recently generalized to LRec in [19].

With the recent success of deep learning in computer vision and natural language processing community [10, 21, 6, 13], neural networks have also found application in building recommender systems. For example, RBM-CF [18] and AutoRec [20] are successful approaches to model the users' explicit feedback, using restricted Boltzmann machine and autoencoder respectively. The recently developed CF-NADE [24] models explicit feedback with a neural autoregressive architecture. In this work, we will generalize CF-NADE and propose a novel CF model for implicit feedback.

3. CF-NADE

We start with the description of CF-NADE, a neural autoregressive architecture for CF tasks which has proved successful in modeling *explicit* ratings [24]. A user u's explicit ratings are denoted as $\mathbf{r}^u = (r^u_{m_{o_1}}, r^u_{m_{o_2}}, \dots, r^u_{m_{o_D}})$, where D is the number of items that the user has rated, $m_i \in \{1, 2, \dots, M\}$ is the index of the ith rated items, M is the total number of items, o is a D-tuple in the set of permutations of $(1, 2, \dots, D)$ which serves as an ordering of the D rated items, and $r^u_{m_{o_i}} \in \{1, 2, \dots, K\}$ denotes the rating that the user gave to item m_{o_i}. For simplicity, we will omit the index u of $\mathbf{r}^u$. As discussed in [24], a random order of the ratings works well in practice and is the key to extend CF-NADE to a deep model.

By the chain rule, CF-NADE models the joint probability of the rating vector $\mathbf{r}$ as a product of conditionals:

$$p(\mathbf{r}) = \prod_{i=1}^{D} p\left(r_{m_{o_i}} | \mathbf{r}_{m_{o_{<i}}}\right) \tag{1}$$

where $\mathbf{r}_{m_{o_{<i}}} = (r_{m_{o_1}}, r_{m_{o_2}}, \dots, r_{m_{o_{i-1}}})$ denotes the first $i-1$ elements of $\mathbf{r}$ indexed by o.

Each conditional in Equation 1 is modeled as:

$$p\left(r_{m_{o_i}} = k | \mathbf{r}_{m_{o_{<i}}}\right) = \frac{\exp\left(s^k_{m_{o_i}}\left(\mathbf{r}_{m_{o_{<i}}}\right)\right)}{\sum_{k'=1}^{K} \exp\left(s^{k'}_{m_{o_i}}\left(\mathbf{r}_{m_{o_{<i}}}\right)\right)} \tag{2}$$

$s^k_{m_{o_i}}(\mathbf{r}_{m_{o_{<i}}})$ is the score indicating the preference that the user gave rating k for item m_{o_i} given previous ratings $\mathbf{r}_{m_{o_{<i}}}$. The score of $s^k_{m_{o_i}}(\mathbf{r}_{m_{o_{<i}}})$ is computed by:

$$s^k_{m_{o_i}}\left(\mathbf{r}_{m_{o_{<i}}}\right) = d^k_{m_{o_i}} + \mathbf{V}^k_{m_{o_i},:}\mathbf{h}\left(\mathbf{r}_{m_{o_{<i}}}\right) \tag{3}$$

where $\mathbf{V}^k \in \mathbb{R}^{M \times H}$ and $\mathbf{d}^k \in \mathbb{R}^M$ are the connection matrix and the bias with rating k, respectively. And

$$\mathbf{h}\left(\mathbf{r}_{m_{o_{<i}}}\right) = \mathbf{g}\left(\mathbf{b} + \sum_{j<i} \mathbf{W}^{r_{m_{o_j}}}_{:,m_{o_j}}\right) \tag{4}$$

where $\mathbf{W}^k \in \mathbb{R}^{H \times M}$ is the connection matrix associated with rating k, $\mathbf{W}^k_{:,j} \in \mathbb{R}^H$ is the j^{th} column of $\mathbf{W}^k$ and $W^k_{i,j}$ is an interaction parameter between the i^{th} hidden unit and item j with rating k, $\mathbf{b} \in \mathbb{R}^H$ is the bias term, $\mathbf{g}(\cdot)$ is the activation function, such as $\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$,

Fitting CF-NADE can be simply maximizing the joint probability $p(\mathbf{r})$. As noticed in [24], maximizing the conditional of Equation 2 can only ensure that the probability of the true rating is the largest among all possibles, while leaving the ordinal nature of ratings disregarded. Hence, a ranking loss is proposed to be added in the objective function, and a significant improvement can be observed.

4. IMPLICIT CF-NADE

As discussed in Section 1, implicit feedback is abundant and easy to obtain. In this section, we describe how to adapt CF-NADE to *implicit* feedback. In the implicit feedback scenario, the "rating" $r^u_i \in \mathbb{R}_{\geq 0}$ that a user u gives to an item i is defined as the number of times that the user interacts with the item. Inspired by [7], we could define a binary scalar t^u_i by binarizing the r^u_i values:

$$t^u_i = \begin{cases} 1 & r^u_i > 0 \\ 0 & r^u_i = 0 \end{cases} \tag{5}$$

where $t^u_i = 1$ indicates user u likes item i as he/she has interacted with this item before, and if user u never interacted with item i, we think that there is no preference of user u on item i. However, deciding whether a user likes or dislikes an item by binarizing r^u_i is oversimplified and can be quite noisy. The reason is twofold: 1) in most cases, a user has not interacted with an item is because of unawareness, not dislike; and 2) the number of times the user interacts with the item can be a good indicator of how much the user likes the item, which is lost in $\mathbf{t}$ after the binarization.

Hence, we need to formalize the confidence that a user likes or dislike an item. Generally speaking, the confidence c^u_i should increase with r^u_i. In this work, we follow [7] and define the confidence as:

$$c^u_i = 1 + \alpha r^u_i \tag{6}$$

where α is the rate of confidence, which is a hyper-parameter controlling how fast the confidence c^u_i increases with r^u_i. We will show the impact of α in the experiments.

A user u's implicit feedback can now be represented as $\mathbf{t}^u = (t^u_1, t^u_2, \dots, t^u_M)$, with corresponding confidence levels as $\mathbf{c}^u = (c^u_1, c^u_2, \dots, c^u_M)$. In the rest of the paper, we will omit the superscript u for simplicity. Similar to CF-NADE, we model the probability of $\mathbf{t}^u$ as a product of conditionals, with the addition of confidence levels in the condition as:

$$p(\mathbf{t}|\mathbf{c}) = \prod_{i=1}^{M} p(t_i | \mathbf{t}_{<i}, \mathbf{c}) \tag{7}$$

where M is the number of items, and $\mathbf{t}_{<i}$ denotes preference of the previous $i-1$ items. Similar to CF-NADE model, the order of the items in Equation 7 is randomly shuffled. As the confidence level c_i should be paired with t_i, Equation 7 can then be rewritten as:

$$p(\mathbf{t}|\mathbf{c}) = \prod_{i=1}^{M} p(t_i | \mathbf{t}_{<i}, \mathbf{c}_{<i}) \tag{8}$$

To define the conditionals in Equation 8, we first define the hidden representation given previous $i-1$ ratings as:

$$\mathbf{h}\left(\mathbf{t}_{<i}, \mathbf{c}_{<i}\right) = \mathbf{g}\left(\mathbf{b} + \left(\mathbf{W}_{:,<i}\mathbf{t}_{<i} + \mathbf{A}_{:,<i}(1-\mathbf{t}_{<i})\right) \odot \mathbf{c}_{<i}\right) \tag{9}$$

where $\mathbf{g}(\cdot)$ is the activation function, $\odot$ is element-wise product, $\mathbf{W} \in \mathbb{R}^{H\times M}$ and $\mathbf{A} \in \mathbb{R}^{H\times M}$ are connection matrices associated with the *"like"* vector $\mathbf{t} = (t_1, t_2, \ldots, t_M)$ and the *"dislike"* vector $1-\mathbf{t}$, $\mathbf{b} \in \mathbb{R}^H$ is the bias vector, $\mathbf{X}_{:,<i}$ is the first $i-1$ columns of matrix $\mathbf{X}$, and similarly, $\mathbf{x}_{<i}$ is the first $i-1$ elements of vector $\mathbf{x}$, and $\mathbf{g}$ is the activation function. Note that if there is no confidence vector $\mathbf{c}$, the *"dislike"* connection matrix $\mathbf{A}$ would be redundant, as the difference between the corresponding parameters $W_{:,i}$ and $A_{:,i}$ for item i would be constant, in which case we can set $\hat{W}_{:,i} = W_{:,i} - A_{:,i}$ and $A_{:,i} = 0$. However, as the confidence varies for each user, $\mathbf{A}$ is required to capture the differences induced by varying confidence levels.

Then the conditionals in Equation 8 can be modeled as:

$$p\left(t_i = 1 | \mathbf{t}_{<i}, \mathbf{c}_{<i}\right) = \operatorname{sigm}\left(d_i + \mathbf{V}_{i,:}\mathbf{h}\left(\mathbf{t}_{<i}, \mathbf{c}_{<i}\right)\right) \tag{10}$$

where $\mathbf{V} \in \mathbb{R}^{M\times H}$ and $\mathbf{d} \in \mathbb{R}^M$ are the connection matrix and bias, and $\mathbf{V}_{i,:}$ and $\mathbf{d}_i$ are the corresponding ith row and element, respectively. $\operatorname{sigm}(x)$ denotes the sigmoid function: $\frac{1}{1+\exp(-x)}$.

Training an implicit CF-NADE can be done by minimizing the negative log-likelihood of Equation 8 directly, as the original CF-NADE. However, due to the noisiness of $\mathbf{t}$, it would be beneficial to incorporate confidence levels $\mathbf{c}$ to reflect the uncertainty of the elements in $\mathbf{t}$. Hence, we formalize the cost function as:

$$\mathcal{C} = -\sum_{i=1}^{M} c_i \log p\left(t_i | \mathbf{t}_{<i}, \mathbf{c}_{<i}\right). \tag{11}$$

In this way, fitting an element $p\left(t_i | \mathbf{t}_{<i}, \mathbf{c}_{<i}\right)$ wrong with high confidence in Equation 11 would cost more than one with low confidence. After the model is trained, we could predict the users' preference for each item as:

$$p(t_i = 1 | \mathbf{t}, \mathbf{c}) = \operatorname{sigm}\left(d_i + \mathbf{V}_{i,:}\mathbf{h}\left(\mathbf{t}, \mathbf{c}\right)\right) \tag{12}$$
$$\mathbf{h}\left(\mathbf{t}, \mathbf{c}\right) = \mathbf{g}\left(\mathbf{b} + \left(\mathbf{W}\mathbf{t} + \mathbf{A}(1-\mathbf{t})\right) \odot \mathbf{c}\right) \tag{13}$$

Thus, the model can learn to balance the huge number of low confident items, which are unobserved or interacted few times by the user, and the items with high confidences but of small quantity. As a result, the model could predict a user's preferences on unobserved items even if the we set $t_i = 0$ (dislike) as the input. And it might also happen that a user has interacted with an item i before ($t_i = 1$) but for only a few time, and the model predict that he/she does not like the item much. For example, a user watches the first episode of a TV show, but he/she never starts the next episode. In this situation, the corresponding t_i is 1 and c_i is small. Then the model can learn to predict a higher probability for $p(t_i = 0 | \mathbf{t}, \mathbf{c})$.

As is noticed by [22, 25, 24], with a randomly sampled ordering (c.f. Section 3), minimizing the negative log-likelihood for NADE based model is equivalent to randomly splitting the input vector into two parts, and treating the first part as the input, and optimizing to maximize the conditionals of the elements in the other part. We refer the reader to [24] for more details.

Thus, the training objective can be rewritten as:

$$\mathcal{C} = \frac{M}{M-i+1}\sum_{j\geq i} -\log p\left(t_{o_i} | \mathbf{t}_{o_{<i}}, \mathbf{c}_{o_{<i}}\right). \tag{14}$$

where o is a random ordering of all items that are sampled at each training update, and o_i and $o_{<i}$ are the ith element and first $i-1$ elements of o. In practice, we will adopt this training objective as it is easy to implement and can be extended to a deep version efficiently.

5. EXPERIMENTS

In this section, we test the performance of implicit CF-NADE on a dataset extracted from a digital TV streaming service, and compare the performance of implicit CF-NADE with the widely used Implicit Matrix Factorization approach [7], as implemented in Spark MLlib [12].

5.1 Dataset Description

Our dataset is composed of watch behaviors of 444480 randomly sampled active users on 17348 movies or TV shows. Here an *item* is either a movie or a TV show. For each user, we track the number of times that the user "completely" watched a movie or an episode of a TV Show during a period of 3 years from April 2013 to April 2016. We define a "complete" watch as any continuous stream of a video (episode or movie) from the start of a video to the end. For each user, the total number of watches of a TV show is the aggregation of the number of times he/she has watched any episode of that show.

As the number of episodes varies dramatically between different TV shows, and a movie usually has only one video, there would be a strong bias if we directly use the number of times for each TV shows and movie. To amend this problem, we propose to use a relative score $\tilde{r}_i^u$ as a surrogate of the "rating" r_i^u in Equation 5 and Equation 6, which is the percentage of users whose watch count of item i is no larger than user u's out of all users who have watched item i. For example, if a user watched a TV show i 10 times (counting all its episodes) and 100 out of 1000 people watched this TV show more no more than 10 times, where 1000 is the total number of people who have watched it, then the relative "rating" $\tilde{r}_i^u$ will be 0.1.This method also applies for movies, as we observed that there are always users who watch a movie multiple times. Figure 1 shows the histograms of watch count for a TV show and a movie, respectively. We can see that there are users who have watched a movie even more than 5 times. If a user's watch count on a specific item (a TV show or a movie) is the largest among all users who have watched the item, we have a good reason to assume that the user likes this item very much. Moreover, we can also observe that most of users watch a movie only once while watch a TV show more than 1 time (namely, more than one episode). Hence, the relative "rating" of a movie watched once will be higher than a TV show also watched once by a user. This is a desirable characteristic as watching a movie entirely strongly indicates that the user likes the movie, while this does not hold if a user has watched only one episode of a TV show. Another advantage of adopting relative "ratings" is that they take values in $[0, 1]$, easy to be interpreted as confidence levels. Thus, we replace r_i^u with $\tilde{r}_i^u$ in Equations 5 and 6, to compute the *like* vector $\mathbf{t}$ and the confidence vector $\mathbf{c}$ throughout all experiments.

5.2 Evaluation Metric

The difficulty with evaluating collaborative filtering methods in the implicit feedback scenario is that we can only test the performance of the models on items that a user has watched before, since non-engagement is noisy and does not necessarily mean negative feedback. As a result, we follow the famous work of [7] and use a recall based evaluation metric *mean percentage ranking* (MPR).

Specifically, for each user, we randomly select 10% relative "ratings" $\tilde{r}_i^u$ on TV shows or movies he/she has watched, as the test set. The samples in the test set are denotes as $\tilde{\gamma}_i^u$ and their counterparts $\tilde{r}_i^u$ in the training are set to 0. Thus, the training set does not con-

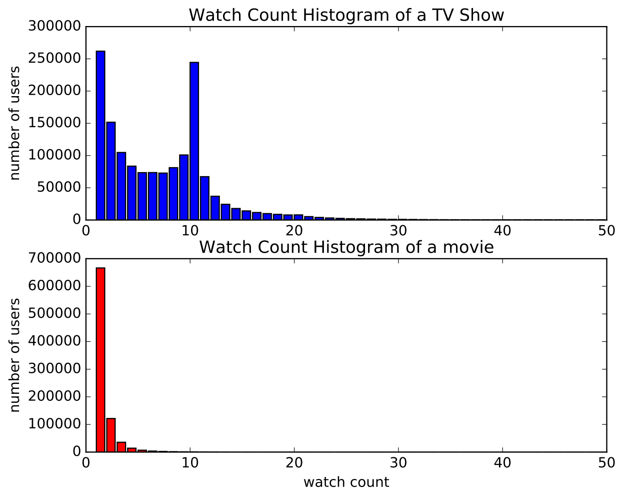

Figure 1: The histogram of watch count for a TV show (above) and a movie (below). The x-axis is the watch count and the y-axis is the number of users who have watched the item.

tain any watch behaviors of the test samples, and the goal of the CF models is to predict the users' preference on the test samples. For each user, we generated a ranked list of all candidate items sorted by preference from Equation 12. Let rank_i^u be the percentile ranking of item i for user u in the ranked list, where $\text{rank}_i^u = 0\%$ means that i is predicted as the highest recommended item for u, and $\text{rank}_i^u = 100\%$ signifies lowest. The MPR is then defined as:

$$\text{MPR} = \frac{\sum_{u,i} \gamma_i^u \text{rank}_i^u}{\sum_{u,i} \gamma_i^u}. \tag{15}$$

Lower values of MPR indicate that users will watch the TV shows or movies higher in the ranked list, which is desirable in practice. As an extreme example, a randomly shuffled list would have an expected MPR of 50%.

5.3 Results

In this section, we compare implicit CF-NADE with Implicit Matrix Factorization (IMF) [7]. Implicit CF-NADE is implemented using Tensorflow [1], and IMF is implemented using Spark MLlib [12]. In the experiments, the implicit CF-NADE model is trained with stochastic gradient decent optimizer with learning rate set to 0.01, batch size 200 and weight decay 0.01. We use only one hidden layer and the number of hidden units is set to 256. The number of factors for IMF is also set to 256 for a fair comparison. Figure 2 depicts MPRs on the test set for different choices of the rate of increase α in Equation 6. We can see that implicit CF-NADE always outperforms IMF, and α should be set to higher than 100 for good performance for both algorithms. The best test MPR (5.2436%) of implicit CF-NADE is achieved at $\alpha = 300$, and for IMF, the best test MPR is 7.362% with $\alpha = 100$.

6. CONCLUSION

In this paper, we generalized the recently developed CF-NADE to *implicit* CF-NADE for real-world collaborative filtering tasks, using implicit feedback. Specifically, we convert a user's watch counts into implicit relative ratings, and then compute a "like" vector and a confidence vector. Implicit CF-NADE is constructed by

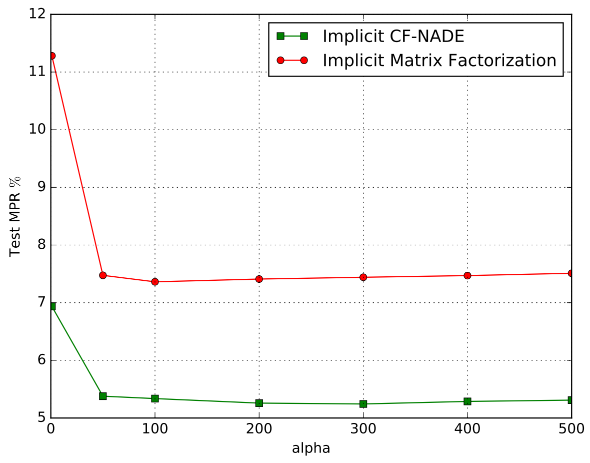

Figure 2: Test MPRs for implicit CF-NADE and implicit matrix factorization w.r.t α in Equation 6.

modifying CF-NADE to be aware of the "like" and confidence vector, in the sense that the joint probability of a "like" vector conditioned on the confidence vector is decomposed into conditionals by chain rule, and the conditionals are modeled by a series of weight sharing neural networks. We augmented the training loss with the confidence vector, taking the uncertainty of ratings into consideration. Experimental results show that implicit CF-NADE outperforms implicit matrix factorization on a dataset extracted from a popular digital TV streaming service. This also indicates that implicit CF-NADE is highly effective in real-world applications.

7. REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[3] G. K. Dziugaite and D. M. Roy. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015.

[4] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[5] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[7] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.

[8] C. C. Johnson. Logistic matrix factorization for implicit feedback data. *Advances in Neural Information Processing Systems*, 27, 2014.

[9] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37,

2009.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] Y. Li, J. Hu, C. Zhai, and Y. Chen. Improving one-class collaborative filtering by incorporating rich user information. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 959–968. ACM, 2010.

[12] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. Mllib: Machine learning in apache spark. *arXiv preprint arXiv:1505.06807*, 2015.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[14] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[15] X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 497–506. IEEE, 2011.

[16] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.

[17] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

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

[19] S. Sedhain, A. Menon, S. Sanner, and D. Braziunas. On the effectiveness of linear models for one-class collaborative filtering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, February 2016.

[20] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 111–112. International World Wide Web Conferences Steering Committee, 2015.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[22] B. Uria, I. Murray, and H. Larochelle. A deep and tractable density estimator. *JMLR: W&CP*, 32(1):467–475, 2014.

[23] Y. Yao, H. Tong, G. Yan, F. Xu, X. Zhang, B. K. Szymanski, and J. Lu. Dual-regularized one-class collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 759–768. ACM, 2014.

[24] Y. Zheng, B. Tang, W. Ding, and H. Zhou. A neural autoregressive approach to collaborative filtering. *arXiv preprint arXiv:1605.09477*, 2016.

[25] Y. Zheng, Y.-J. Zhang, and H. Larochelle. A deep and autoregressive approach for topic modeling of multimodal data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PP(99):1–1, 2015.