Hybrid Recommender System based on Autoencoders: Integrating together users and items Metadata

Sagi Eden
Beer Sheva, Israel
sagiede@post.bgu.ac.il

Omer Nizri
Beer Sheva, Israel
omerniz@post.bgu.ac.il

ABSTRACT

Proficient Recommender Systems (RS) heavily rely on Matrix Factorization (MF) techniques. MF aims at reconstructing a matrix of ratings from an incomplete and noisy initial matrix; this prediction is then used to build the actual recommendation. Simultaneously, Neural Networks (NN) met tremendous successes in the last decades but few attempts have been made to perform recommendation with autoencoders. Furthermore, few attempts even incorparated side-information with autoencoders for RS.

In this paper, we gather the best practice from the literature to achieve this goal. We first highlight the link between these autoencoder based approaches and MF. Then, we refine the training approach of autoencoders to handle incomplete data. Second, we extend [24] Hybrid Algorithm, to include both item's and user's side-information simultaneously. Finally, we empirically evaluate these approaches on the MovieLens and Amazon reviews data-set. Our results shows that incorporating both side-information to autoencoders models trained with small amount of data, Might damage the results in total. But on Cold-items the results are similar.

INTRODUCTION

Recommendation systems (RS) advise users on which items (movies, musics, books, etc.) they are more likely to be interested in. A good RS may dramatically increase the amount of sales of a firm or retain customers. For instance, 80% of movies watched on Netflix come from the RS of the company [6]. One efficient way to design such algorithm is to predict how a user would rate a given item. Two key methods coexist to tackle this issue: Content-Based Filtering (CBF) and Collaborative Filtering (CF).

CBF uses the user/item knowledge to estimate a new rating. For instance, user information can be the age, gender, or graph of friends etc. Item information can be the movie genre, a short description, or the tags. On the other side, CF uses the ratings history of users and items. The feedback of one user on some items is combined with the feedback of all other users on all items to predict a new rating. For instance, if someone rated a few books, Collaborative Filtering aims at estimating the ratings he would have given to thousands of other books by using the ratings of all the other readers. CF is often preferred to CBF because it wins the agnostic vs. studied contest: CF only relies on the ratings of the users while CBF requires advanced engineering on items to well perform [15].

The most successful approach in CF is to retrieve potential latent factors from the sparse matrix of ratings. Book latent factors are likely to encapsulate the book genre (spy novel, fantasy, etc.) or some writing styles. Common latent factor techniques compute a low-rank rating matrix by applying Singular Value Decomposition through gradient descent [10] or Regularized Alternating Least Square algorithm [30]. However, these methods are linear and cannot catch subtle factors. Newer algorithms were explored to face those constraints such as Factorization Machines [17].

More recent works combine several low-rank matrices such as Local Low Rank Matrix Approximation [12] or WEMAREC [3] to enhance the recommendation. Another limitation of CF is known as the cold start problem: how to recommend an item to a user when no rating exists for either the user or the item? To overcome this issue, one idea is to build a hybrid model mixing CF and CBF where side information is integrated into the training process. The goal is to supplant the lack of ratings through side information. A successful approach [1, 16] extends the Bayesian Probabilistic Matrix Factorization Framework [20] to integrate side information. However, recent algorithms outperform them in the general case [13]. In this paper we introduce a CF approach based on Stacked Denoising Autoencoders [26, 31] which tackles both challenges: learning a non-linear representation of users and items, and alleviating the cold start problem by integrating side information. Compared to previous attempts in that direction [21, 23, 25, 4, 28, 14], our framework integrates the matrix of ratings and side information in a unique Network. Comparing previous attempts to combine side information to auto-encoder based models [29, 24, 7], We are integrating both user's and item's side information in same model. Models which integrates side information leads to a scalable and robust approach which beats state-of-the-art results in CF.

As stated, Our model extends the work has been done in [24]. We are using Denoising Autoencoders for learning the hidden representations of the users/items. Similar to [24], we provide 2 versions, U-CFN (user-based), and I-CFN (item-based) models. in U-CFN, the model concatenate the user side information to the hidden layer of the auto encoder. The

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

extension we provide in this work, is integrating in the U-CFN model the items side information to the decoding weights (in I-CFN we concatenate the item side information to the hidden layer and the users side information we integrate to the denoising weights respectively). Our motivation is to improve the accuracy mostly on cold items.

The paper is organized as follows. First, Sec. 2 presents state of the art on related approaches. Then, our model is described in Sec. 3. In Sec. 4. We describe the Experiments we did, the evaluation method and matrices we used. Finally, experimental results are given and discussed in Sec. 5 and Sec. 6. In Sec. 7 we concludes the paper. For source code visit our project in github.

RELATED WORK

Latent Factor Models

The use of Latent Factor Models for CF became popular after their success in the Netflix competition [2] . These methods, such as SVD [22] and BPR [18], model each user-item relation in a low-dimensional vector, so that the users and items are represented in a hidden latent space. Usually, it is implemented by holding an embedded matrix for the users and embedded one for the items. By multiplying these matrices we get the value for the user-item (can be the predicted rating, or how much the user prefer this item in relation to other items, etc.). More specifically, Given N users and M items, we denote r_{ij} the rating given by the i^{th} user for the j^{th} item. It entails an incomplete matrix of ratings $R \in R^{N*M}$. MF aims at finding a rank k matrix $\hat{R} \in R^{N*M}$ which matches known values of R and predicts unknown ones. Typically, $\hat{R} = UV^T$ with $U \in R^{N*M}$, $V \in R^{M*K}$.

As CF methods became popular it used mainly explicit data [22] one of the major challenges with CF methods is the sparsity of the data, i.e, most of the users don't use most of the items, and when they do they are seldom leaving feedback and rate items. Therefore many methods started to include implicit data for better results. for instance Y. Koren suggested SVD++ [9]. An algorithm which extends the traditional SVD [22] for rating prediction task. SVD++ considering all items for which the user made implicit feedback. It further considers all explicit ratings also as implicit feedback interactions.

Auto Encoders

Recently, autoencoders have been used to handle CF problems [23, 25]. An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. As stated in Kramer's paper [11]. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal "noise". Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input, hence its name. Several variants exist to the basic model, with the aim of forcing the learned representations of the input to assume useful properties. Examples are the regularized autoencoders (Sparse, Denoising and Contractive autoencoders) [26, 19, 8], proven effective in learning representations for subsequent classification tasks, and Variational autoencoders, with their recent applications as generative models. In this paper we will be used Denoising Auto encoders, as their results seems to fit the recommendation system domain before [28, 24].

Auto Encoders introduced to recommendation systems domain before with Autorec [23] and CDL [28]. The former Solving rating prediction task. The paper purposed user-based model and item-based model. The latter proposed a novel model based on DAE for solving ranking task. The model proposed integrates users' factors for solving Top-N recommendations. Both papers made an Adjustment in AE for RS: The model gets as input instead of raw data/ features is a ratings vector (user-based model vector is the user ratings of all items, symmetric for item-based model), dense it and try to reconstruct it' as can see in fig 1.

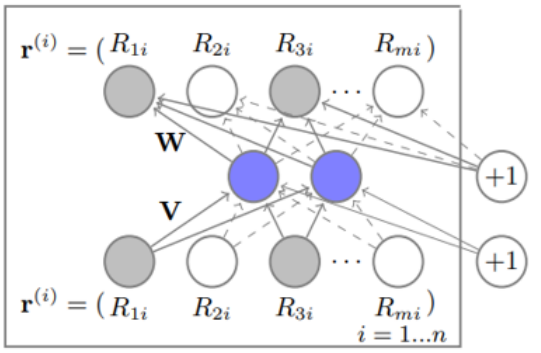

Figure 1. CDAE Architecture

In the learning phase AutoRec also introduced masking technique for handling the unknown values such that the model learns only from known values (ratings) 2.

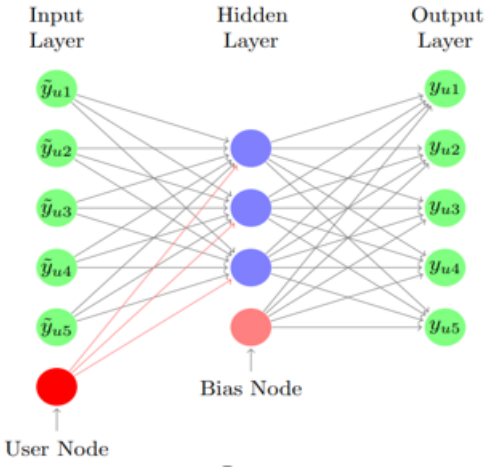

Figure 2. Autorec Architecture

Other forms of autoencoders have also been suggested for recommendation tasks. AutoSVD++ [29], A recent study that combines constructive autoencoders and matrix factorization to provide recommendations based on content data and implicit user feedback. CDL [27] A hierarchical Bayesian model

which integrates stacked denoising autoencoder (SDAE) into probabilistic matrix factorization . CVAE [14] A variational autoencoder that learns the deep latent representation and the implicit relationship between users and items from ratings and content data. Our Method Differ from these algorithms as it integrate both users and items self information into one DAE Model.

Denoising Auto Encoders

Recent work in Deep Learning advocates to stack pretrained encoders to initialize Deep Neural Networks [5]. This process enables the lowest layers of the Network to find low dimensional representations. It experimentally increases the quality of the whole Network. Yet, classic Autoencoders may degenerate into identity Networks and they fail to learn the latent relationship between data. [26] tackle this issue by corrupting inputs, pushing the Network to denoise the final outputs. One method is to add Gaussian noise on a random fraction of the input. Another method is to mask a random fraction of the input by replacing them with zero. In this case, the Denoising Auto-Encoder (DAE) loss function is modified to emphasize the denoising aspect of the Network.

A recent paper (September 2018) published integrated both user and item self information in DAE [7] . Our Improvement Differ from that paper in 2 Things. First, the Model Suggested by Y. Jhamb uses the features as context and use it as Attention for user's hidden representation vector. Second, The Model choose for each user subset of items from past user interaction to extract features from or integrate general context for item, while our suggestion takes into account all of the item's self information. The model architecture can be viewed in fig 3.

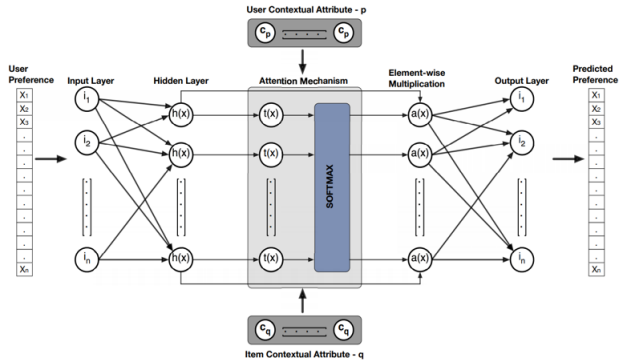

Figure 3. Attentive Contextual DAE Architecture

Hybrid Recommender System based on Auto-encoders

The algorithm [24] suggested 2 types of DAE Models for Rating Prediction. U-CFN (users based) and V-CFN (items based). both models integrated the side-information as concatenation in each layer at the DAE, while U-CFN integrated only users side information and V-CFN integrated only items side information.

In the paper, First, the authors design a training process to predict missing ratings from incomplete vectors. Secondly, they extended CF techniques using side information to auto-encoders to improve predictions for users/items with few ratings.

Handling with missing data

MC tasks introduce two major difficulties for auto-encoders. The training process must handle incomplete input/target vectors. The other challenge is to predict missing ratings as opposed to the initial goal of auto-encoders to rebuild the initial input. The authors handle those obstacles by two means.

First, They inhibit input and output nodes corresponding to missing data. For input nodes, the inhibition is obtained by setting their value to zero. To inhibit the back-propagated unknown errors, They used an empirical loss that disregards the loss of unknown values. No error is back-propagated for missing values, while the error is back-propagated for actual zero values.

Second, they shake the training data and design a specific loss function to enforce reconstruction of missing values. Indeed, basic auto-encoders loss function only aims at reconstructing the initial input. Such approach misses the point that CF goal is to predict missing ratings. Worse, there is little interest to reconstruct already known ratings. To increase the generalization properties of auto-encoders, they applied the Denoising Auto-Encoder (DAE) approach [26] to the CF task. The DAE approach corrupts the input vectors and lets the network denoise the outputs. The corresponding loss function reflects that objective and is based on two main hyper-parameters: α and β. They balance whether the network would focus on denoising the input (α) or reconstructing the input (β). In our context, the data are corrupted by masking a small fraction of the known rating. This corruption simulates missing values in the training process to train the auto-encoders to predict them. They also set $\alpha > \beta$ to emphasize the prediction of missing ratings over the reconstruction of known ratings.

The overall training process is described in 4.

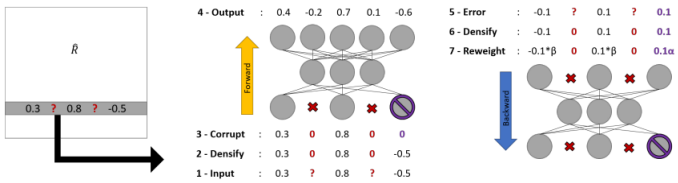

Figure 4. Training steps for auto-encoders with incomplete data. The input is drawn from the matrix of ratings, unknown values are turned to zero, some ratings are masked (corruption) and a dense estimate is obtained. Before back-propagation, unknown ratings are turned to zero error, prediction errors are re-weighted by α and reconstruction errors are re-weighted by β.

Integrating Side Information

CF only relies on the feedback of the users regarding a set of items. However, adding more information may help to increase the prediction accuracy and robustness. Furthermore, CF suffers from the cold start problem: when little information is available on an item, it will greatly lower the prediction accuracy. They integrated side information to CFN to alleviate

the cold start scenario. The integration applied by (i) appending side information to incomplete input vectors, (ii) injecting side information to every layer of the autoencoders as in 5.

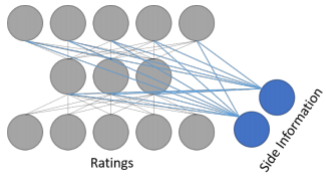

Figure 5. Side information is wired to all the neuron.

Firstly, Injecting side information to the last hidden layer enables to partially retrieve the error function of classic hybrid systems. Secondly, injecting side information to the other intermediate hidden layers enhance the internal representation. Finally, appending side information to the input vectors supplants the absence of input data when no rating is available. The autoencoder fits CF standards while being trained end-to-end by backpropagation

METHOD

In this section, we introduce our proposed 2 hybrid models, Users based, named U-CFN*, and items based, named I-CFN*. The models integrates both side information of users and items.

Our model extends [24], and the base architecture remains the same. assuming N users and M items in the system. For simplicity reasons, and because of the symmetry between the 2 models, we will explain our U-CFN architecture.

The model is Auto-encoder with M nodes in input, which each node correspond to respective item. the value of node i^th is the rating the user in the current iteration gives to item i in the train data, and 0 if the user didnt rate item i. Because of the sparsity, most of the nodes will get the value 0.

Furthermore, we used Denoising auto-encoders, Hence, we masked known ratings with zeros in input for avoiding regularization to the model. Side information of the user added as neurons in each layer, for getting stronger representation, same as done in [24]. For stronger representation for colde items, The item's side info is also added - it summed with the corresponding node weights, before the multiplication with the hidden layer neurons (user latent representation). Hence the final rating score for item i in iteration of user u will be calculated as: $(w_i + v_i) * h_u$ where w_i is the weights corresponding to node i. v_i is the side information of item i (same dimension as w_i). h_u is the hidden layer contains both the NN neurons from the auto-encoder layer and the concatenated user side info neurons. The model architecture can be viewed in 6. In our model we hope to get stronger representation for both, items and users, in the same model.

EVALUATION

To evaluate the model we've used the *Root Mean Square Error* (RMSE). We've compared the different models on 4 different datasets:

- Amazon Video Games dataset

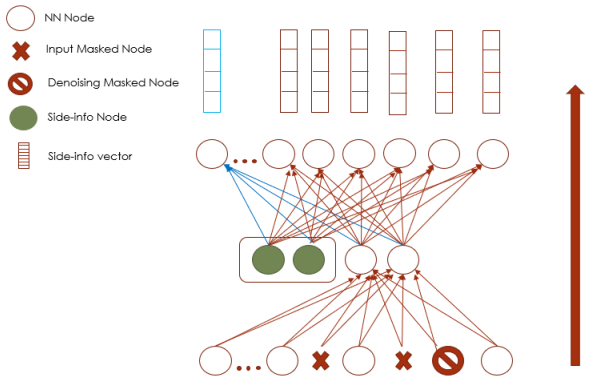

Figure 6. Our Method Architecture, the blue vector and blue weights are summed together before the multipication.

- Amazon Video Games dataset - cold start
- MovieLens 1M
- MovieLens 1M - cold start

The MovieLens1M is a datasets for movies rating by users, that contains ratings and side information for both users and movies. The side information for users include demographics information (age, occupation and gender, all 3 are categorical feautres), and the side information for movie contains the movie genres. There's total of 1M ratings in this dataset. The Amazon video games dataset is a subset of ratings and reviews for video games by users, who have rated at least 5 products, and only products with at least 5 reviews. This data set contains 231,780 reviews.

The side information features from amazon data-set extracted for users/items from the train data as follows: we grouped together reviews of each user and combined them together (we used only reviews from the train data-set), lets define it as document. After that we used TF-IDF on all of the user's "documents" and reduced the dimensions with SVD. The result was features vector for each user. We did the same process while grouping item's reviews together for getting the item's side-information. was TF-IDF on all of the reviews in the train data-set.

In order to get a good evaluation of the results, our loss function was adapted to be masked, so that the loss will only be calculated for entries that we now their respective expected rating. i.e. records that doesn't appeared in the test set are not includes in the RMSE calculation, since it's not possible to assess their expected rating. In this paper we compared our algorithm denoted by $*$ to the standard collaborative filtering neural network, denoted by u-cfn for users vectors and i-cfn for items vectors. We've used $++$ to denote the models that contains only one kind of side-information. To compare our algorithm to more naive baseline to see whether the complex models actually improve result, we append to the comparison of the results a popularity based model as well. The popularity based model learn the mean rating for each user based on the

train set, and it evaluates each rating for each user to be the respective mean.

RESULTS

Table 5 shows the RMSE for each algorithm and dataset. As we can see, our algorithm didn't improved U-CFN++ neither I-CFN++. For each dataset, the best record appears in bold.

In *MovieLens-1M* dataset, we have a very limited side information regarding the users and the products and it's hard to have a significant benefit over not using side-info at all. However, in the situation of cold start the improvement are more significant. While the results don't show any improvement by utilizing the side information of the user and the products together, there is still a need to evaluate the algorithm on bigger datasets with more side information and more records, and to consider other methods to combine those two kinds of side information.

One can see that for *Amazon Video Games* dataset, the results using side information are not superior. One reason for that could be that the side-information being used is tf-idf vector to represent the centroid review. Such type of information is naturally sparse and requires more samples for learning than categorical data like in MovieLens-1M.

We can see in figure 7 the Test-RMSE on MovieLens-1M, It's clear from the graphs that our method is longer to converge and perform worse. However, we can see in figure 8 that our algorithm is more beneficial for cold start.

In figure 9 we can see that using side information was inferior to regular DAE, but not inferior to popularity measure. However, in the case of cold start evaluation (figure 10, we can see that the side information of the user improved the model.

In all the cases, it's clear that all the models being evaluated surpassed the popularity based model, except to the items based models in the case of amazon video games cold start (see 10.

DISCUSSION

While this paper doesn't proved our method to be beneficial, there is still a need to conduct better evaluation. It's clear from the research that, especially for cold start, it's better to use side-information. It's logical to hypothesize that combining both kind of side-information will improve results. However, in the case of the current research, the model was quite complex relative to the amount of available data. In the case of movielens, we can suggest the claim that I-CFN models provide better results because their respective (main) side-information is the genres of movies. i.e., some movies genres are more popular than others. While U-CFN used user features like age that might be less correlated with movies rating, at least when not being evaluated for specific movie. A further study that research the demographics of users in the dataset and the different ratings based on genres, might prove or disprove these claims.

CONCLUSIONS AND FUTURE WORK

To conclude, one can see that using side-information is essential method to improve recommendation systems, especially in the case of cold start. Future work might investigate our algorithms on bigger datasets or on datasets with more descriptive features as side-information. Also, future work might suggest other ways to combine the different kinds of side-information in the same model in ways that will be less noisy.

REFERENCES

[1] Ryan Prescott Adams, George E Dahl, and Iain Murray. 2010. Incorporating side information in probabilistic matrix factorization with gaussian processes. *arXiv preprint arXiv:1003.4944* (2010).

[2] James Bennett, Stan Lanning, and others. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. Citeseer, 35.

[3] Chao Chen, Dongsheng Li, Yingying Zhao, Qin Lv, and Li Shang. 2015. WEMAREC: Accurate and scalable recommendation through weighted and ensemble matrix approximation. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 303–312.

[4] Gintare Karolina Dziugaite and Daniel M Roy. 2015. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443* (2015).

[5] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[6] Carlos A Gomez-Uribe and Neil Hunt. 2015. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)* 6, 4 (2015), 1–19.

[7] Yogesh Jhamb, Travis Ebesu, and Yi Fang. 2018. Attentive contextual denoising autoencoder for recommendation. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*. 27–34.

[8] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[9] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

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

[11] Mark A Kramer. 1991. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal* 37, 2 (1991), 233–243.

[12] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *International conference on machine learning*. 82–90.

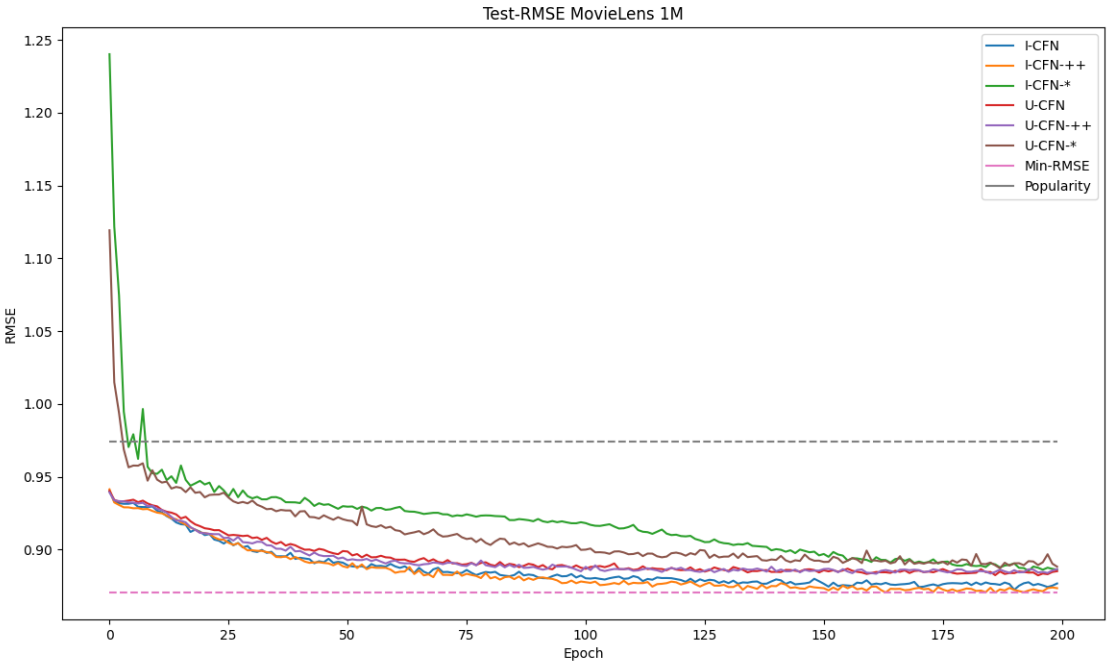

Figure 7. MovieLens-1M - algorithms comparison

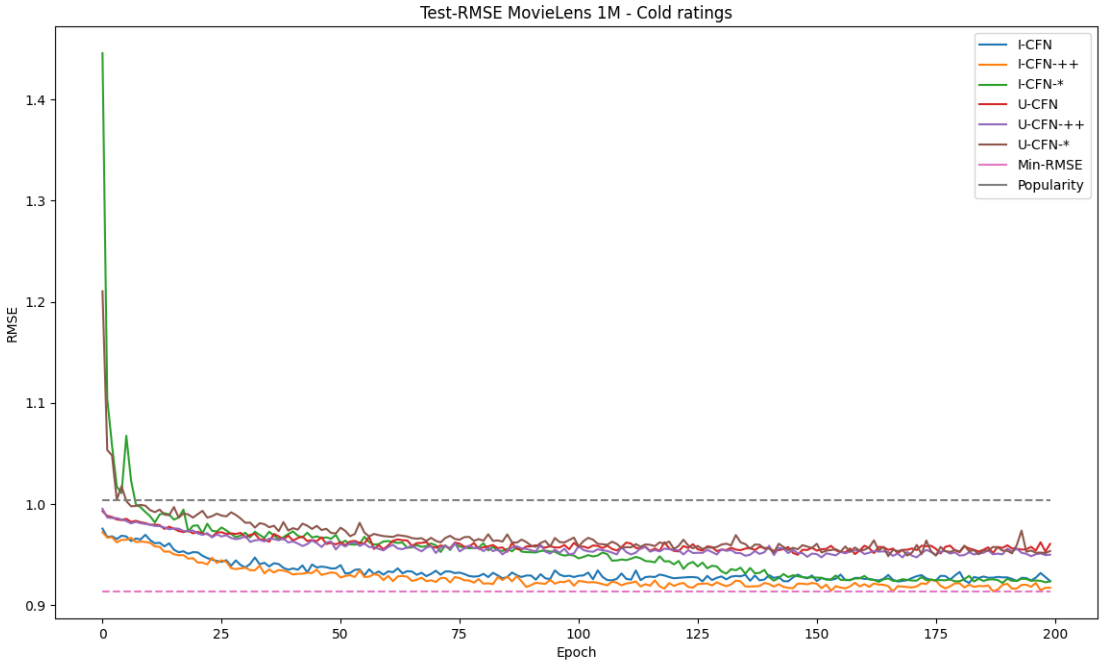

Figure 8. MovieLens-1M cold start algorithms comparison

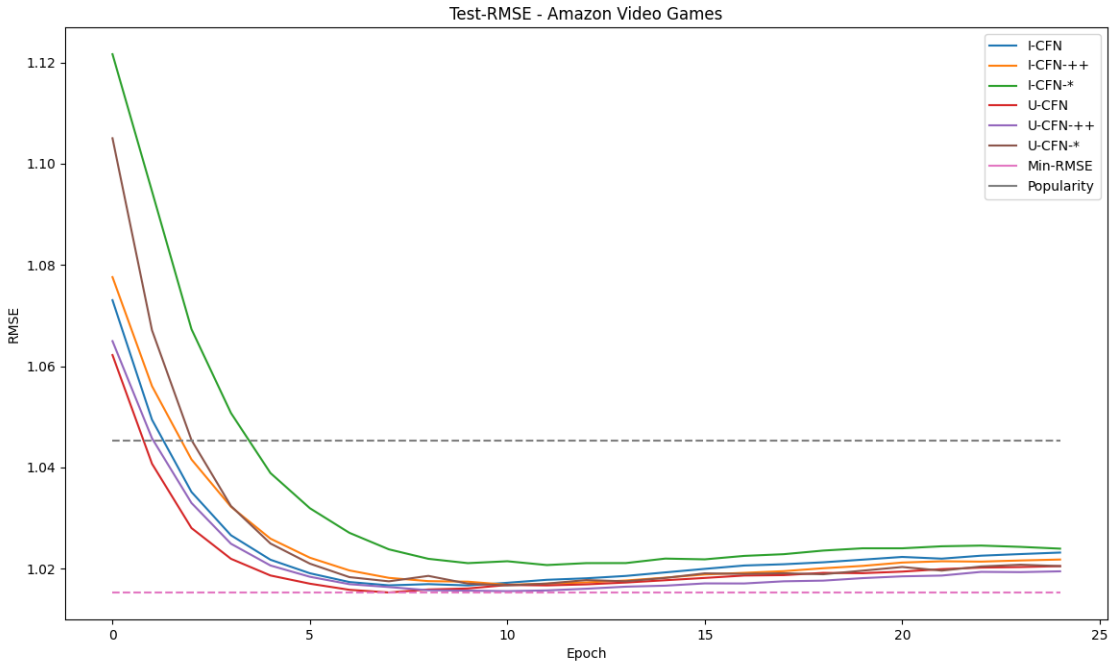

Figure 9. Amazon video games - algorithms comparison

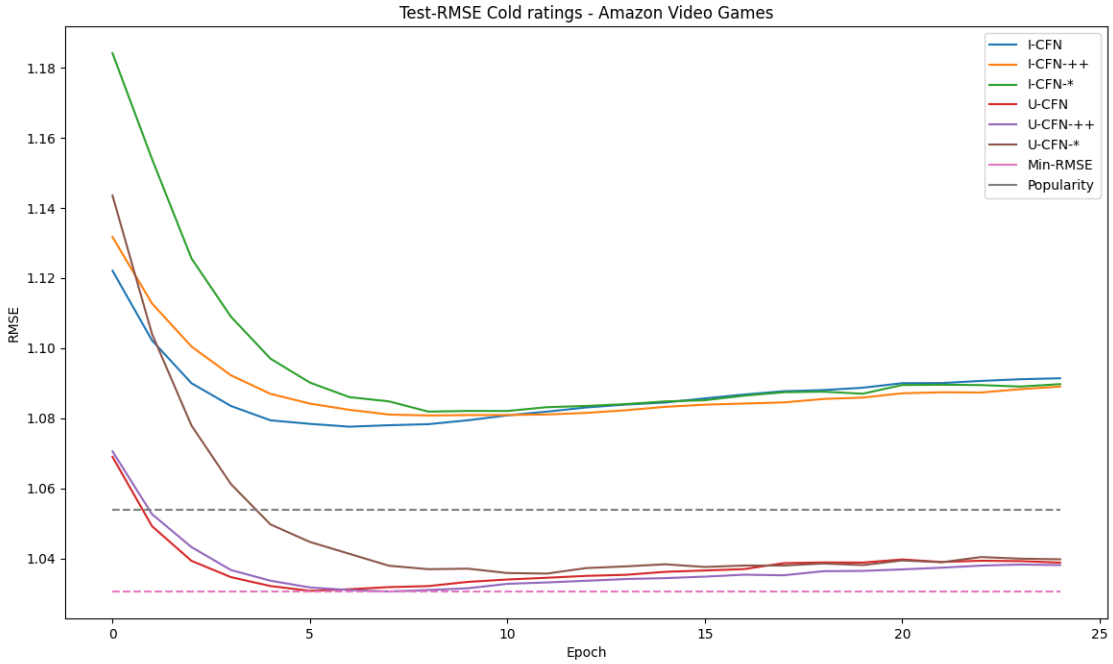

Figure 10. Amazon video games - cold start algorithms comparison

Dataset	U-CFN	U-CFN++	U-CFN*	I-CFN	I-CFN++	I-CFN*	Popularity
MovieLens1M	0.8823	0.8834	0.8872	0.8724	**0.8706**	0.8854	0.9744
MovieLens1M-Cold start	0.9498	0.9472	0.9485	0.9219	**0.9135**	0.9219	1.004
Amazon Video Games	**1.0152**	1.0155	1.0166	1.0166	1.0168	1.0206	1.0453
Amazon Video Games - Cold Start	1.0308	**1.0306**	1.0357	1.0776	1.0808	1.0819	1.054

Table 1. Results Summary

[13] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. 2012. A comparative study of collaborative filtering algorithms. *arXiv preprint arXiv:1205.3193* (2012).

[14] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 305–314.

[15] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*. Springer, 73–105.

[16] Ian Porteous, Arthur Asuncion, and Max Welling. 2010. Bayesian matrix factorization with side information and dirichlet process mixtures. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[17] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[18] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[19] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *Icml*.

[20] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*. 880–887.

[21] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*. 791–798.

[22] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth international conference on computer and information science*, Vol. 27. Citeseer, 28.

[23] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*. 111–112.

[24] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 11–16.

[25] Florian Strub and Jeremie Mary. 2020. Collaborative filtering with stacked denoising autoencoders and sparse inputs. (2020).

[26] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* 11, 12 (2010).

[27] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1235–1244.

[28] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 153–162.

[29] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. AutoSVD++ An Efficient Hybrid Collaborative Filtering Model via Contractive Auto-encoders. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 957–960.

[30] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. 2008. Large-scale parallel collaborative filtering for the netflix prize. In *International conference on algorithmic applications in management*. Springer, 337–348.

[31] Fuzhen Zhuang, Dan Luo, Xin Jin, Hui Xiong, Ping Luo, and Qing He. 2015. Representation learning via semi-supervised autoencoder for multi-task learning. In *2015 IEEE International Conference on Data Mining*. IEEE, 1141–1146.