DÆM: Denoising Autoencoder Model for Collaborative Filtering

Yannick Schmid, Loïc Holbein, Matthew Weingarten
Group: DÆM
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Highly accurate Recommender Systems, including Collaborative Filtering, lie at the heart of a satisfactory customer experience and continuous user engagement for a plethora of large-scale online platforms. While Matrix Factorization is the most widely studied and applied Collaborative Filtering approach, there is evidence to suggest that linear techniques lack the complexity to sufficiently capture the underlying relationship between users and items. The use of neural networks like Autoencoders offers a potential remedy and may more accurately represent this relationship. In this work, we propose our Denoising Autoencoder Model (DÆM) for highly accurate Collaborative Filtering and show improvement over four evaluated state-of-the-art models.**

I. Introduction

Recommendations from a trusted source, akin to "Word of Mouth" recommendations from a friend, command profound influence over consumer behavior [1], [2], [3], [4]. Now imagine an omniscient friend who always correctly predicts one's satisfaction with a product. In the advent of information overload, recommender systems attempt to come as close as possible to this unachievable gold standard. They try to model the relationship between the value generated for a *user* by a particular *item*. An item typically corresponds to either a movie, music, or retail product and is used by companies like Netflix, Amazon, and Google. Specifically, *Collaborative Filtering* (CF) builds this representation from past rating behavior from all users and subsequently predicts the rating of an unseen user-item pair.

Autoencoders are neural networks designed to represent high-dimensional data with low-dimensional codes. They have shown promising results in dimensionality reduction [5]. This indicates the use of Autoencoders has high potential for application in CF and has indeed already attracted attention in literature. Therefore, we propose a *Denoising* Autoencoder Model (DÆM) to robustly describe and extract an underlying relationship between users and items, from which we can extract predictions for unseen user-item pairs.

This work contributes the following:

- Implementation of a novel denoising Autoencoder approach for Collaborative Filtering.
- Tuning and evaluation of hyperparameters to minimize prediction error.
- An evaluation of our approach in comparison to state-of-the-art baseline implementations.

II. Background

Literature on Collaborative Filtering can be broken down into three most common techniques, namely Memory-based, Model-based, and Neural-based [2], [6]. Some practical implementations combine them.

A. Memory-based

This type of system uses statistical methods, commonly user or item similarity metrics, e.g. adjusted cosine similarity [7], to find a set of k-nearest-neighbors. The set is then aggregated by a similarity weighted mean to compute the rating prediction, the goal of CF. While this method is simple, it can suffer from poor performance, especially when dealing with high levels of sparsity [8].

B. Model-based

On the other hand, Model-based CF produces a model from the rating matrix offline to generate recommendations [9]. Model-based techniques include Matrix Factorization, which has been the de facto standard approach since the Netflix prize in 2007[10]. MF assigns vectors of latent features to users and items, i.e. find matrix U and V such that $A \approx UV^T$, A being a rating matrix. MF techniques include Singular Value Decompositions (SVD), Alternating Least Squares (ALS) [11], or Sparse Linear Methods (SLIM) [12].

C. Neural-based

Neural-based approaches, such as neural CF [6], sometimes classified as a subset of Model-based techniques, aim to address one of the major criticisms of previous MF techniques, namely the linear nature of the dot product used to compute a prediction for an unseen value. It may not be expressive enough to capture the principal relationship between users and items, hence Neural-based CF approaches replace the dot product of latent feature vectors with neural networks [6], [13].

III. Model

Collaborative filtering boils down to the following formulation: A set of users $U = \{u_1, ..., u_n\}$ and a set of items $I = \{i_1, ..., i_m\}$ form a rating matrix $A \in \mathbb{R}^{n \times m}$ containing numerical values $a_{u,i}$ (representing some kind of metric about interaction or (dis)satisfaction) for every observed rating of item i by a user u. The goal is to fill

in some or all of the unobserved values. We refer to the boolean matrix $\Omega \in \{0,1\}^{n \times m}$, with $\omega_{u,i} = 1$ if user u has rated/interacted with item i and 0 otherwise, as the observation matrix.

Some approaches to solving CF make use of additional context-sensitive information (such as movie genres or keywords, further non-rating user data, etc) [14], [15]. However, this is beyond the scope of this work where we only evaluate and implement context-insensitive approaches.

Our user-based denoising Autoencoder (DAE) finds a low-dimensional representation for the rating relationship between users and items. An Autoencoder encodes an input vector, in our case row A_u of matrix A, into a low dimensional representation, and subsequently decodes it back such that it is close to the input. We define our Autoencoder model $\mathcal{M}$:

$$\hat{A}_u = \mathcal{M}(A_u, \Omega_u) = dec(enc(A_u, sparse(\Omega_u))) \quad (1)$$

with dec and enc being a decoding (dimension expanding) and an encoding (dimension reducing) neural network respectively. The function $sparse$ is explained in the next subsection. Our model aims to minimize the following optimization problem:

$$\min_{\theta} \sum_{u \in U} \|\Omega_u \odot (A_u - \mathcal{M}_\theta(A_u, \Omega_u))\|_2^2 \quad (2)$$

with $\odot$ denoting element-wise multiplication.

A. Denoising

Our approach adds artificially induced noise to the input data, hence is classified as a *denoising* Autoencoder [16]. We set randomly sampled values of Ω to zero, i.e. further sparsifying the matrix before feeding it to the encoder. In equation 1 this is shown with the function called $sparse$, which sets each entry of its input to zero with independent probability r.

This approach nicely fits the Collaborative Filtering framework, as its input is sparse/noisy to begin with, and we can justify it by the following natural assumption: a model which makes reasonable predictions on a certain dataset is also capable of making good inferences on a less sparse one.

Further, having artificial sparsification implies that we have access to a ground-truth rating for the dropped entries. Let $\mathcal{I}_u$ be the indices of these entries for user vector u, i.e. $\mathcal{I}_u = \{i \mid (\Omega_u \odot (1 - sparse(\Omega_u)))_i = 1\}$. Over this $\mathcal{I}_u$ we define our mean squared error (MSE) [17] based loss $\mathcal{L}$:

$$\mathcal{L}(A_u, \hat{A}_u) = \frac{\sum_{i \in \mathcal{I}_u} (A_u - \hat{A}_u)_i^2}{|\mathcal{I}_u|} \quad (3)$$

Intuitively, we calculate the MSE only on values that were set to zero during the dropout stage by the $sparse$ function. The equation shows the loss for one user vector. During training with mini-batches, the average is taken with respect to all the $\mathcal{I}_u$s of the batch.

Refer to figure 1 for a schematic system overview.

B. Model parameters

We characterize the shape of our neural networks using two hyper-parameters: the *depth* (D) and the *width* (W). The *depth* is the number of hidden layers in the encoder. We only consider symmetric encoder and decoder shapes, as such, it is equal to the total number of hidden layers minus one divided by two. Additionally, we use the activation function ReLU.

The *width* on the other hand is the number of nodes in the middle layer, i.e. the dimension bottleneck and thus the expressiveness of the model. The encoder halves the number of units on each propagation step, and the decoder doubles them. For example *depth* $= 2$ and *width* $= 4$ results in a total of 5 hidden layers with sizes 16,8,4,8 and 16 respectively.

The sparsification pipeline requires another two hyper-parameters: the probability r of zeroing an entry in $sparse$ and what we call the dropout adjustment strategy. We formulate and explore three different strategies: *standard*, *effective*, and *renormalize*. *Standard* multiplies each remaining rating by $1/(1-r)$. For *effective* we do something similar: we multiply by $1/(1-t)$ where t is the effective dropout rate: the fraction of entries that are either unobserved or dropped. Finally, *renormalize* z-transforms $A_u \odot sparse(\Omega_u)$ to bring the row (back) to mean 0 and standard deviation 1. The performance comparison is discussed in section IV.

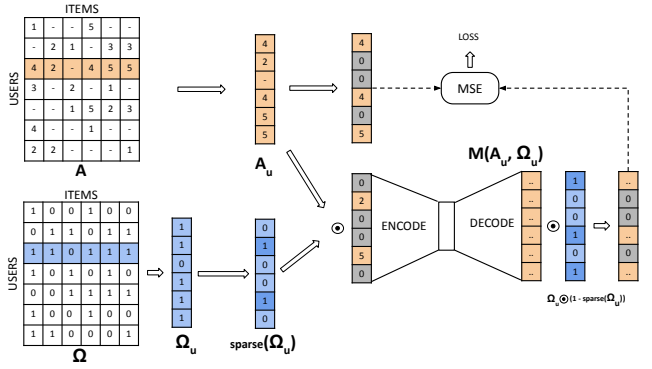

Figure 1: Scheme of our denoising Autoencoder model with a toy rating matrix (top left) and a toy observation matrix (bottom left).

IV. Results

A. Setting

Our dataset contains 10000 users and 1000 items, ratings are given as integers from 1 to 5, and has a sparsity of 88%. No further contextual information on the items or users is available. For all models, we z-normalize the data across users to adjust for biased user scales.

This is the only preprocessing performed and we do not explicitly address the data distribution's top-heaviness, i.e.

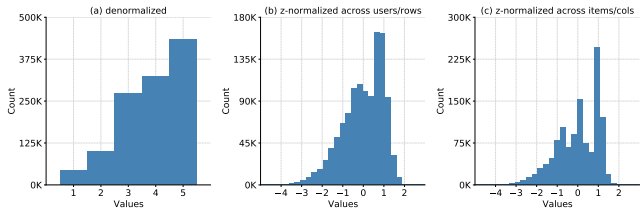

Figure 2: Distribution of training data ratings.

higher integer values occurring more often (see figure 2), as it might be a property specific to our dataset and not a general trend in Collaborative Filtering.

B. Evaluation Metric

For evaluation we use the root mean squared error (RMSE) [17]. It is defined as:

$$RMSE = \sqrt{\sum_{(u,i)\in\mathcal{J}} \frac{(\hat{a}_{u,i} - a_{u,i})^2}{|\mathcal{J}|}} \tag{4}$$

with $\mathcal{J}$ being a set of indices (u, i) for which the true value $a_{u,i}$ in the matrix A is known, and $\hat{a}_{u,i}$ being a prediction for that true value.

All RMSE values, unless otherwise stated, are computed over a randomly selected holdout-set containing 10% of all observed entries of the original input matrix.

C. Model Selection

Additional to the four hyper-parameters discussed in section III-B we introduce N as the number of bootstraps in a simple bagging approach [18]. We run the train and prediction pipeline of the model N times and take the average predicted rating as the final output. We don't split the training data explicitly into multiple bootstrap sets as would be the usual approach. In our case randomly zeroing entries from the input vectors creates different learning data sets implicitly.

Experiments where we use a more standard approach of bagging, i.e. creating N bootstrap datasets by drawing random rows of the matrix with replacement, showed worse performance and are omitted.

Table I shows the performance measured in RMSE with varying network shapes. The best-performing tested network has $D = 1$, $W = 16$ and $N = 8$. We observe slight overfitting, i.e. an increase in RMSE, across the board when going from 400 to 1000 epochs.

To look at the impact of the dropout adjustment strategies we only consider the aforementioned best-performing model configuration over 400 epochs with $r = 0.5$. An experiment rerunning that model with different strategies shows an RMSE (averaged over 5 runs) of 0.981580 for *effective*, 0.996017 for *renormalize*, and 0.978601 for *standard*. Despite its simplicity, *standard* shows the best performance.

		T=400		T=1000	
		N=1	N=8	N=1	N=8
D	**W**	**RMSE**	**RMSE**	**RMSE**	**RMSE**
0	4	0.985462	0.985257	0.984707	0.984261
	8	**0.981922**	0.980909	**0.982014**	0.981679
	16	0.982301	0.980006	0.982646	0.979875
1	4	0.986735	0.984824	0.986729	0.984459
	8	0.982721	0.980125	0.982260	0.979585
	16	0.985756	**0.977757**	0.987778	**0.978434**
2	4	0.986411	0.984980	0.986939	0.985130
	8	0.986529	0.981728	0.990841	0.981730
	16	0.993193	0.980765	1.001669	0.981352

Table I: Performance with varying network shape. T represents the number of training epochs. Dropout probability r is 0.5 and the strategy is *standard*. The score shown is the mean RMSE aggregated over 5 runs. Minimum RMSE per column in bold.

The results of an investigation into the dropout probability r are shown in figure 3 (a). We plot the performance of our model with three different D, W, and T configurations. With higher r the performance generally gets worse, while with low r the different models show varying results. We consider $r = 0.3$ and $r = 0.4$ to be a general sweet spot: there the RMSE is reasonably low and predictable.

In comparison figure 3 (b) shows the same scenario without bagging. Here higher r configurations mostly outperform their lower counterparts. This leads us to the assumption that low to medium dropout rates induce low bias but increased variance models, which as weak learners significantly profit from bagging.

Further, we briefly discuss an alternative formulation of our model's loss: we take the MSE over all observed ratings of the corresponding vectors instead of only the not dropped ones, i.e. equation 3 with $\mathcal{I} = \{i \mid (\Omega_u)_i = 1\}$. This change allows us to run experiments with $r = 0$ and hence gives us a model which is more similar to U-AutoRec [19] a model without any additional sparsification, which is further discussed in section V.

We see the resulting RMSEs plotted in figure 3 (c). The change leads to a more predictable outcome. As in figure 3 (b) higher dropout probability results in lower RMSE. But the alternative loss generally performs worse. The RMSE gets comparable for high r values, but we conjecture that high dropout rates get problematic for sparser data-sets as they may lead to empty input vectors $sparse(\Omega_u)$. Further, with r tending to 1 the two loss formulations become equivalent, as $\Omega_u \odot (1 - sparse(\Omega_u))$ gets closer and closer to Ω_u.

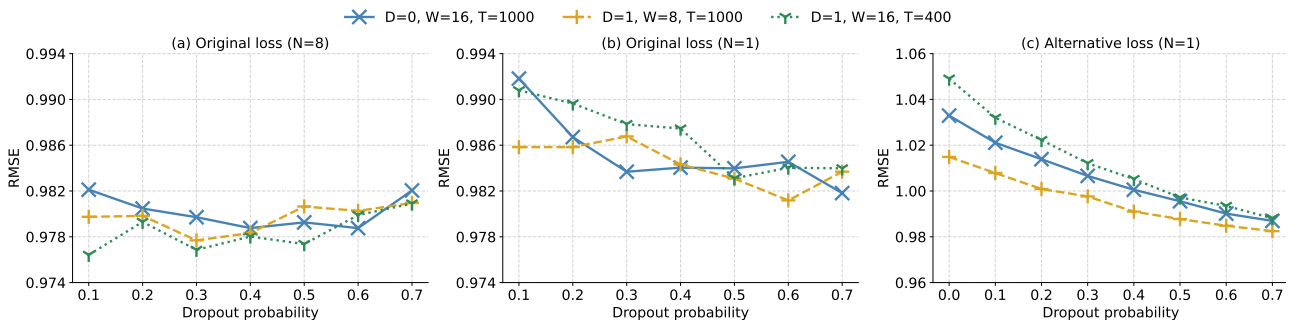

Figure 3: Performance analysis under varying dropout probabilities r using strategy *standard*. Model parameters are specified in the legend. Subfigures (a) and (b) use standard loss function: (a) with bagging ($N = 8$), (b) without ($N = 1$). Subfigure (c) shows performance with an alternative loss function.

D. Comparison

We compare DÆM to different baseline algorithms. The following list shows their names and their considered specification during a grid-search.

- SVD: imputation of row mean and reconstruction using $k \in \{1, 2, 3, 4, \mathbf{5}\}$ largest singular values.
- ALS [11]: *number of hidden features* in $\{2, \mathbf{3}, 4, 5\}$, $\lambda \in \{0.05, 0.1, \mathbf{0.2}\}$
- NCF [6]: **GMF**, MLP and the fusion variants considered, *predictive factors* in $\{\mathbf{4}, 8, 16, 32\}$
- SLIM [12]: ℓ_1 term in $\{0.5, 1, 2.5, 5, 10, \mathbf{25}, 50, 75\}$

The best performing hyper-parameter values are highlighted in bold, and the actual RMSE comparison can be seen in table II. For DÆM we choose $D = 0$, $W = 8$, $N = 1$, and $T = 400$, as it is the lowest RMSE seen in table I not using bagging (other models don't incorporate any ensemble methods). Our model outperforms the baseline configurations tested. ALS shows the next lowest RMSE and the only other approach utilizing neural networks NCF takes third.

SVD	ALS	NCF	SLIM	DÆM
1.0103	0.9906	0.9998	1.0192	**0.9819**
± 0.0030	± 0.0004	± 0.0033	± 0.0005	± 0.0021

Table II: RMSE values, including standard deviation below, averaged over 5 runs (except ALS only over 3 runs due to its lengthy runtime) of the tested models.

V. Discussion & Related Work

AutoRec proposes a similar architecture to ours [19]. In contrast to our model, they do not use any additional sparsification, i.e. train standard Autoencoders. During their evaluation, they report a better prediction error for an item-based approach (splitting input matrix across columns) than a user-based. Early testing for DÆM showed the contrary. It is worth noticing that their tested datasets (Movielens 1M, 10M) have lower user to item ratios of 1.5 and 7.2 in comparison to our 10. Furthermore, their optimal latent space dimensions are much higher (500), suggesting that denoising can reduce the necessary latent space needed to learn the relation, or their dataset has an intrinsically higher-dimensional latent space.

Denoising Autoencoders have been applied to CF before [20], but this example displays the goal to learn a $A = UV^T$ decomposition, where U represents latent factors between users, and V the same between items. This is similar to an SVD-decomposition $A = U'DV'^T$ where U' and V' play a similar role. To find such U and V they now apply DAE: in their setting, they have access to *side information* matrices X for users and Y for items. Finally, they use a DAE to learn a latent space representation of U by using X as an input. The same for V and Y. So they heavily rely on this side information, which they assume to be given. In short, they perform no row-wise or column-wise splitting of the matrix and their Autoencoder-based reconstruction function does not aim to learn the same user-item relationship as DÆM.

VI. Summary

We presented our implementation of a user-based denoising Autoencoder to improve upon context-insensitive Collaborative Filtering approaches. After tuning the model's hyperparameters, we show our implementation can reduce RMSE when compared to the best baseline by 0.0087. These results further reinforce the usage of Autoencoders to solve Collaborative Filtering problems. The introduction of sparsity in the form of dropout and our custom loss function allows the model to better learn a low-dimensional code in latent space for user-item relationships and directly compare the reconstruction of unknown values to ground-truth rating data.

REFERENCES

[1] J. Bughin, J. Doogan, and O. J. Vetvik, "A new way to measure word-of-mouth marketing," *McKinsey Quarterly*, vol. 2, no. 1, pp. 113–116, 2010.

[2] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems: an introduction*. Cambridge University Press, 2010.

[3] R. E. Hostler, V. Y. Yoon, Z. Guo, T. Guimaraes, and G. Forgionne, "Assessing the impact of recommender agents on online consumer unplanned purchase behavior," *Information & Management*, vol. 48, no. 8, pp. 336–343, 2011.

[4] S. S. Sundar, A. Oeldorf-Hirsch, and Q. Xu, "The bandwagon effect of collaborative filtering technology," in *CHI'08 extended abstracts on Human factors in computing systems*, 2008, pp. 3453–3458.

[5] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[6] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[7] 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*, 2001, pp. 285–295.

[8] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009.

[9] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[10] J. Bennett, S. Lanning *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007. New York, 2007, p. 35.

[11] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *International conference on algorithmic applications in management*. Springer, 2008, pp. 337–348.

[12] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *2011 IEEE 11th international conference on data mining*. IEEE, 2011, pp. 497–506.

[13] S. Rendle, W. Krichene, L. Zhang, and J. Anderson, "Neural collaborative filtering vs. matrix factorization revisited," in *Fourteenth ACM conference on recommender systems*, 2020, pp. 240–248.

[14] M. F. Alhamid, M. Rawashdeh, H. Al Osman, M. S. Hossain, and A. El Saddik, "Towards context-sensitive collaborative media recommender system," *Multimedia Tools and Applications*, vol. 74, no. 24, pp. 11 399–11 428, 2015.

[15] D. F. Murad, R. Hassan, B. D. Wijanarko, R. Leandros, and S. A. Murad, "Evaluation of hybrid collaborative filtering approach with context-sensitive recommendation system," in *2022 7th International Conference on Business and Industrial Research (ICBIR)*. IEEE, 2022, pp. 7–12.

[16] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, p. 3371–3408, dec 2010.

[17] G. Schröder, M. Thiele, and W. Lehner, "Setting goals and choosing metrics for recommender system evaluations," in *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, vol. 23, 2011, p. 53.

[18] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[19] 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*, 2015, pp. 111–112.

[20] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 811–820.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

DÆM: Denoising Autoencoder Model for Collaborative Filtering

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Schmid	Yannick
Weingarten	Matthew
Holbein	Loïc

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
16.7.22	Yannick Schmid
	Matthew Weingarten
	Loïc Holbein

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.