Feature Augmented Deep Neural Networks for Collaborative Filtering

Devang Kulshreshtha
Indian Institute of Technology, IIT-BHU, Varanasi, India
devang.kushreshtha.cse14@itbhu.ac.in

Abstract

In this paper, we present a novel framework termed as DAHCF, short for Deep Augmented Hybrid Collaborative Filtering. The model uses unsupervised deep learning architectures such as Restricted Boltzmann Machines, Autoencoders and Deep Belief Networks for learning low-dimensional latent representations of users and items. A feed-forward neural network is then proposed which utilizes these augmented features in a hybrid fashion and outputs the predicted ratings for the purpose of collaborative filtering. Empirically, the proposed DAHCF model beats classical approaches and also rivals the current state-of-the-art CF approach on the MovieLens-100K and MovieLens-1M datasets.

1 Introduction

Collaborative Filtering(CF) [Su and Khoshgoftaar, 2009] is a technique to recommend new items to a user based on the past transactions of similar users. Popularized by the Netflix challenge, various classical methods such as Matrix Factorization(linear [Sarwar *et al.*, 2000] and non-linear[Lawrence and Urtasun, 2009]), memory-based algorithms [Yu *et al.*, 2004], model-based [Aggarwal, 2016] algorithms have been proposed. Recently, deep neural networks have been successful in many tasks such as text and speech processing [Collobert and Weston, 2008], vision [He *et al.*, 2016], dimensionality reduction [Hinton and Salakhutdinov, 2006] etc. No wonder many researchers have made use of deep learning in Collaborative Filtering [Van den Oord *et al.*, 2013], [Covington *et al.*, 2016].

In this paper, we propose *DAHCF*, short form of Deep Augmented Hybrid Collaborative Filtering. It is composed of two broad steps. In the 1^{st} step, it makes use of one CF model to learn representations of items and/or users. In the 2^{nd} step it uses the extracted user and item features for rating prediction of items. The first model is an unsupervised deep-learning architecture such as AutoEncoder, Restricted Boltzmann Machines(RBM), Deep Belief Networks(DBN). The second model is a deep feed-forward neural network which learns to map the augmented features to prediction scores. Our hybrid approach performs well on two MovieLens datasets, outperforms some widely used approaches to CF and also rivals the current state-of-the-art approach in CF.

2 Proposed Approach

Our model is a sequence of two steps. In the 1^{st} step, we use 1 of the 3 architectures-RBM, Autoencoder, DBN for learning effective latent representations of users and items. The 2^{nd} step uses a feed-forward neural net to map these latent representations to predict ratings. These 2 steps are described in detail below.

2.1 Feature augmentation: Learning latent factors

In rating-based collaborative filtering, we have a sparsely-filled matrix $R \in \mathbb{R}^{N \times M}$ of N users and M items. A single entry R_{ui} is either empty or denotes the rating of user u on item i. Each user $u \in U = \{1...N\}$ can be represented by a sparse vector $s^{(u)} = (R_{u1}...R_{uM}) \in \mathbb{R}^M$. Similarly, each item $i \in I = \{1...M\} \in \mathbb{R}^N$ can be represented by a sparse vector $s^{(i)} = (R_{1i}...R_{Ni})$. The task is to learn deep user and item latent embeddings $p^{(u)} \in \mathbb{R}^{d_1}$ and $q^{(i)} \in \mathbb{R}^{d_2}$ from the sparse vectors $s^{(u)}$ and $s^{(i)}$, where $d_1 << M$ and $d_2 << N$. For learning high-level abstractions(e.g. in language, vision and other tasks), one needs deep architectures. Luckily, several unsupervised deep architectures(e.g. RBMs [Salakhutdinov *et al.*, 2007], Autoencoders [Sedhain *et al.*, 2015], DBNs [Zhao *et al.*, 2016]) have been applied successfully in Collaborative Filtering. They have performed well and beaten some very good classical approaches. We use these neural architectures i.e. Restricted Boltzmann Machines, Auto-encoders, and Deep Belief Networks to map each user and item into a latent space. Please note that all these architectures have been applied for CF independently but here they are being used for representing users and items into dense feature vectors, which is necessary for step 2.

Approach 1: Restricted Boltzmann Machine

RBM is a generative graphical model that learns a latent stochastic representation of the inputs. To use it in our model, we implement the RBM-CF [Salakhutdinov *et al.*, 2007] which is a generative probabilistic model based on Restricted Boltzmann Machine. The model projects each user as a training example of binary rating matrix V of size $M * K$, where K is the distinct values rating might take and $v_u^i = 1$ if user

u has rated item i as k and 0 otherwise. Each item is represented as a softmax input unit$v^1...v^M$. The M units are connected to a set of d hidden binary units. The network is trained by minimizing the gradient of Contrastive Divergence [Hinton, 2002]. The mathematical equations for inferring hidden states, reconstructing input nodes and weight updates have been omitted for the sake of clarity and can be found at [Salakhutdinov *et al.*, 2007].
After training, we project each user's binary matrix V to compute the activation of hidden units. As opposed to predicting the missing ratings by reconstructing the inputs using these hidden units, we use these activation values as latent representation of user. Mathematically, an RBM will transform sparse binary matrix V of each user into a dense latent vector $p^{(u)} \in \mathbb{R}^{d_1}$ where $d_1 << M$. Now, item latent features can be learnt in a similar way. We treat each item as a training example and each user as a softmax input unit. After training, the hidden activations will give us a dense vector $q^{(i)} \in \mathbb{R}^{d_2}$ for each item i ($d_2 << N$).

Approach 2: Autoencoders

In this approach, we use the AutoRec [Sedhain *et al.*, 2015] model which is based on Autoencoder framework for Collaborative Filtering. It takes the partially-filled user vector $s^{(u)}$ and project it into a lower-dimensional latent vector $H \in \mathbb{R}^{d_1}$. Then it reconstructs input in the output space $s'(u)$ to predict missing ratings for the purpose of collaborative filtering. The network will minimize the reconstruction error between actual($s^{(u)}$) and predicted($s^{'(u)}$) ratings. Again, we omit the mathematical equations regarding the update rule, objective function, encoding and decoding function and can be found at [Sedhain *et al.*, 2015].
After training, the latent layer H serves a dense vector ($p^{(u)}$) for each user u. We also compute q^i for each item i similarly by replacing $s^{(u)}$ in the input vector by $s^{(i)}$ and get $q \in \mathbb{R}^{d_2}$ for each item i.

Approach 3: Deep Belief Networks

[Zhao *et al.*, 2016] applied DBNs successfully to Collaborative Filtering and got superior results than RBM-CF. It consists of 4 layers: the initial 2 layers are an RBM, and the top 2 layers are an associative memory. Similar to RBM, the model treats each user as a training example, represented by a binary indicator matrix V. Weights are learned using an efficient greedy layer-wise unsupervised pretraining [Hinton *et al.*, 2006] between each layers. Note that the 1^{st} layer weights would be learned in the same fashion as RBM-CF. After this pre-training, weights are further fine-tuned by performing a contrastive version of wake-sleep algorithm [Hinton *et al.*, 2006] to reach the optimal solution. More details about learning and inference methods can be found at [Hinton *et al.*, 2006], [Zhao *et al.*, 2016].
After pre-training and fine-tuning, we take the last hidden layer to serve as a latent representation $p^{(u)} = \{H^1...H^d\}$ for each user u. Similarly, we compute q^i for each item i by clamping item's indicator matrix on the input layer with each user represented by an input neuron.

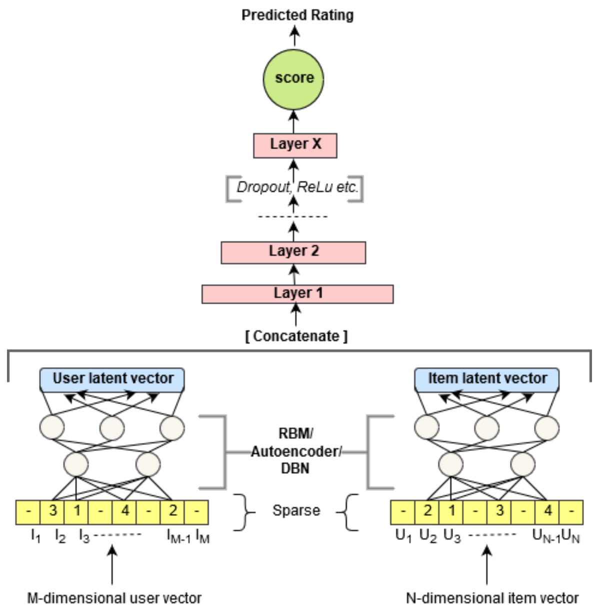

Figure 1: The DAHCF model for Collaborative Filtering

2.2 Deep Augmented Hybrid Collaborative Filtering

From the step 1, we have latent representation $p^{(u)}$ and $q^{(i)}$ of N users and M items. Our aim is to design a model that takes as input $p^{(u)}$ and $q^{(i)}$, discovers certain user-item latent interactions, and finally map the latent vectors to prediction scores.
Network Architecture: We adopt a multi-layer architecture to model each observed user-item interaction r_{ui}. The input to the model are 2 concatenated feature vectors p_u and q_i, which represent user u and item i respectively. These vectors are obtained from step 1. Above this input layer is a fully-connected layer that captures associations of common latent structures between p_u and q_i. The layer can also learn more complex features from only p_u (and/or q_i) respectively. We also have multiple hidden layers to discover more complex latent structures. The last hidden layer is followed by a single neuron which predicts the rating value r_{ui}.
Training: The network parameters are learned by jointly minimizing the Mean Absolute Error(MAE) between predicted (y_{ui}) and actual (r_{ui}) ratings and learned parameters(in order to avoid over-fitting).

$$L = \sum_{u,i,r_{ui}\neq 0} |r_{ui} - y(p_u, q_i)| + \lambda/2 \times (||W||^2) \quad (1)$$

Above optimization is preformed by back-propagating through layers via Mini-batch Gradient Descent.
DAHCF is distinct than existing CF techniques. Compared with Matrix Factorization approach [Koren *et al.*, 2009], which projects users and items into a shared latent space, DAHCF discards this restriction which can result in producing more accurate latent vectors for each of them. Yet, the interaction between features which represent the same concept in user vector and item vector would be captured by

Approach	ML-100K	ML-1M
I-RBM	0.774	0.710
U-RBM	0.782	0.724
R-DAHCF	**0.706**	**0.651**
I-AutoRec	0.717	0.674
U-AutoRec	0.730	0.689
A-DAHCF	**0.697**	**0.648**
I-DBN	0.752	0.699
U-DBN	0.757	0.720
D-DAHCF	**0.707**	**0.653**

Table 1: Comparison of the MAE of DAHCF and simple RBM/AutoRec/DBN models

the hidden neural layers. For e.g., if items are movies, hidden states of step 1 architectures could learn user latent dimensions as $\{H_{u1}$ *:fantasy*, H_{u2} *:action*$\}$, and item dimensions as $\{H_{i1}$ *:action*, H_{i2} *:romance*$\}$. In our model, one of the neuron can represent the feature $H_{u2} * H_{i1}$. Other neuron can represent $H_{u2} * H_{v2}$. Obviously, the neuron of the 2^{nd} feature is not important and it's weight on the output would be much less as compared to the $1st$ neuron. Further, MF allows learning linear latent representation, while DAHCF allows learning non-linear latent structures due to sigmoid and dropout present at various hidden layers. Also, the hidden layer(s) can also learn more complex features from the $p_i(q_j)$ vector only.

3 Experimental Results

3.1 Datasets and Evaluation Metric

We perform our experiments on MovieLens-100K and MovieLens-1M datasets [1], which consist of movie ratings collected over a period of time. MovieLens-100K contains 100,000 ratings from 943 users on 1,682 movies. While MovieLens-1M consists of 1,000,209 anonymous ratings of 3,900 movies made by 6,040 MovieLens users. For testing, we perform 5-fold cross validation with 80%-20% split between training and validation data. We report the average Mean Absolute Error(MAE) of the predicted ratings from true ratings of test data.

3.2 Details Training Architecture

We describe the model paramaters for each of the networks. We report those decisions or parameters that gave the lowest MAE on the 5-fold validation.

RBM: RBMs are trained using CD with $T = 1$ Gibb's sampling for 20 epochs. Weights are updated using learning rate=0.005, decay=0.95, regularization of 0.001 and momentum set to 0.6. Weights and biaises are initialized to 0. The number of hidden units is set to 75(user-RBM) and 50(item-RBM).

DBN: Individual RBMs are trained using same settings as above. We developed a DBN with 3 hidden layers of (50,125,200) and (75,125,200) units in item and user DBN.

AutoRec: For training Autoencoders, we use regularization strength as 0.01 and learning rate of 0.02. The latent dimension d is set to 300 for both user and item-based AutoRec.

[1]https://grouplens.org/datasets/movielens/

Model	ML-100K	ML-1M
PMF	0.793	0.731
H-NLPCA	0.784	0.726
L8	0.767	0.719
SCMF	0.715	0.668
U-CFN++	0.713	0.672
V-CFN++	0.706	0.654
R-DAHCF(ours)	0.706	0.651
D-DAHCF(ours)	0.707	0.653
A-DAHCF(ours)	**0.697**	**0.648**
State-of-the-art	0.680 (mSDA-CF)	0.640 (ORD-BM-COR)

Table 2: Comparison of the MAE of various models on MovieLens datasets

DAHCF: The concatenated user-item feature vector passes through three hidden layers of (375,125,50) units in the network. We use dropout(cite) with $p = 0.3$ after the 1^{st} hidden layer to regularize the model. We also use *ReLu* non-linearity on each of the hidden layers as it gives the best results on cross-validation set. The network is trained using SGD optimization with Adam update rule($lr = 0.001, beta1 = 0.9$).

3.3 Results & Discussions

Comparision with RBM/AutoRec/DBN models: Table 1 suggests neural networks combined/augmented with any of RBM/DBN/AutoRec perform much better than individual approaches. There is a huge MAE difference of around 0.7 in RBM-CF and R-DAHCF, which is quite an improvement. Difference in MAE of DBN and AutoRec with D-DAHCF and A-DAHCF are also noticeable.

Which is better among the 3 - R-DAHCF, D-DAHCF or A-DAHCF? We can see from Table 1 that A-DAHCF performs better than the other two, the reason could be attributed to the lower MAE of AutoRec as compared to RBM/DBN, and thus better latent feature learned. Also, DBN-CF has a lower MAE than RBM-CF yet the performace of D-DAHCF is almost similar to R-DAHCF. This is because the deeper layers of DBN-CF model correlations between lower layers of RBM-CF. It is possible to learn these features in R-DAHCF also. The FC hidden layers can also learn features from user(item) vector only if trained with the right parameters.

Comparision with classical approaches and state-of-the-art approach: Table 2 shows that DAHCF outperforms many baselines, including Probabilistic Matrix Factorization(PMF) [Mnih and Salakhutdinov, 2008], H-NLPCA [Vozalis *et al.*, 2010], Graph-based CF(L8) [Taranto *et al.*, 2012] and Sparse Covariance Matrix Factorization(SCMF) [Shi *et al.*, 2013]. It beats V-CNF++ [Strub *et al.*, 2016], a hybrid CF approach with Autoencoders and Neural Networks which also makes use of side information. Our model is also very close to the current state-of-the-art(mSDA-CF) [Li *et al.*, 2015],ORD-UI-BM-COR [Phung *et al.*, 2009]) with an MAE difference of only 0.017 and 0.008 respectively.

References

[Aggarwal, 2016] Charu C Aggarwal. Model-based collaborative filtering. In *Recommender Systems*, pages 71–138. Springer, 2016.

[Collobert and Weston, 2008] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM, 2016.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

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

[Hinton *et al.*, 2006] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

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

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

[Lawrence and Urtasun, 2009] Neil D Lawrence and Raquel Urtasun. Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 601–608. ACM, 2009.

[Li *et al.*, 2015] Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising autoencoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 811–820. ACM, 2015.

[Mnih and Salakhutdinov, 2008] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[Phung *et al.*, 2009] Dinh Q Phung, Svetha Venkatesh, et al. Ordinal boltzmann machines for collaborative filtering. In *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence*, pages 548–556. AUAI Press, 2009.

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

[Sarwar *et al.*, 2000] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[Sedhain *et al.*, 2015] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112. ACM, 2015.

[Shi *et al.*, 2013] Jianping Shi, Naiyan Wang, Yang Xia, Dit-Yan Yeung, Irwin King, and Jiaya Jia. Scmf: Sparse covariance matrix factorization for collaborative filtering. In *IJCAI*, 2013.

[Strub *et al.*, 2016] Florian Strub, Jérémie Mary, and Romaric Gaudel. Hybrid collaborative filtering with autoencoders. *arXiv preprint arXiv:1603.00806*, 2016.

[Su and Khoshgoftaar, 2009] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[Taranto *et al.*, 2012] Claudio Taranto, Nicola Di Mauro, and Floriana Esposito. Uncertain graphs meet collaborative filtering. In *IIR*, pages 89–100, 2012.

[Van den Oord *et al.*, 2013] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[Vozalis *et al.*, 2010] Manolis Vozalis, Angelos Markos, and Konstantinos Margaritis. Collaborative filtering through svd-based and hierarchical nonlinear pca. *Artificial Neural Networks–ICANN 2010*, pages 395–400, 2010.

[Yu *et al.*, 2004] Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, and H-P Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):56–69, 2004.

[Zhao *et al.*, 2016] Chong Zhao, Jiyun Shi, and Tao Jiang. Application of deep belief nets for collaborative filtering. *2016 16th International Symposium on Communications and Information Technologies (ISCIT)*, 2016.