Deep Autoencoder Neural Network Based Book Recommendation System

Jing Zhou, Jun Wang, Xiaowen Ding, Yu Zheng

*Abstract*—The paper presents the implementation of deep autoencoder neural network based book recommendation system. We detail the system design and compare the result with a baseline implementation of Alternating Least Square (ALS) model. The experiment results on Amazon dataset [1] show that the performance of autoencoder algorithm is much better comparing with ALS with a minimum root mean square error (RMSE) of 1.12. The project proves that the autoencoder neural network model can efficiently learns the latent structure of data with missing elements and can perfectly solve the recommendation system problem.

Keywords—Recommendation Systems; ALS; Autoencoder;

# Introduction (*Heading 1*)

On the Internet, where the number of choices is overwhelming, there is need to filter, prioritize and efficiently deliver relevant information in order to alleviate the problem of information overload, which has created a potential problem to many Internet users. Recommender systems solve this problem by searching through large volume of dynamically generated information to provide users with personalized content and services. Right now the most common approach to solve recommendation systems is to use ALS. ALS works by trying to find the optimal representation of a user and a product matrix – which when combined, should accurately represent the original dataset. The genius part of ALS is that it alternates between finding the optimal values for the user matrix and the product matrix. However, these methods are linear and cannot catch subtle factors. Recently a new algorithms – Autoencoders neural network were explored to solve those challenges [2]. Autocoders tackles the challenges: learning a non-linear representation of users and items. Autoencoders takes input as the sparse matrix of ratings learns a generative model. The model leads to a scalable and robust approach which beats state-of-the-art results in collaborative filtering.

The paper is organized as follows. First, a general review of ALS and its limitation. Then we explore Autoencoder algorithm aspects. Finally our experimental results are described in section III and related work are discussed in section IV.

# Alogrithm

## Alternating Least Square (ALS)

Recommendation system is typically a collaborative filtering problem. The collaborative filtering problem starts with a low rank approximation of the user-item matrix R (R={rij}u×v where each item Rij represents the rating score of item j by user i with the value being either a real number or missing. Here u designates the number of users and v designates the number of items. In this setup, collaborative filtering is designed to estimate missing values in R based upon the known values. ) Then models both users and items by giving them coordinates in a low dimensional feature space. Both the users and items have individual feature vectors where the rating of an item by a user is modeled as the inner product of the desired user and book feature vectors. Let U represent the user feature matrix and V represent the item feature matrix composed of both user and item feature vectors respectively.

Ideally ri,j =< ui, vj > ∀i, j, but in practice we need to

minimize loss functions of U and V to obtain these matrices.The loss function is the root mean square error (RSME).

where and are predicted and observed ratings for

user u and item v respectively. Here the predicted value is

computed via the following equation:

pi,j =< ui, vj >

The low rank approximation problem is thus formulated as

follows to learn the factor vectors (ui, vj ):

ALS rotates between fixing one of the unknowns ui or vj . When one is fixed the other can be computed by solving the least-squares problem. A general desctiption of the algorithm for ALS algorithm is as follows:

Step 1 Initialize matrix V by assigning the average rating for that book as the first row, and small random numbers for the remaining entries.

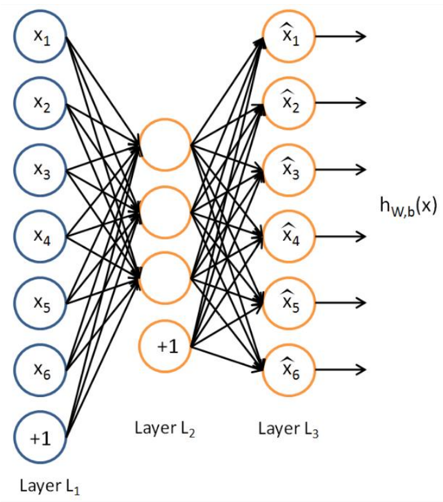
Step 2 Fix V , solve U by minimizing the RMSE function.

Step 3 Fix U, solve V by minimizing the RMSE function similarly.

Step 4 Repeat Steps 2 and 3 until convergence.

## Autoencoder

A solution to the problem of “backpropagation without a teacher” was first attempted by Hinton in the 1980s and Rumelhart in 1986[1]. Autoencoders were proposed as a solution to this problem, and the input data was used as the teacher. The backpropagation algorithm is traditionally used to train neural networks and literally “back propagates” the error in the network backward from the output layer to the input layer. Since labeled data is required for this task, this is categorized as a supervised learning algorithm. The capability of deep neural networks is limited due to weaknesses of this algorithm[2]. Backpropagation was not effective in deep layers and also most of the available input data was unlabeled. In 2006, Hinton attempted to overcome these problems through Deep Belief Networks (DBN), which are composed of a stack of Restricted Boltzmann Machines (RBM). Greedy layer-by-layer training is one of the core concepts of DBNs. A similarstrategy is used by stacking autoencoders yielding similar results[3]. High dimensional data is difficult to store and makes classification and visualization difficult. Reducing the dimensionality of such data is the key to such problems. Autoencoders are also known as “Autoassociators” or “Diabolo networks”. They work by encoding the input into a representation, and a decoder works to reconstruct the representation into an output. The output is, therefore, the input itself[4].



Autoencoders are feed-forward Neural Networks popularized by Kramer. They are unsupervised Networks where the output of the Network aims at reconstructing the initial input. The Network is trained by back-propagating the squared error loss on the reconstruction. Recent work in Deep Learning advocates to stack pre trained encoders to initialize Deep Neural Networks . This process enables the lowest layers of the Network to find low dimensional representations. It experimentally increases the quality of the whole Network.

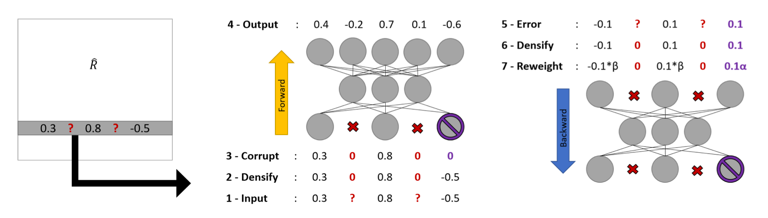
There is no standard approach for using sparse vectors as inputs of Neural Networks. Most of the papers dealing with sparse inputs get around by pre-computing an estimate of the missing values. In this project, we want the Autoencoder to handle this prediction issue by itself. Such problems have already been studied in industry where 5% of the values are missing. However in Collaborative Filtering we often face datasets with more than 95% missing values. Furthermore, missing values are not known during training in Collaborative Filtering which makes the task even more difficult. The approach we use includes three ingredients to handle the training of sparse Autoencoders:

• inhibit the edges of the input layers by zeroing out values in the input

• inhibit the edges of the output layers by zeroing out back-propagated values

• use a denoising loss to emphasize rating prediction over rating reconstruction.

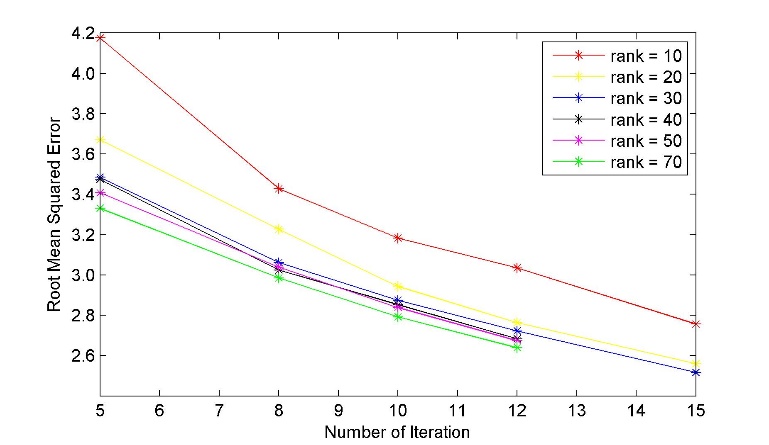
One way to inhibit the input edges is to turn missing values to zero. To keep the Autoencoder from always returning zero, we also use an empirical loss that disregards the loss of unknown values. No error is back-propagated for missing values, while the error is back-propagated for actual zero values. In other words, missing values do not bring information to the Network. This operation is equivalent to removing the neurons with missing values described in. However, Our method has important computational advantages because only one Neural Networks is trained whereas other techniques has to share the weights among thousands of Networks. Finally, we take advantage of the masking noise from the Denoising AutoEncoders (DAE) empirical loss. By simulating missing values in the training process, Autoencoders are trained to predict them. In Collaborative Filtering, this prediction aspect is actually the final target. Thus, emphasizing the prediction criterion turns the classic unsupervised training of Autoencoders into a simulated supervised learning. By mixing both the reconstruction and prediction criteria, the training can be thought as a pseudo-semi-supervised learning[5]. See figure below. The sparse input is drawn from the matrix of ratings, unknown values are turned to zero, some ratings are masked (input corruption) and a dense estimate is finally obtained. Before backpropagation, unknown ratings are turned to zero error, prediction errors are reweighed by α and reconstruction errors are reweighed by β.



From the time being U/V-CFN only relies on the feedback of users regarding a set of items. Let’s now incorporate additional information about the users and the items. These information help in several ways: increase the prediction accuracy, speed up the training, increase the robustness of the model, etc. Last but not least, incorporating side information is a well-known approach to tackle the cold start problem: when very little information is available on a user/item, Collaborative Filtering will have difficulties to infer its ratings. Instead of only adding the side information to the first layer of the Autoencoder, we propose to inject that information to every layer inputs of the Network. See figure below. By injecting the side information in every layer, the dynamic Autoencoders representation is forced to integrate this new data. However, to avoid side information to overstep the dense rating representation, we enforce the following constraints. The dimension of the sparse input must be greater than the dimension of the Autoencoder bottleneck which must be greater than the dimension of the side information.

# Experimental Results

## Alternating Least Square (ALS)



*(rank* is the number of latent factors in the model.)

## Autoencoder



# Related Work

In this section, we discuss the few related work on recommender systems. Restricted Boltzmann Machines (RBM) is the first work that applies neural network models to recommender systems. However, RBM targets rating prediction, not top-N recommendation, and its loss function considers only the observed ratings. It is technically challenging to incorporate negative sampling, which would be required for top-N recommendation, into the training of RBM. We are aware of a concurrent proposal called AutoRec, which uses the Auto-Encoder for rating prediction. The main differences are as follows: 1) AutoRec only considers the observed ratings in the loss function, which does not guarantee the performance for top-N recommendation. 2) They use the vanilla AutoEncoder structure, while we prove that introducing user factors in the model can greatly improve performance. 3) AutoRec does not employ the denoising technique. Another related work is [7], which also uses the Auto-Encoder for recommender systems. This work studies the particular problem of article recommendation, and improves the well-known model Collaborative Topic Regression [6] by replacing its Topic Model component by a Bayesian Auto-Encoder, which is used for learning the latent feature representations for the articles. Other contributions deal with this problem by using Neural Networks properties for CBF: Neural Networks are first trained to learn a feature representation from the item which is then processed by a CF approach such as Probabilistic Matrix Factorization to provide the final rating. For instance, respectively auto-encode bag-of-words from restaurant reviews and movie plots, auto-encode heterogeneous side information from users and items. Finally, use Convolutional Networks on music samples.

##### References

[1] C. Gomez-Uribe and N. Hunt, “The netflix recommender system: Algorithms, business value, and innovation,” ACM Trans. Manage. Inf. Syst., vol. 6, no. 4, pp. 13:1–13:19, 2015.

[2] P. Lops, M. D. Gemmis, and G. Semeraro, “Content-based recom- mender systems: State of the art and trends,” in Recommender systems handbook. Springer, 2011, pp. 73–105.

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

[4] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, “Large-scale parallel collaborative filtering for the netflix prize,” in Algorithmic Aspects in Information and Management. Springer, 2008, pp. 337–348.

[5] S. Rendle, “Factorization machines,” in Proc. of ICDM’10, 2010, pp. 995–1000.

[6] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD ’11, pages 448–456. ACM, 2011.

[7] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’15, pages 1235–1244. ACM, 2015.