STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **12**, March 2024, pp 524–529.
Published online in International Academic Press (www.IAPress.org)

Using Transfer Adaptation Method for Dynamic Features Expansion in Multi-label Deep Neural Network for Recommender Systems

Fargana Abdullayeva, Suleyman Suleymanzade

Institute of Information Technology, Baku, Azerbaijan

Abstract In this paper, we propose to use a convertible deep neural network (DNN) model with a transfer adaptation mechanism to deal with varying input and output numbers of neurons. The flexible DNN model serves as a multi-label classifier for the recommender system as part of the retrieval systems' push mechanism, which learns the combination of tabular features and proposes the number of discrete offers (targets). Our retrieval system uses the *transfer adaptation*, mechanism, when the number of features changes, it replaces the input layer of the neural network then freezes all gradients on the following layers, trains only replaced layer, and unfreezes the entire model. The experiments show that using the transfer adaptation technique impacts stable loss decreasing and learning speed during the training process. Furthermore, our proposed model demonstrates notable advantages in production scenarios. Specifically, it exhibits enhanced efficiency, manifesting in accelerated processing times and improved resource utilization, thereby contributing to a more sustainable and cost-effective training of machine learning solutions in real-world applications.

Keywords Retrieval systems, recommender system, DNN, transfer learning, multilabel-classification

DOI:10.19139/soic-2310-5070-1836

1. Introduction.

An automation process for discovering the proper recommendations, play crucial role in e-commerce industry [1] [2] . Recommender systems are used for push phase [3] in most retrieval engines [4] [5]. There are multiple strategies to build proposals mechanism [6], commonly two most accepted methods are: the content based filtering [7] and collaborative filtering [8]. Generally, in both methods, there are two disjoin sets of discreet nodes, called *profiles* and *items*. The profiles are active agents with historical data that represents an interaction with items. For example, profile P_i buys item I_j, where interactions between profiles and items can carry different meaning. The purpose of the recommender system is to forecasting items for profiles by considering their historical data.

In content-based filtering, recommendations determined by the content of profiles data that already made some actions, later, when the new profile is registered, the classifier starts to measure distances [9] between registered and the rest of profiles' features. Based on these distances [10], the system proposes same or similar offers.

For collaborative filtering, the central criteria to propose new offers, done by analyzing the interaction between profiles and items. This mapping with historical data gives non-linear data representation where the various AI methods such as graph neural networks [11] can be used. Additionally, the structure can be enriched with temporal information, what gives the new dimension to evaluate recommender decisions. There are also hybrid approaches that take into account profiles content as well [12] [13] [14] [15] as the profiles-items interaction most of them based on DNN [16] [17], to approximate matrix factorization via embedding layers.

In our system, we build recommender system as multi-label deep learning network for handling tabular data [16] where on each set of tabular features F, there are $[0..n]$ number of discreet offers. The challenge that

ISSN 2310-5070 (online) ISSN 2311-004X (print)

we faced with such architecture. described in "problem statement" section. Next, in the "method" section, we presented multiphase algorithm so solve dynamic input expansion in multi-label classifier. Afterward, the results in experiment section, confirms our assumptions regarding the transfer-adaptation strategy. In conclusion section, we describe some possible enlargement for proposed method.

2. Problem statement.

Suppose there are DNN based model M in production, that accepts F features, $F = \{f_1, f_2 \ldots f_n\}$ n - number of features to get the set of recommended offers $O = M(F)$ where $O \in \{o_1, o_2 \ldots o_m\}$; m - offers' number. At some point there the number of features changed from n to k where $k > n$. The system must adapt model M to be able to accept extended features number k during the runtime.

There are multiple ways to handle features expansion [18] for neural networks model in production. The most common approach, based on manually adjusting [19] neural networks input sizes, following training phase, starting from the initial state. But there are several cons with such organizing: 1. Factually, this is the replacement of one model with another, where the set of learnable parameters **W** are lost and must be rediscovered from the ground up, 2. It causes to more time and GPUs energy consumption for retraining process.

One way to handle dynamic attributes rising, is to enhance fully connected layers with convolutional predecessor network [20] [21]. This approach involves having multiple filters to conduct the output with a certain size. It works good for large datasets, but still has obstacles on the input part where convolutional input layer, requires to be initialized with known number of channels.

Dealing with re-adjustable feature numbers in deep neural network causes to rethink classical neural network model view, with dynamically changeable architecture on the input and output layers, that we describe in following section.

3. Method.

Our approach provides a non-expensive way to adapt DNN for the features expansion. These operations are provided by the part of our retrieval system that responsible for recommenders push service [22]. The method contains three-phase steps to handle feature extension in production shown below.

1. Reproduction phase.
2. Adaptation phase.
3. Update phase.

3.1. Reproduction phase.

To provide the reliable online service for multiple clients during architectural update, the model M is replicated M'. Afterwards all clients' requests are redirected to the new copied model M'. This, proxy model still handles n previous number of features. In production such approach called blue-green strategy [23].

3.2. Adaptation phase.

An adaptation phase includes core concept of the transfer adaptation (transfer learning [24] like) approach. Retrieval system updates model's input/output layer, depending on the features/offers expansion, and trains only the updated layer. The algorithms of transfer-adaptation shown in Figure1.

```
1.  function adapt($\mathcal{M}, k, \mathcal{E}, \eta, D^{k\times N}, O^{m\times N}$)
2.    $\mathcal{M}$.layer[0] = linear($k$, $\mathcal{M}$.layer[0].output)
3.    for layer in 1..$\mathcal{M}$.layers
4.      layer.autograd ← false #freeze
5.    endfor
6.    for $e$ in range($\mathcal{E}$) #train
7.      $O'$ ← $\mathcal{M}$.forward(D)
8.      $\mathcal{L}$ ← Loss(O, O')
9.      $\mathcal{M}$.update($\mathcal{L}, \eta$)
10.   endfor
11.   for layer in 1..$\mathcal{M}$.layers
12.     layer.autograd ← true #unfreeze
13.   endfor
14. endfunction
```

Figure 1. Transfer-adaptation algorithm.

The new input layer of size k is created and the input matrix with shape of $k \times L_2^{input}$ replaces previous matrix $k \times L_2^{input}$, where L_2^{input} – is second layer's input size. After replacing and initializing [25] an input layer, the rest of DNN's layers freeze (autograd [26] [27] parameter are set to False). The process starts to train (adapt) models' input layer for an e number of epochs (in our experiments $[5..15]$) to fit k features number. Then, all models' layers are unfrozen and autograds are activated. The algorithm. The similar operation is done for output layer multi-label's classifier. The chosen number of epochs is not necessary to be as high as in full training. Because there is only one layer that must be retrained, the adaptation process is fast. After an adaptation is done, then whole model can be continue to be train during relatively low number of epochs. In Experiment part we show the difference between train the whole model with skipping an adaptation part versus including the adaptation.

3.3. Update phase.

After the adaptation, a model sets back to production. The figure 2. shows how retrieval system handles features extension to provide push service during the runtime. $M(F_n)$ – is the model that handles n number of the features n, that comes from the preprocessor object $P(S)$ that receives structural/unistructural features from the different sources $S \in \{s_1, s_2, \ldots s_l\}$ including other models.

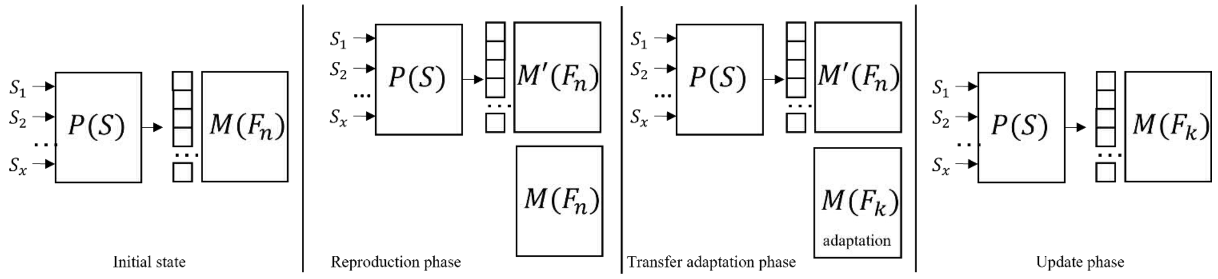

Figure 2. Features extension phases, that's done by the retrieval system' process.

Preprocess object P, provides structural data for model that is in production (shown by the arrow), on the last stage is sets back by the retrieval system from the copied model $M'(F_n)$ to adapted model $M(F_k)$. Then, model M' is destroyed and garbage collected.

4. Experiment.

To proof, that our assumptions work, we decided to provide experiments with various structural datasets that have multiple discreet target classes simultaneously. These multi-label classes represent offers while, features defined as aggregated data that belongs to profiles.

4.1. Model.

We took a simplest model architecture that consist of fully connected layer [28] stack, with ReLU [29] nonlinear activation functions and list of the output fully-connected layers with sigmoid activation function. We use sigmoid activation function on the last layer instead of softmax [30], because of the multi-label classification problem where each offer's output is independent from the other. For optimization strategy we used Adam [31] optimizer.

4.2. Experiment 1.

For the experience we took the yeast [32] [33] and *foodtruck* [34] structural dataset, with 14 binary labels, and 103 continues features. To check how features adaptation works, we created four datasets: two for each train/validation before and after features expansion phases. We dropped 40 last continues features to simulate the state before features expansion and hold the all the features for the second phase, so we had two datasets with 103 and 63 features (each with train/validation sets respectively).

For training we created 2 model copies: On the first model we not applied transfer adaptation, on another we applied Figure 3. The training process took 3 phases that shows on the Figure 3: on first phase there was training with 63 features X_{63}, next another dataset has been applied with 103 features X_{103} for 20 more epochs. In first model update the first neural network's layer, but we didn't apply the transfer adaptation technique.

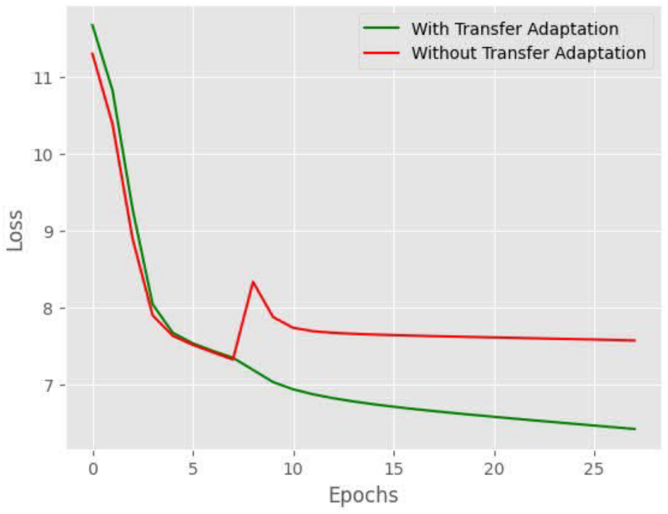

Figure 3. Training process with/without transfer adaptaion technique (experiment 1).

In the second experiment all the steps were the same except the transfer adaptation that has been applied after updating the first neural networks layer. For train and adaptation phase we use the same learning rate $Lr = 0.0004$. The adaptation phase took 15 epochs for only one layer which was fast and inexpensive to compute.

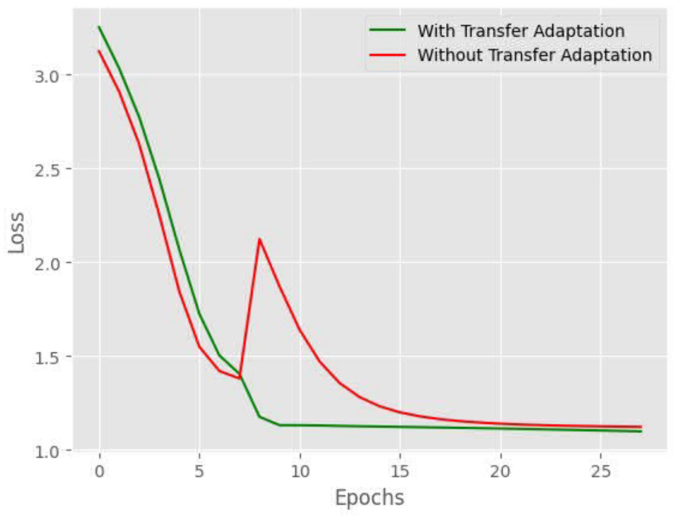

Figure 4. Training process with/without transfer adaptaion technique (experiment 2).

Table 1. The contrast of learning process with and without applied transfer adaptation.

Datasets	Metrics	Without transfer adaptation	With transfer adaptation
Yeast	Precision	0.34959	**0.59974**
	Recall	**0.74533**	0.73599
	F1	0.47594	**0.66092**
	Jaccard	0.31229	**0.49356**
FoodTruck	Precision	0.48648	**0.50648**
	Recall	**0.57029**	0.56600
	F1	0.68899	**0.70588**
	Jaccard	0.39889	**0.40449**

5. Conclusion

Our paper shows the advantage of partial fine-tuning process after updating neural networks architecture. Just by adapting one layer, the loss reduces much smoother and for the same amount of epochs the results contrast is significant. This approach can also be used for another parts of retrieval systems for example in the ranking model, where features number are also increases during the runtime.

REFERENCES

1. Ben Schafer, Joseph Konstan, John Riedi, *Recommender Systems in E-Commerce*, 1999.
2. Sanjeevan Sivapalan, Alireza Sadeghian, Hossein Rahanam, Asad M. Madni, *Recommender Systems in E-Commerce*, in World Automation Congress (WAC), Kona, Hawaii, 2014.
3. Xiao Liu, Xiao Zou, Zilong Ji, Gengshuo Tian, Yuanyuan Mi, Tiejun Huang, K. Y. Michael Wong, *Push-pull Feedback Implements Hierarchical*, in 33rd International Conference on Neural Information Processing Systems, Vancouver, Canada., 2019.
4. Fuji Ren, David B. Bracewell, *Advanced Information Retrieval*, Electronic Notes in Theoretical Computer Science, vol. 225, pp. 303-317, 2009.
5. Wei Dong Huang, Min Qian Li, Ji Dong Yang, *The Design of the Emergency Retrieval System Based on User Preference*, Applied Mechanics and Materials, Vols. 157-158, pp. 882-886, 2012.
6. Robin BurkeAlexander, Felfernig Alexander, FelfernigMehmet H. Göker, *Recommender Systems: An Overview*, AI Magazine, no. 32, pp. 13-18, 2011.

7. Francesco Colace,Dajana Conte,Massimo De Santo,Marco Lombardi,Domenico Santaniello, *A content-based recommendation approach based on singular value decomposition*, CONNECTION SCIENCE, vol. 34, no. 1, pp. 2158-2176 , 2022.
8. J. Ben Schafer, Dan Frankowski, Jon Herlocker & Shilad Sen , *Collaborative Filtering Recommender Systems*, The Adaptice Web, vol. 4321, pp. 291-324, 2007.
9. Zhan Su, Zhong Huang, Jun Ai , Xuanxiong Zhang, Lihui Shang, Fengyu Zhao, *Enhancing the scalability of distance-based link prediction algorithms in recommender systems through similarity selection*, in PLoS ONE, 2022.
10. Narges Hasanzadeh, Yahya Forghani, *Improving the Test Time of M-Distance based Recommendation System*, Journal of The Institution of Engineers (India) Series B , vol. 103, no. 7, 2021 .
11. Shiwen Wu, Wentao Zhang, Fei Sun, Bin Cui, Xu Xie, *Graph Neural Networks in Recommender Systems: A Survey*, ACM Computing Surveys, vol. 55, no. 5, p. 1–37, 2023.
12. Oumaima Stitini, Soulaimane Kaloun, Omar Bencharef, *An Improved Recommender System Solution to Mitigate the Over-Specialization Problem Using Genetic Algorithms*, Computer and System Engineering Laboratory, Faculty of Science and Technology, Cadi Ayyad University, Marrakech , 2022.
13. Mohammed Baidada, Khalifa Mansouri, Franck Poirier, *Hybrid Filtering Recommendation System in an Educational Context.*, International Journal of Web-Based Learning and Teaching Technologies, vol. 17, no. 1, 2022.
14. Paula A. Rodríguez, Demetrio A. Ovalle & Néstor D. Duque, *A Student-Centered Hybrid Recommender System to Provide Relevant Learning Objects from Repositories*, in International Conference on Learning and Collaboration Technologies, Las Vegas, NV, USA, 2015.
15. Mojtaba Salehi, Isa Nakhai Kmalabadi, *A Hybrid Attribute–based Recommender System for E–learning Material Recommendation*, IERI Procedia, vol. 2, pp. 565-570, 2012.
16. Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, Gjergji Kasneci, *Deep Neural Networks and Tabular Data: A Survey*, 2022.
17. Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua, *Neural Collaborative Filtering*, in Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 2017.
18. Mate Kovacs, Victor V. Kryssanov, *Expanding the Feature Space of Deep Neural Networks*, International Journal of Machine Learning and Computing, vol. 10, no. 2, 2020.
19. Nathaniel Egwu, Thomas Mrziglod, & Andreas Schuppert. *Neural network input feature selection using structured l^2 - norm penalization*, Applied Intelligence, 2022.
20. Yitan Zhu, Thomas Brettin, Fangfang Xia, Alexander Partin, Maulik Shukla, Hyunseung Yoo, Yvonne A. Evrard, James H. Doroshow, Rick L. Stevens, *Converting tabular data into images for deep learning with convolutional neural networks*, Scientific Reports, vol. 11325, no. 11, 2021.
21. Andre Ye & Andy Wang *Applying Convolutional Structures to Tabular Data*, Modern Deep Learning
22. Gebrekirstos G. Gebremeskel, Arjen P. de Vries, *Pull–push: a measure of over- or underpersonalization in recommendation*, International Journal of Data Science and Analytics, 2022.
23. Bo Yang, Anca Sailer, Ajay Mohindra, *Survey and Evaluation of Blue-Green Deployment Techniques in Cloud Native Environments*, in ICSOC 2019 Workshops, 2020.
24. Asmaul Hosna, Ethel Merry, Jigmey Gyalmo, Zulfikar Alom, Zeyar Aung, & Mohammad Abdul Azim, *Transfer learning: a friendly introduction*, Journal of Big Data, vol. 9, no. 102, 2022.
25. Kit Wong, Rolf Dornberger, & Thomas Hanne, *An analysis of weight initialization methods in connection with different activation functions for feedforward neural networks*, 2022.
26. Michael Bartholomew-Biggs, Steven Brown, BruceChristianson, *Automatic differentiation of algorithms*, Journal of Computational and Applied Mathematics, vol. 124, no. 1-2, pp. 171-190, 2000.
27. Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Siskind, *Automatic differentiation in machine learning: A survey*, Journal of Machine Learning Research , vol. 18, no. 153, pp. 1-43, 2018.
28. Erol Gelenbe, Yonghua Yin, *Deep Learning with Dense Random Neural Networks*, in ICMMI '17, Cracow, 2017.
29. A. F. Agarap, *Deep Learning using Rectified Linear Units (ReLU)*, 2018.
30. Kunal Banerjee, Vishak Prasad C, Rishi Raj Gupta, Karthik Vyas, Anushree H, & Biswajit Mishra, *Exploring Alternatives to Softmax Function*, Intel Corporation, 2020.
31. Diederik P. Kingma, Jimmy Ba, *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*, in 3rd International Conference for Learning Representations, San Diego, 2015.
32. N. Chumuang, *Comparative Algorithm for Predicting the Protein Localization Sites with Yeast Dataset*, in International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Las Palmas de Gran Canaria, 2018.
33. Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, Guojie Li, Runsheng Chen, *Topological structure analysis of the protein-protein interaction network in budding yeast*, Nucleic Acids Research, vol. 31, no. 9, p. 2443–2450, 2003.
34. Adriano Rivolli, Larissa C. Parker, Andre de Carvalho, *Food Truck Recommendation Using Multi-label Classification*, in Portuguese Conference on Artificial Intelligence, 2017.