Toward Auto-Learning Hyperparameters for Deep Learning-Based Recommender Systems

Bo Sun[1,2], Di Wu[1,3(✉)], Mingsheng Shang[1], and Yi He[4]

[1] Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China
wudi.cigit@gmail.com
[2] Chongqing School, University of Chinese Academy of Sciences, Chongqing 400714, China
[3] Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangzhou 510006, China
[4] Department of Computer Science, Old Dominion University, Norfolk, VA 23462, USA

Abstract. Deep learning (DL)-based recommendation system (RS) has drawn extensive attention during the past years. Its performance heavily relies on hyperparameter tuning. However, the most common approach of hyperparameters tuning is still Grid Search—a tedious task that consumes immerse computational resources and human efforts. To aid this issue, this paper proposes a general hyperparameter optimization framework for existing DL-based RSs based on differential evolution (DE), named DE-Opt. Its main idea is to incorporate DE into a DL-based RS model's training process to auto-learn its hyperparameters λ (regularization coefficient) and η (learning rate) simultaneously at layer-granularity. Empirical studies on three benchmark datasets verify that: 1) DE-Opt is compatible with and can automate the training of the most recent DL-based RSs by making their λ and η adaptively learned, and 2) DE-Opt significantly outperforms the state-of-the-art hyperparameter searching competitors in terms of both higher learning performance and lower runtime.

Keywords: Recommender systems · Hyperparameter tuning · Grid search · Deep learning · Differential evolution

1 Introduction

Recommender system (RS) has become an indispensable building block in online service, which boosts business and elevates user experience [1]. Traditional RSs tend to have inferior performance in dealing with data sparsity and cold-start issues and balancing the recommendation qualities against different evaluation metrics [2–4]. To aid the issues, recently advanced RSs manifesting remarkable results mostly advocated deep learning (DL) techniques [3, 5], thanks to their strong capability of modeling user-item interactions that are complex and non-linear in practice [5, 6].

A. Bhattacharya et al. (Eds.): DASFAA 2022, LNCS 13246, pp. 323–331, 2022.
https://doi.org/10.1007/978-3-031-00126-0_25

Despite their successes, existing DL-based RSs mainly suffer from a common drawback—the tedious process of hyperparameter tuning [7], for two reasons. On the one hand, due to the black-box nature of neural architecture, the hyperparameters in a DL-based RS directly decide the system properties in both training and trained states, e.g., how fast can the RS converge and how well the converged equilibrium optimizes the designed learning objective. On the other hand, the ubiquity of DL-based RSs exacerbates this drawback. In fact, various regularization terms have been engineered to incorporate side information of the items and users (e.g., textual description of items and geographical relations among users) in a wide range of recommendation applications, where each such term added to the learning objective introduces a new hyperparameter, and a well-tuned DL-based RS can usually lead to significantly superior performance [7–9]. However, the most common approach of tuning the RS hyperparameters is still Grid Search [10, 15, 16], which is a tedious process that consumes immerse computational resources and human efforts.

To alleviate the drawback of Grid Search, some efforts are made to tune hyperparameters of RSs automatically. A classic study is SGDA [11] that adaptively tunes λ (regularization coefficient) based on alternating optimization. Inspired by SGDA, λOpt is proposed [8] to tune λ adaptively by enforcing regularization during training. Besides, to tune the learning rate η, an η-adaptive optimizer can be adopted. Such as Adam [17], AMSGrad [12], and AdaMod [13]. However, these approaches can only tune a single hyperparameter λ or η adaptively.

In this paper, we propose a general hyperparameter optimization framework for DL-based RSs based on differential evolution (DE) [18, 19], named DE-Opt, with its key ideas being twofold. First, DE-Opt decomposes and controls the RS training process in a finer level of granularity by allowing the hyperparameters to differ across neural layers. This enlarges the search space for optimal hyperparameters yet alleviates the impact of each hyperparameter so as to better the resultant models. Second, DE-Opt frames the hyperparameter tuning task as an optimization problem and embeds it into the RS training framework, so that the model parameters and the hyperparameters of an RS are jointly trained. As such, DE-Opt allows to train the RSs *only once*, thereby substantially improving the training efficiency of DL-based RSs by attaining their optimal RS models and hyperparameters all at once.

Specific Contributions of this paper are summarized as follows:

- This is the first work to propose the auto-learning of the model parameters and hyperparameters of DL-based RSs at layer-wise granularity.
- The proposed DE-Opt framework is agnostic to the learning objectives and thus is compatible with most existing DL-based RSs, boosting their training efficacy by making hyperparameters auto-learned along with the model parameters jointly.
- Extensive experiments on three real-world datasets are carried out to show that our DE-Opt 1) can automate the training of the state-of-the-art DL-based RS models, and 2) outperforms the state-of-the-art hyperparameter searching competitors.
- To promote reproducible DL research, we open-access our code implementation by a link at https://github.com/Wuziqiao/DE-Opt.

2 The Proposed DE-Opt Optimization Framework

We design the architecture of DE-Opt. It consists of two parts, as shown in Fig. 1.

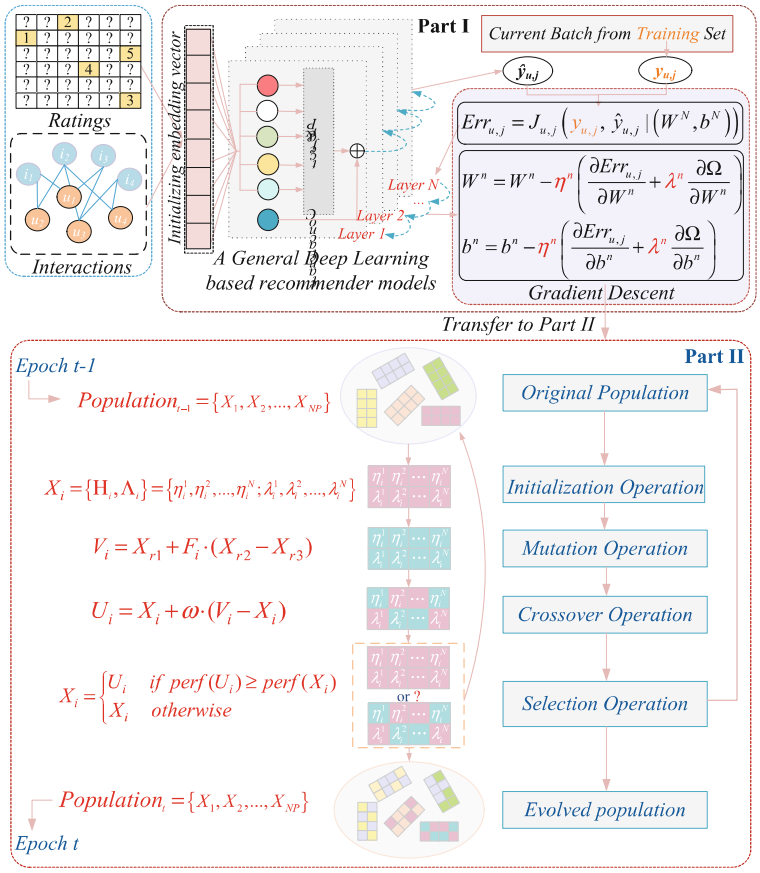


Fig. 1. The architecture of DE-Opt.

2.1 Part I: RS Training with Layer-Wise Granular Hyperparameter Control

To make different layers be finely controlled, we match η and λ with model depth (the number of hidden layers), as shown in Table 1. H and Λ corresponding to the η and λ vectors, respectively, where η^n denotes η at n-th layer and λ^n denotes λ at n-th layer, $n \in \{1, 2, .., N\}$.

Table 1. The vectorized hyperparameters of matching to model depth.

Hyperparameters	Layer 1	…	Layer n	…	Layer N
Learning rate: H	η^1		η^n		η^N
Regularization coefficient: Λ	λ^1		λ^n		λ^N

When the observed ground-truth $y_{u,j}$ on interaction (u, j) of user u and item j is input from the current batch of the training set, the current objective function is:

$$L_{u,j}(\Theta) = L_{u,j}(W, b) = J_{u,j}(W, b) + \Lambda \otimes \Omega(W, b), \tag{1}$$

where $J_{u,j}(\cdot)$ denotes the loss function on $y_{u,j}$ and $\otimes$ denotes the element-wise product between Λ and layers (λ^n corresponding to n-th layer). Although different DL-based recommendation models have different working mechanisms, their DL architectures can be uniformly expressed as follows [5, 15]:

$$a^1 = W^1 x + b^1,\ a^n = f\left(W^n a^{n-1} + b^n\right),\ a^N = f\left(W^N a^{N-1} + b^N\right), \tag{2}$$

where x denotes initializing embedding vector, a^n denotes the embedding vector of n-th layer, $f(\cdot)$ denotes the activation function, W^n denotes the n-th weight vector or matrix, b^n denotes the n-th bias term. After aggregating the last embedding vector a^N, we can obtain the final prediction output. Then, we can calculate the error $Err_{u,j}$ of the output of the last N-th layer on (u, j) as follows:

$$Err_{u,j} = J_{u,j}\left(y_{u,j},\ \hat{y}_{u,j}|\left(W^N, b^N\right)\right). \tag{3}$$

where $\hat{y}_{u,j}$ denotes the corresponding prediction of $y_{u,j}$. By employing stochastic gradient descent (SGD) to (1) and (3) and according to the chain rule [5–7], we can obtain the updating rules of model parameters of n-th layer as follows:

$$W^n \leftarrow W^n - \eta^n \frac{\partial L_{u,j}}{\partial W^n} = W^n - \eta^n \left(\delta^n a^{n-1} + \lambda^n \frac{\partial \Omega}{\partial W^n}\right), \tag{4}$$

$$b^n \leftarrow b^n - \eta^n \frac{\partial L_{u,j}}{\partial b^n} = b^n - \eta^n \left(\delta^n + \lambda^n \frac{\partial \Omega}{\partial b^n}\right), \tag{5}$$

Formulas (4) and (5) show that each layer is finely controlled with different hyperparameters at layer-granularity.

2.2 Part II: DE-Based Hyperparameter Auto-Learning

This part is to employ DE to auto-learn H and Λ simultaneously, including four operations of Initialization, Mutation, Crossover, and Selection [18, 19].

Initialization. At training epoch $t - 1$, part II starts with a pair of initial H and Λ with NP individuals, where each pair is called a 'target vector' as follows:

$$X_i = \{\mathrm{H}_i, \Lambda_i\} = \left\{\left[\eta_i^1, \eta_i^2, .., \eta_i^N\right], \left[\lambda_i^1, \lambda_i^2, ..., \lambda_i^N\right]\right\}, \tag{6}$$

where X_i stands for the i-th individual, $i \in \{1, 2, \ldots, NP\}$. All the NP individuals group together to form an original population.

Mutation. Mutant vector V_i is generated for each target vector X_i. We choose the *DE/Rand/1* mutation strategy to generate V_i for the sake of robustness [18, 19] as:

$$V_i = X_{r1} + F_i \cdot (X_{r2} - X_{r3}), \tag{7}$$

where $r1, r2, r3 \in \{1, 2, \ldots, NP\}$ are randomly different from each other and i, and F_i is the scaling factor that positively controls the scaling of various vectors. The setting of scaling factor F_i decides the balance of convergence speed and the optimal performance. We adapt the scale factor local search in DE (SFLSDE) to set F_i self-adaptively according to [18, 19].

Crossover. After mutation, crossover uses V_i to disturb X_i to increase population diversity. X_i and V_i are subject to the crossover operation to generate a new trial vector U_i. We adopt the *DE/CurrentToRand/1* strategy [18, 19] to generate the trial vector U_i, which linearly combines X_i and V_i as follows:

$$U_i = X_i + \omega \cdot (V_i - X_i), \tag{8}$$

where ω is a random number between 0 and 1. Joining (7) into (8), we arrive at:

$$U_i = X_i + \omega \cdot (X_{r1} - X_i) + F_i \cdot (X_{r2} - X_{r3}). \tag{9}$$

Selection. Next, we test a DL-based recommendation model's performance with U_i or X_i on validation dataset as follows:

$$X_i = \begin{cases} U_i \; if \; perf\,(U_i) \geq perf\,(X_i) \\ X_i \; otherwise \end{cases}, \tag{10}$$

where function $perf\,(\cdot)$ denotes evaluating a DL-based recommendation model's performance on a validation dataset.

After all the NP target vectors (individuals) are evolved at training epoch $t - 1$, DE-Opt moves to the next training epoch t.

3 Experiments

We mainly aim to investigate the following research questions:

RQ.1. Can DE-Opt improve the state-of-the-art (SoTA) models with hyperparameter searched and fixed ad-hoc?
RQ.2. Can DE-Opt outperform the SoTA hyperparameter searching competitors?

3.1 General Settings

Datasets. Three real-world datasets [5], i.e., *FilmTrust*,[1] *MovieLens 1M* (See footnote 1), and *Each Movie* (See footnote 1), are adopted to benchmark our DE-Opt, with their statistics summarized in Table 2. Each dataset is randomly split into the training/validation/testing sets that contain 70%/10%/20% observed entries, respectively.

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

Table 2. Statistics of the studied datasets

No.	Name	#User	#Item	#Interaction	Density
D1	Film trust	1508	2071	37,919	1.21%
D2	MovieLens 1 M	6040	3706	1,000,209	4.47%
D3	Each movie	72,916	1628	2,811,983	2.36%

Baselines. We compare our DE-Opt with the following state-of-the-art models.

Fixed Hyperparameter Models. Three item ranking models: NeuMF (WWW 2017) [14], LRML (WWW 2018) [16], and NGCF (SIGIR 2019) [20].

Adaptive Hyperparameter Models. Two λ-adaptive models: SGDA (WSDM 2012) [11] and λOpt (KDD 2019) [8] whose η is set by Grid Search. We respectively adopt Adam (ICLR 2015) [17], AMSGrad (ICLR 2018) [12], and AdaMod (2019) [13] to optimize λOpt to implement three λ-η-adaptive models: Adam-λOpt, AMSGrad-λOpt, and AdaMod-λOpt.

Evaluation Metrics. To evaluate item ranking accuracy, we adopt three widely-used evaluation protocols [5], i.e., NDCG@K, Recall@K, and Hit Ratio (HR)@K, where K is a cutoff parameter in the top-ranked list.

Implementation Details. The searching space of Grid Search and DE-Opt are the same. We adopt the ten times amplification principle for setting searching space, i.e., the searching space is ten times the optimal values set by Grid Search. Except for λ and η, all the other hyperparameters of involved models are set according to original papers. The default optimizer is SGD.

4 Comparison with Fixed Hyperparameter Models (RQ.1)

Table 3 records the detailed comparison results of item ranking accuracy. We conduct the Wilcoxon signed-ranks test [21] on the comparison results of Table 3 with a significance level of 0.05. The results demonstrate that DE-Opt can significantly improve NeuMF and LRML in item ranking accuracy. Although the hypothesis is not accepted on NGCF, we can see that DE-Opt outperforms Grid Search in most cases. Besides, Fig. 2 shows the comparison results of CPU running time in item ranking tasks. We find that DE-Opt consumes significantly less time than Grid Search.

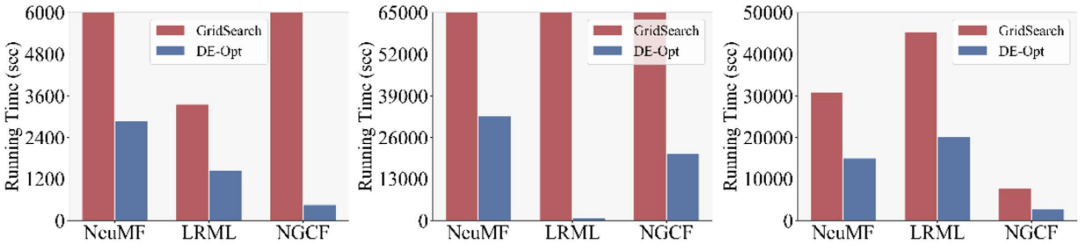
4.1 Comparison with Adaptive Hyperparameter Models (RQ.2)

Table 4 presents the comparison results. These comparisons are also checked by the Wilcoxon signed-ranks test [21] with a significance level of 0.05. Table 4 clearly shows that DE-Opt achieves significantly higher item ranking accuracy than all the comparison models. Besides, we also compare their computational efficiencies. The results are shown in Fig. 3, where we see that DE-Opt significantly outperforms the adaptive hyperparameter models in computational efficiency.

Table 3. The comparison results of item ranking accuracy

Dataset	Metric*	NeuMF		LRML		NGCF	
		Grid search	DE-Opt	Grid search	DE-Opt	Grid search	DE-Opt
D1	NDCG@10	0.4513	**0.4641**	**0.3781**	0.3746	0.3734	**0.3773**
	Recall@10	0.5404	**0.5500**	0.4385	**0.4460**	**0.4002**	0.3987
D2	NDCG@10	0.3035	**0.3047**	0.2136	**0.2234**	0.3541	**0.3544**
	Recall@10	0.1241	**0.1262**	0.0806	**0.1296**	0.1366	**0.1392**
D3	NDCG@10	0.1078	**0.1105**	0.0655	**0.0752**	0.0523	**0.0528**
	Recall@10	0.1908	**0.1921**	0.1138	**0.1315**	0.0651	**0.0663**
Statistic	*p-value*	**0.0156**		**0.0313**		0.1094	

* A higher value indicates a higher item ranking accuracy



(a) Dataset-D1 (b) Dataset-D2 (c) Dataset-D3

Fig. 2. The comparison results of GPU running time in item ranking task.

Table 4. The comparison results with adaptive hyperparameter models

Dataset	**Metric***	SGDA	λOpt	Adam-λOpt	AMSGrad-λOp	AdaMod-λOpt	**DE-Opt**
D1	HR@50	0.8696	0.8724	0.8738	0.8749	0.8765	**0.8791**
	NDCG@50	0.4515	0.4692	0.4916	**0.5003**	0.4981	0.4993
D2	HR@50	0.3877	0.4115	0.4213	0.4302	0.4242	**0.4349**
	NDCG@50	0.1335	0.1471	0.1524	0.1553	0.1529	**0.1556**
D3	HR@50	0.4430	0.4720	0.5123	0.5217	0.5285	**0.5339**
	NDCG@50	0.1278	0.1351	0.1462	0.1478	0.1484	**0.1535**
Statistic	*p-value*	**0.0156**	**0.0156**	**0.0156**	**0.0469**	**0.0156**	**—**

* A higher value indicates a higher item ranking accuracy

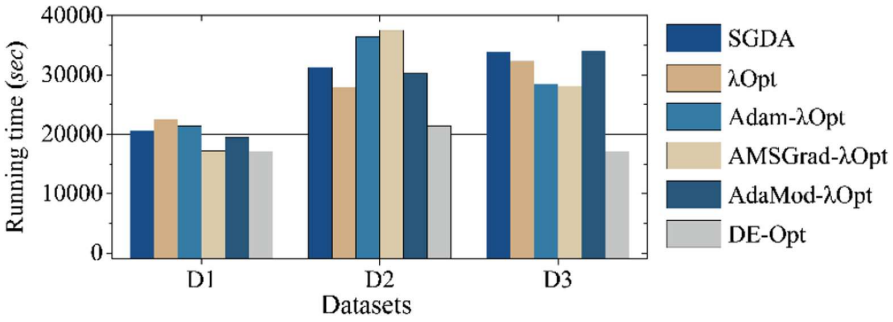


Fig. 3. The comparison results of GPU running time

5 Conclusion

This paper proposes a general hyperparameter optimization framework based on differential evolution (DE) for existing deep learning (DL)-based recommender systems (RSs), named DE-Opt. Its idea of DE-Opt is to incorporate DE into a DL-based recommendation model's training process to auto-learn its hyperparameters λ (regularization coefficient) and η (learning rate) simultaneously at layer-granularity (each layer has different λ and η). Empirical studies on three benchmark datasets verify that: 1) DE-Opt can significantly improve state-of-the-art DL-based RS models' performance by making their λ and η adaptive, and 2) DE-Opt also significantly outperforms state-of-the-art adaptive hyperparameter models.

Acknowledgements. This work is supported in part by the National Natural Science Foundation of China under Grants 62176070, 62072429, 62002337, and 61902370, in part by the Chinese Academy of Sciences "Light of West China" Program, and in part by the Key Cooperation Project of Chongqing Municipal Education Commission (HZ2021008).

References

1. Wu, D., Shang, M., Luo, X., Wang, Z.: An L1-and-L2-norm-oriented latent factor model for recommender systems. IEEE Trans. Neural Netw. Learn. Syst. (2021). https://doi.org/10.1109/TNNLS.2021.3071392
2. Wei, T., et al.: Fast adaptation for cold-start collaborative filtering with meta-learning. In: ICDM, pp. 661–670 (2020)
3. Meng, L., Shi, C., Hao, S., Su, X.: DCAN: deep co-attention network by modeling user preference and news lifecycle for news recommendation. In: DASFAA, pp. 100–114 (2021)
4. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-characteristic-aware latent factor model for web service QoS prediction. IEEE Trans. Knowl. Data Eng. (2020). https://doi.org/10.1109/TKDE.2020.3014302
5. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: a survey and new perspectives. ACM Comput. Surv. **52**(1), 1–38 (2019)
6. Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: RecSys, pp. 191–198 (2016)
7. Maher, M., Sakr, S.: SmartML: a meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms. In: EDBT, pp. 554–557 (2019)

8. Chen, Y., et al: λopt: learn to regularize recommender models in finer levels. In: SIGKDD, pp. 978–986 (2019)
9. Dong, X., Shen, J., Wang, W., Shao, L., Ling, H., Porikli, F.: Dynamical hyperparameter optimization via deep reinforcement learning in tracking. IEEE Trans. Pattern Anal. Mach Intell. **43**(5), 1515–1529 (2021)
10. Ndiaye, E., Le, T., Fercoq, O., Salmon, J., Takeuchi, I.: Safe grid search with optimal complexity. In: ICML, pp. 4771–4780 (2019)
11. Rendle, S.: Learning recommender systems with adaptive regularization. In: WSDM, pp. 133–142 (2012)
12. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of adam and beyond. In: ICLR (2018)
13. Ding, J., Ren, X., Luo, R., Sun, X.: An adaptive and momental bound method for stochastic learning. arXiv preprint arXiv:1910.12249 (2019)
14. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: WWW, pp. 173–182 (2017)
15. Fu, Z., et al.: Deep learning for search and recommender systems in practice. In: SIGKDD, pp. 3515–3516 (2020)
16. Tay, Y., Anh Tuan, L., Hui, S.C.: Latent relational metric learning via memory-based attention for collaborative ranking. In: WWW, pp. 729–739 (2018)
17. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
18. Wu, D., Luo, X., Wang, G., Shang, M., Yuan, Y., Yan, H.: A highly-accurate framework for self-labeled semi-supervised classification in industrial applications. IEEE Trans. Ind. Informat. **14**(3), 909–920 (2018)
19. Neri, F., Tirronen, V.: Scale factor local search in differential evolution. Memetic Comput. **1**(2), 153–171 (2009)
20. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: SIGIR, pp. 165–174 (2019)
21. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)