TẠP CHÍ KHOA HỌC
TRƯỜNG ĐẠI HỌC SƯ PHẠM TP HỒ CHÍ MINH
Tập 22, Số 9 (2025): 1564-1575
Website: https://journal.hcmue.edu.vn
ISSN: 2734-9918

HO CHI MINH CITY UNIVERSITY OF EDUCATION
JOURNAL OF SCIENCE
Vol. 22, No. 9 (2025): 1564-1575
https://doi.org/10.54607/hcmue.js.22.9.4540(2025)

Research Article

OPTIMIZING RECOMMENDER SYSTEMS: INTEGRATING COLLABORATIVE FILTERING WITH BAYESIAN AND GAUSSIAN TECHNIQUES

Nguyen Tuan Anh

Ho Chi Minh City University of Foreign Languages – Information Technology, Vietnam
Corresponding author: Nguyen Tuan Anh – Email: anhnt1@huflit.edu.vn
Received: July 10, 2024; Revised: February 07, 2025; Accepted: July 08, 2025

ABSTRACT

One frequently utilized method in recommendation systems is collaborative filtering (CF). Fine-tuning the hyperparameters of CF algorithms remains a difficult task even with developments in modeling consumers and products/services. This work explores an alternative approach for this aim by means of Bayesian optimization using Gaussian processes during hyperparameter alteration. This method reduces the time and effort usually needed for manual tuning by autonomously adjusting hyperparameters for two basic and simple CF algorithms on three popular datasets, yielding competitive results: Netflix Prize, Movielens 1M, and Movielens 10M. Therefore, it could enable practitioners to improve the performance of their recommendation systems whilst greatly shortening the time and effort spent on tuning their systems.

Keywords: Bayesian optimization; collaborative filtering; Movielens; Netflix Prize

1. Introduction

1.1. Background

Customers have numerous options across various digital retail and multimedia platforms. The abundance of options hinders products' capacity to meet specific consumer preferences. Businesses like Amazon, Google, and Netflix rely heavily on tailored recommendation systems. These technologies enable item customization to align with consumer preferences, thereby improving customer satisfaction and loyalty (Adomavicius & Tuzhilin, 2005). The primary method for building recommendation systems today is collaborative filtering (CF). CF relies solely on previous user activity, such as ratings or purchases, rather than explicit user profiles. This method enables CF to provide recommendations using minimal data collection and without the need for domain-specific knowledge (Yan, 2014). Additionally, user behavior may allow CF to uncover patterns and preferences not evident in traditional methods. Amazon and Netflix have shown effectiveness in commercial applications (Zhang et al., 2014).

Cite this article as: Nguyen, T. A. (2025). Optimizing recommender systems: Integrating collaborative filtering with Bayesian and Gaussian techniques. *Ho Chi Minh City University of Education Journal of Science, 22*(9), 1564-1575. https://doi.org/10.54607/hcmue.js.22.9.4540(2025)

1.2. Problem Statement

While modeling approaches have evolved, hyperparameter tuning in CF systems has stayed mostly human work (Rendle et al., 2019). Usually labor-intensive and maybe inconsistent, this approach is not always the best one. To attain the best performance, for instance, the winning solution from the Netflix Prize competition required a complete hand-off of hyperparameters (Koren, 2008, 2009).

1.3. Objective and Contributions

Using Bayesian optimization with Gaussian priors, several beneficial applications for hyperparameter optimization have shown success (Imani et al., 2022; Morita et al., 2022; Snoek et al., 2012). Still, its capacity to improve CF methods on well-known datasets has not been discovered. This work uses Bayesian optimization with Gaussian processes to optimize the hyperparameters in two simple CF algorithms, hence bridging this gap. Therefore, our research work aims to:

a) automate hyperparameter adjustment, reducing dependence on user interaction,
b) boosting the performance of CF algorithms on extensively used datasets.

Our findings on the Netflix Prize, Movielens 1M, and Movielens 10M datasets illustrate the efficacy of our approach, getting competitive performance notwithstanding not being at the top (our source code for these tests is published at https://github.com/anhnt1/netflix). As far as we know, there are no research results on applying Bayesian optimization with Gaussian priors to the two testing CF algorithms on all three datasets.

1.4. Paper Outline

The rest of this paper will be organized as follows: The second part will go over our research objects and research methodology. The current research, with a particular focus on hand hyperparameter tweaking, is also presented in the second section. The next section will then present our research results with two basic and simple CF algorithms applied to the three datasets: Netflix Prize, Movielens 1M, and Movielens 10M. This paper then concludes with a conclusion section.

2. Research objects and methodology

2.1. Literature Review

Research on recommender systems has been remarkable, with an emphasis on modeling people, objects, and their interactions. CF has been a pillar in this field with major research on several strategies to raise its efficiency. Early CF publications include Koren et al. (Koren, 2008, 2009) and others concentrating on matrix factorization techniques for user-item interactions. These techniques split the user-item interaction matrix into latent components reflecting people and objects, and then apply them to forecast user preferences. Further research has proposed more intricate models, including hybrid methods combining content-based filtering with CF to leverage both user behavior and item attributes (Z. Li et al., 2017). Horasan et al. (2023) argue that although typical matrix decomposition methods utilized in CF, including Singular Value Decomposition (SVD) and Non-negative Matrix

Factorization (NMF), have great computational complexity, they are outstanding in managing scalability. This motivates the computationally efficient substitute for the scalability method, so the Truncated–ULV decomposition technique (T–ULVD) has been proposed. Including knowledge graphs (KGs) in CF systems allows the authors of (Chen et al., 2021) to overcome data sparsity. Including user and item characteristics from KGs improves the simulation of user-item interactions. By means of trust linkages between users in CF models, the authors of (Khaledian & Mardukhi, 2022) can also overcome data sparsity. The performance of CF systems has been increasingly enhanced by deep learning applications. Complex user-item interactions and latent characteristics may now be learned using methods including multi-model deep learning and collaborative deep forest learning (CDFL). These approaches combine deep neural networks with traditional CF techniques to offer better control of data sparsity and higher recommendation accuracy (Aljunid & Doddaghatta Huchaiah, 2020; Molaei et al., 2021). Furthermore, attention structures and recurrent neural networks have captured information from long-distance interactions, thereby improving the depth of feature representation (Xia et al., 2021). Using Bayesian statistics for matrix factorization, Previous studies (Rendle, 2012; Rendle et al., 2019; Salakhutdinov & Mnih, 2008) estimate missing values in a matrix so that their models may better anticipate users' preferences. Their primary goal is thus the same as that of other research works, which concentrate just on more accurate parameter estimation of CF models.

Although during the past two decades, advanced CF models have made great progress, hyperparameter tuning is still vital to improve these systems. Usually requiring manual adjustment, traditional approaches are not only time-consuming but also prone to producing worse results due to the broad search area and complicated interactions between parameters. Their effects on generalizing, computing efficiency, and model accuracy complicate the tuning process (Bardenet et al., 2013; Rendle et al., 2019; Szabó & Genge, 2020). For practitioners just entering the industry and requiring great topic knowledge, this manual technique could be very difficult.

Like Bayesian optimization, automated approaches have benefits in tackling the challenges of hyperparameter tweaking, hence improving the correctness of the model and lowering the modeling time needed (Hoos, 2012). The success of such automatic tuning methods has been demonstrated in several real-world scenarios (Imani et al., 2022; Morita et al., 2022; Snoek et al., 2012), thereby stressing a possible path of research in recommender systems. We want to investigate a new, useful, and practical approach to adjust hyperparameters in CF systems with the potential to get good performance. In this sense, the dependency on human labor will be lessened, and the performance of the CF algorithm might be improved. This might help to hasten innovation in this field by letting more practitioners create and apply successful recommender systems.

2.2. The Datasets

The Netflix Prize competition used more than 100 million movie evaluations for training, and the quiz and test sets each has 1.4 million contemporary ratings for assessment.

We utilize the same datasets for training, testing, and validation, unlike other studies (Kim & Suh, 2019; Steck, 2019) that employ a modified methodology involving the binarization of ratings and user filtering, or by randomly dividing the original training set to create additional datasets (Shenbin et al., 2020). The Movielens 1M and 10M datasets are extensively used for evaluating the effectiveness of recommender systems. We randomly divided these two datasets in a 90:10 ratio for training and testing, a commonly used proportion in research, as cited in (Rendle et al., 2019).

2.3. Two CF Algorithms

The two fundamental collaborative filtering algorithms from Koren (2008) presented in this section provide the foundation for more sophisticated models. The first method embeds individuals as well as objects into a common latent factor space. The objective is to ascertain underlying preferences by representing users and items on related vectors. Each user u is associated with a vector $p_u \in R^n$, referred to as the user-factor vector, whereas each item i is linked to a vector $q_i \in R^n$, known as the item-factor vector. The anticipated ratings, represented as $\hat{r}_{ui}$, are computed as the inner product of these vectors, namely $\hat{r}_{ui} = p_u^T q_i$. The optimization of latent components seeks to reduce the discrepancy between anticipated ratings and actual ratings r_{ui} throughout the procedure:

$$minimize \sum_{(u,i)\in K} (r_{ui} - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2)$$

K is a set defined as $K = \{(u,i)|r_{ui}\ is\ known\}$, λ is a regularization parameter that has to be tuned. Thus, the prediction error for any estimate can be given as $e_{ui} = r_{ui} - \hat{r}_{ui}$. We take each training instance and pass through all ratings included in the set K. For every rating r_{ui}, we make changes to model parameters counter to the gradients given by:

$$p_u \leftarrow p_u + l_r(e_{ui}q_i - \lambda p_u), \qquad q_i \leftarrow q_i + l_r(e_{ui}p_u - \lambda q_i)$$

where l_r is the learning rate.

The second CF method enhances the first by including baseline estimation to address the inherent user and item biases in CF data. Baseline estimations are used to address these issues: μ is the overall mean rating, while b_u and b_i signify the departures of the user and item, respectively, from the unbiased component $\mu + p_u^T q_i$. The forecast may then be calculated as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i$$

The associated loss is minimized to estimate parameters:

$$minimize \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2)$$

Analogous to the first approach, we iterate over all known ratings in the dataset K. Model parameters are adjusted for each rating r_{ui} as follows:

$$b_u \leftarrow b_u + l_r(e_{ui} - \lambda b_u), \qquad b_i \leftarrow b_i + l_r(e_{ui} - \lambda b_i)$$
$$p_u \leftarrow p_u + l_r(e_{ui}q_i - \lambda p_u), \qquad q_i \leftarrow q_i + l_r(e_{ui}p_u - \lambda q_i)$$

The algorithms repeat K times and finish when the loss does not considerably decrease after many iterations.

2.4. Bayesian Optimization with Gaussian Processes

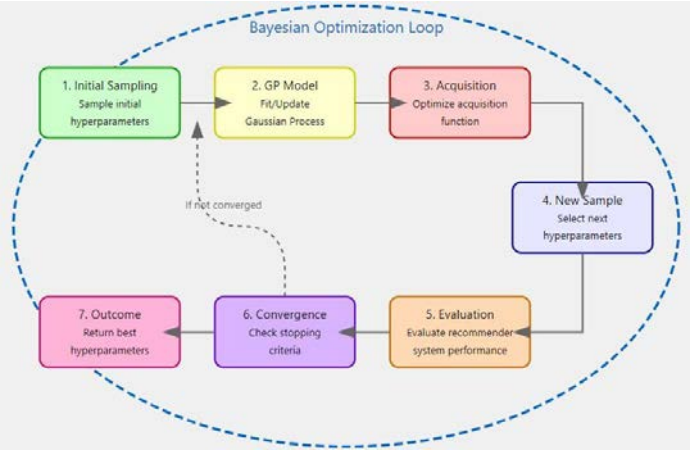

Figure 1. *Bayesian Optimization Process for CFs.*

Figure 1 depicts the iterative nature of Bayesian optimization using a loop that covers the entire process. When convergence is not achieved, the dashed arrow from the convergence check indicates the iterative nature of the procedure.

1. Initial Sampling: First, the approach selects beginning hyperparameters from the hyperparameter space.

2. Gaussian Process Model: Second, the given data fits the Gaussian Process model. New data updates the model in subsequent rounds. In this stage, we employ Bayesian statistics, which takes into account every previous function evaluation to update our beliefs and generate a posterior distribution and uncertainty estimates.

3. Acquisition Function: Third, the current GP model is utilized to develop an acquisition function. This helps balance exploitation and exploration.

4. New Sample: Fourth, the acquisition function is adjusted to choose the next set of hyperparameters to test.

5. Evaluation: Fifth, the recommender system's performance is evaluated using the hyperparameters that were selected.

6. Convergence Check: Sixth, the technique checks whether the halting requirements (performance criterion or maximum iterations) have been reached.

7. Outcome: Seventh, if converged, the process yields the best hyperparameters identified. This should not be the case; it returns to step 2.

A comprehensive elucidation of the fundamental concepts of Bayesian optimization is available in (Frazier, 2018). The primary benefit of Bayesian optimization is its ability to use all available data effectively, which incorporates all prior function evaluations when selecting the subsequent exploratory point, hence enhancing its efficiency and effectiveness in optimization.

2.5. Evaluation Metrics

This study uses Root Mean Square Error (RMSE) as the assessment measure. It is mathematically defined as the square root of the mean of the squared deviations between the expected and actual values:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (P_i - O_i)^2}$$

where P_i is the predicted value for the i-th observation, O_i is the actual value for the i-th observation, and n is the total number of observations. A lower RMSE value indicates that the model's predictions are closer to the actual values, meaning the model has better accuracy.

3. Results and discussion

3.1. Empirical Findings

In our trials, we used scikit-optimize (Tim et al., 2021) for hyperparameter optimization, which employs Bayesian optimization. Specifically, we use the gp_minimize function of the library with all the possible default values (including the default acquisition function, number of iterations, number of initial random iterations, and kernel). Based on our experiences when optimizing hyperparameters (the search space should be wide enough to capture the potential variations in model performance, while also being manageable to ensure efficient computation during training and evaluation phases), we set the initial learning rate l_r is established at 5e-3, the regularization parameter λ varies from 1e-4 to 1.0, and the dimension n ranges from 50 to 500. If the loss reduction is below 1e-4 after two rounds, the learning rate is reduced by a factor of 10. The parameter optimization procedure, which refines b_u, b_i, p_u, q_i, terminates if the loss reduction is less than 1e-4 after five iterations (stopping criterion).

At each stage, two primary tasks delineate the complexity of Bayesian optimization using Gaussian processes: adjusting the hyperparameters of the Gaussian process and optimizing the acquisition function to choose a new set of hyperparameters. The complexities of each of these tasks are $O(N^3)$ and $O(N^2)$, respectively, where N denotes the number of data points (Garnett, 2023; Snoek et al., 2012). Our findings and experience demonstrate that a single iteration of Bayesian optimization requires significantly less time than optimizing parameters in one iteration; consequently, the time needed to search for a new set of hyperparameters (a new exploration point) is nearly negligible compared to the time necessary to optimize the parameters of a CF algorithm. Figure 2 illustrates the duration required for Bayesian optimization using Gaussian processes in our experiment, which involves two hyperparameters and 100 iterations. Note that this time consumption is unaffected by CF algorithms since they operate as a black box inside the Bayesian optimization framework. In contrast to alternative methods, Grid Search (Bergstra & Bengio, 2012) exhibits exponential complexity $O(k^d)$, where k represents values per parameter and

d denotes dimensions, rendering it unfeasible for high-dimensional spaces. Conversely, Random Search (Bergstra & Bengio, 2012) scales linearly with the number of iterations $O(k)$, but it may be less sample-efficient. Evolutionary Algorithms (Guido et al., 2023) frequently exhibit significant complexity based on population size and the number of generations, but techniques such as Hyperband (L. Li et al., 2017) ($O(k * log\ k)$) provide efficiency improvements through adaptive resource allocation. Despite the elevated per-iteration cost of Bayesian optimization with Gaussian processes due to the cubic term, its efficacy is evident in its demand for substantially fewer costly function evaluations (N) to identify optimal hyperparameters, in contrast to Grid or Random Search, rendering it particularly advantageous when the evaluation of the objective function constitutes the principal constraint.

In our experiment, the two algorithms attain RMSE values of 0.8103 (λ = 3.36E-02 and n = 90) and 0.8107 (λ = 0.026905275 and n = 87) on the Test set of the Netflix Prize datasets, respectively. A recent study (Steck, 2019) uses a modified approach to datasets by binarizing ratings and filtering users, or by randomly partitioning the original training set to generate supplementary datasets (Shenbin et al., 2020). This hinders our ability to compare the findings; nonetheless, the top-performing algorithm in the Netflix Prize competition attained an RMSE score of 0.8567 on the Test set (Koren, 2009).

Tables 1 and 2 show the outcomes of the two CF methods on the Movielens 1M and Movielens 10M datasets. The outcomes for the Movielens 1M dataset from other techniques are derived from (Han et al., 2021), whilst those for the Movielens 10M dataset are from Rendle et al. (2019).

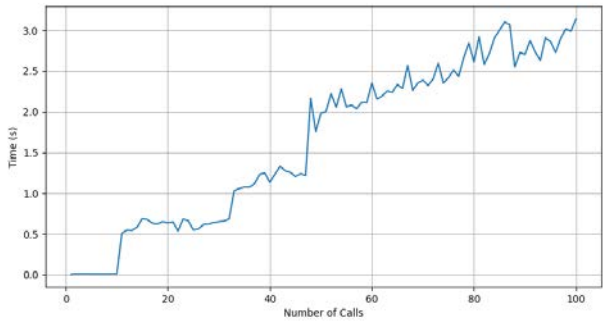

Figure 2. *Bayesian optimization time: time for finding evaluation points*

3.2. Analysis

The findings indicate that Bayesian optimization using Gaussian priors is beneficial for hyperparameter adjustment in the two CF models. This approach achieves competent performance based on findings from the three popular datasets, and practitioners can always opt to use more complicated CF algorithms to get better results. While not yielding the most optimal results relative to other advanced methodologies, the hyperparameter-tuned algorithms demonstrated commendable performance, highlighting the capacity of Bayesian optimization to improve recommender systems in this context. Therefore, this method's pragmatic merits include time efficiency and a much-reduced need for human adaptation.

Table 1. *Performance comparison of various algorithms on the Movielens 1M dataset*

Algorithm	RMSE (Movielens 1M)
LLORMA	0.833
CF-NADE	0.829
GC-MC	0.832
GraphRec	0.843
GraphRec+Extra	0.842
SparseFC	0.824
IGMC	0.857
GLocal-K	0.822
Our approach (first algo.)	**0.8410**
Our approach (second algo.)	**0.8459**

Table 2. *Performance comparison of various algorithms on the Movielens 10M dataset*

Algorithm	RMSE (Movielens 10M)
RSVD	0.8256
GSMF	0.8012
I-AutoRec	0.7820
LLORMA	0.7815
AdaError	0.7644
SGD MF	0.7720
Bayesian SVD++	0.7563
Bayesian timeSVD++	0.7523
Bayesian timeSVD++ flipped	0.7485
Our approach (first algo.)	**0.7757**
Our approach (second algo.)	**0.7787**

3.3. Strengths and Limitations

This study demonstrated that Gaussian processes can be used in Bayesian optimization to automate hyperparameter tuning, minimizing the need for human intervention due to the labor-intensive and the error-prone nature of manual tuning. This approach significantly reduces time and effort while achieving competitive performance across various neural networks and datasets, including Netflix Prize, Movielens 1M, and 10M. This approach may enhance primary collaborative filtering performance due to its simplicity and practicality for those without manual tuning skills. This approach can be applied in various optimization algorithms or domains requiring hyperparameter optimization, extending its applicability beyond recommender systems. In the hyperparameter search campaign, this is insufficient, leading to unequal benchmark results. It has focused on basic collaborative filtering algorithms. While simple implementations serve as test cases, results may not fully represent the optimization potential in more complex models. The framework was assessed on only three datasets, potentially restricting its applicability to real-world scenarios.

4. Conclusion

4.1. Summary of Findings

The evaluation of the method was performed using the three most prominent datasets for the collaborative filtering systems, namely the Netflix Prize, Movielens 1M, and Movielens 10M. The results revealed that this technique massively reduces, if not almost

eliminates, model tuning while still ensuring the competitiveness of the model. Although the strategy does not score the best RMSE relative to that of the state-of-the-art models, it seems to still achieve good results without the need to use very many complicated collaborative filtering algorithms. This research presents a novel, effective, and pragmatic method for hyperparameter adjustment in CF systems, with the potential to enhance performance. This methodology has not been explicitly examined for CF and recommender systems.

4.2. Implications

This study presents an automated Bayesian hyperparameter tuning framework with some significant implications for dynamic recommender systems, which require continuous updates to recommendations in response to evolving user behavior and catalog content.

Ongoing self-improvement: replacing manual grid search with Bayesian optimization allows the model to re-tune autonomously on a rolling basis, such as nightly or in response to drift alerts, requiring minimal human intervention. This reduces operational costs and decreases the adaptation loop from days to hours, enabling the recommender to monitor swift changes in user preferences or item catalogs.

Accelerated deployment and testing: the reduction in tuning time (around 50% in our experiments) enables business teams to deploy new features or data signals into production, receive immediate feedback, and perform rollbacks or iterations without enduring a prolonged retraining backlog. In dynamic environments such as news, e-commerce flash sales, and streaming content launches, the speed of operations directly influences user engagement and revenue generation.

Facilitating democratization for teams with limited resources: small and mid-sized companies frequently do not have specialized machine learning engineers. An auto-tuning layer simplifies the deployment of advanced collaborative filtering, allowing teams to utilize state-of-the-art techniques with minimal MLOps infrastructure.

Robustness in real-time under non-stationary conditions: dynamic recommenders encounter concept drift and seasonal variations. Bayesian optimization employs a probabilistic surrogate, specifically a Gaussian Process, to quantify uncertainty, enabling it to prioritize hyperparameter regions that are likely to maintain stability under drift.

4.3. Future Work

Future possible studies might investigate more complex applications of Bayesian optimization to other, more sophisticated CF algorithms that use deep learning or hybrid algorithm designs. Also, an increased number of datasets, such as those showing distinct user behavior patterns or varying levels of sparsity from other sources, would give a broader understanding of the method's implementation.

❖ ***Conflict of Interest:*** *Author has no conflict of interest to declare.*

❖ ***Acknowledgments:*** *This research was conducted independently by the author without any funding from any organizations or individuals.*

REFERENCES

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734-749. https://doi.org/10.1109/TKDE.2005.99

Aljunid, M. F., & Doddaghatta Huchaiah, M. (2020). Multi-model deep learning approach for collaborative filtering recommendation system. *CAAI Transactions on Intelligence Technology*, *5*(4), 268-275. https://doi.org/10.1049/trit.2020.0031

Bardenet, R., Brendel, M., Kégl, B., & Sebag, M. (2013). Collaborative hyperparameter tuning. *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, II–199–II–207.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research, 13*, 281-305. https://doi.org/10.5555/2188385.2188395

Chen, Y., Mensah, S., Ma, F., Wang, H., & Jiang, Z. (2021). Collaborative filtering grounded on knowledge graphs. *Pattern Recognition Letters*, *151*, 55-61. https://doi.org/https://doi.org/10.1016/j.patrec.2021.07.022

Cho, Y. H., Kim, J. K., & Kim, S. H. (2002). A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, *23*(3), 329-342. https://doi.org/https://doi.org/10.1016/S0957-4174(02)00052-0

Frazier, P. I. (2018). *A Tutorial on Bayesian Optimization*. https://arxiv.org/abs/1807.02811

Garnett, R. (2023). gaussian processes. In *Bayesian Optimization* (pp. 15-44). Cambridge University Press.

Guido, R., Groccia, M. C., & Conforti, D. (2023). A hyper-parameter tuning approach for cost-sensitive support vector machine classifiers. *Soft Computing*, *27*(18), 12863-12881. https://doi.org/10.1007/S00500-022-06768-8/FIGURES/7

Han, S. C., Lim, T., Long, S., Burgstaller, B., & Poon, J. (2021). GLocal-K: Global and Local Kernels for Recommender Systems. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3063-3067. https://doi.org/10.1145/3459637.3482112

Hoos, H. H. (2012). Automated Algorithm Configuration and Parameter Tuning. In E. and S. F. Hamadi Youssef and Monfroy (Ed.), *Autonomous Search* (pp. 37-71). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-21434-9_3

Horasan, F., Yurttakal, A. H., & Gündüz, S. (2023). A novel model based collaborative filtering recommender system via truncated ULV decomposition. *Journal of King Saud University - Computer and Information Sciences*, *35*(8), Article 101724. https://doi.org/https://doi.org/10.1016/j.jksuci.2023.101724

Imani, M., Imani, M., & Ghoreishi, S. F. (2022). Bayesian Optimization for Expensive Smooth-Varying Functions. *IEEE Intelligent Systems*, *37*(4), 44-55. https://doi.org/10.1109/MIS.2022.3163227

Khaledian, N., & Mardukhi, F. (2022). CFMT: a collaborative filtering approach based on the nonnegative matrix factorization technique and trust relationships. *Journal of Ambient Intelligence and Humanized Computing*, *13*(5), 2667-2683. https://doi.org/10.1007/s12652-021-03368-6

Kim, D., & Suh, B. (2019, September). Enhancing VAEs for collaborative filtering: flexible priors & gating mechanisms. *Proceedings of the 13th ACM Conference on Recommender Systems*. https://doi.org/10.1145/3298689.3347015

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp.426-434). https://doi.org/10.1145/1401890.1401944

Koren, Y. (2009). *The bellkor solution to the netflix grand prize*. https://api.semanticscholar.org/CorpusID:6114578

Lam, C. P. (2005). Collaborative Filtering Using Associative Neural Memory. In S. S. Mobasher Bamshad and Anand (Ed.), *Intelligent Techniques for Web Personalization* (pp. 153-168). Springer Berlin Heidelberg.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband. *The Journal of Machine Learning Research, 18*, 1-52. https://doi.org/10.5555/3122009.3242042

Li, Z., Huang, J., & Zhong, N. (2017). Exploiting user and item embedding in latent factor models for recommendations. *Proceedings of the International Conference on Web Intelligence* (pp.1241-1245). https://doi.org/10.1145/3106426.3109437

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, *7*(1), 76-80. https://doi.org/10.1109/MIC.2003.1167344

Molaei, S., Havvaei, A., Zare, H., & Jalili, M. (2021). Collaborative Deep Forest Learning for Recommender Systems. *IEEE Access*, *9*, 22053-22061. https://doi.org/10.1109/ACCESS.2021.3054818

Morita, Y., Rezaeiravesh, S., Tabatabaei, N., Vinuesa, R., Fukagata, K., & Schlatter, P. (2022). Applying Bayesian optimization with Gaussian process regression to computational fluid dynamics problems. *Journal of Computational Physics*, *449*, Article 110788. https://doi.org/10.1016/j.jcp.2021.110788

Rendle, S. (2012). Factorization Machines with libFM. *ACM Trans. Intell. Syst. Technol.*, *3*(3), 1-22 https://doi.org/10.1145/2168752.2168771

Rendle, S., Zhang, L., & Koren, Y. (2019). On the Difficulty of Evaluating Baselines: A Study on Recommender Systems. *ArXiv*, *abs/1905.01395*. https://api.semanticscholar.org/CorpusID:146120960

Salakhutdinov, R., & Mnih, A. (2008). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. *Proceedings of the 25th International Conference on Machine Learning* (pp. 880-887). https://doi.org/10.1145/1390156.1390267

Shenbin, I., Alekseev, A., Tutubalina, E., Malykh, V., & Nikolenko, S. I. (2020). RecVAE: A New Variational Autoencoder for Top-N Recommendations with Implicit Feedback. *Proceedings of the 13th International Conference on Web Search and Data Mining* (pp. 528-536). https://doi.org/10.1145/3336191.3371831

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, 2951-2959.

Srikumar, K. (2004). A Framework of Agent-Based Personalized Recommender System for E-Commerce. *South Asian Journal of Management, 11*, Article 66. https://api.semanticscholar.org/CorpusID:166413341

Steck, H. (2019, May). Embarrassingly Shallow Autoencoders for Sparse Data. *The World Wide Web Conference*. https://doi.org/10.1145/3308558.3313710

Szabó, P., & Genge, B. (2020). Hybrid Hyper-parameter Optimization for Collaborative Filtering. *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 210-217. https://doi.org/10.1109/SYNASC51798.2020.00042

Tim, H., Manoj, K., Holger, N., Gilles, L., & Shcherbaty, I. (2021). *scikit-optimize/scikit-optimize (v0.9.0)*. https://scikit-optimize.github.io/stable/

Xia, H., Luo, Y., & Liu, Y. (2021). Attention neural collaboration filtering based on GRU for recommender systems. *Complex & Intelligent Systems, 7*(3), 1367-1379. https://doi.org/10.1007/s40747-021-00274-4

Yan, S. (2014). A Collaborative Filtering Recommender Approach by Investigating Interactions of Interest and Trust. In T. and L. H. Sun Fuchun and Li (Ed.), *Knowledge Engineering and Management* (pp. 173-188). Springer Berlin Heidelberg.

Zhang, R., Liu, Q., Chun-Gui, Wei, J.-X., & Huiyi-Ma. (2014). Collaborative Filtering for Recommender Systems. *2014 Second International Conference on Advanced Cloud and Big Data*, 301-308. https://doi.org/10.1109/CBD.2014.47

TỐI ƯU HÓA HỆ THỐNG ĐỀ XUẤT: TÍCH HỢP LỌC CỘNG TÁC VỚI CÁC KĨ THUẬT BAYESIAN VÀ GAUSSIAN

Nguyễn Tuấn Anh

Trường Đại học Ngoại ngữ – Tin học Thành phố Hồ Chí Minh, Việt Nam

Tác giả liên hệ: Nguyễn Tuấn Anh – Email: anhnt1@huflit.edu.vn

Ngày nhận bài: 10-7-2024; Ngày nhận bài sửa: 07-02-2025; Ngày duyệt đăng: 08-7-2025

TÓM TẮT

Một phương pháp thường được sử dụng trong hệ thống đề xuất là lọc cộng tác (Collaborative Filtering - CF). Tinh chỉnh các siêu tham số (hyperparameters) của các thuật toán CF vẫn là một công việc khó khăn ngay cả với những khám phá mới trong việc mô hình hóa người dùng và các sản phẩm/dịch vụ. Nghiên cứu này đề xuất một phương pháp thay thế cho công việc này thông qua tối ưu hóa Bayesian sử dụng quá trình ngẫu nhiên Gaussian trong quá trình thay đổi các siêu tham số. Phương pháp này giảm thời gian và công sức cần thiết cho việc tinh chỉnh thủ công bằng cách tự động điều chỉnh các siêu tham số cho hai thuật toán lọc cộng tác cơ bản (và đơn giản) trên ba tập dữ liệu phổ biến: Netflix Prize, Movielens 1M và Movielens 10M. Do đó, nó có thể giúp các nhà thực hành cải thiện hiệu suất của hệ thống đề xuất, đồng thời rút ngắn đáng kể thời gian và công sức dành cho việc tinh chỉnh hệ thống của họ.

Từ khóa: tối ưu hóa Bayesian; lọc cộng tác; Movielens; Netflix Prize