Enhancing Recommender Systems: A New Approach Using Collaborative Filtering with Bayesian Optimization and Gaussian Processes

Tuan-Anh Nguyen
Ho Chi Minh City University of Foreign Languages - Information Technology, Ho Chi Minh City, Vietnam
Correspondence: Tuan-Anh Nguyen. Email: anhnt1@huflit.edu.vn
Communication: received 19/11/2024, revised 23/02/2025, accepted 25/04/2025
DOI: 10.32913/mic-ict-research.v2025.n2.1348

***Abstract*: Recommendation systems frequently make use of collaborative filtering (CF). CF derives its strength from the fact that in order to profile its users, it does not require a lot of knowledge about them, but it depends on earlier users' ratings in which they were choosing products to recommend. Although there has been progress in modeling users as well as items, calibrating CF algorithms' hyperparameters will continue to be a difficult task. This paper has come up with a new way of doing this by the use of Bayesian optimization with Gaussian processes throughout hyperparameter tuning. The method in question automatically works on hyperparameters to make two fundamental and straightforward CF algorithms have solid results against three famous datasets (Netflix Prize, MovieLens 1M, MovieLens 10M), and two other datasets (Douban and Jester dataset 2). This solution not only exhibits solid performance in controlled experiments but also provides a simplified approach for practitioners, minimizing time-intensive manual adjustments while ensuring low overhead, therefore necessitating relatively low computational and developmental resources. This results in expedited deployment and simplified integration into practical systems, hence enhancing the dependability and scalability of recommendation engines.**

Keywords: *Bayesian optimization, collaborative filtering, Gaussian processes, recommender system.*

I. Introduction

Today consumers face a broad selection of online platforms offering e-commerce services and content. This multitude of options makes it challenging to match products with individual consumer tastes. All corporations operating in diverse industries such as Amazon, Google, and Netflix have found that personalized recommendation systems are of absolute importance to them. By employing these technologies, organizations can personalize their products to suit different consumer tastes therefore increasing their levels of satisfaction and loyalty [1–3].

Collaborative filtering (CF) is now the major method used for creating recommendation systems. In contrast to systems that require detailed user profiles, CF is based strictly on past user activity such as ratings and purchases. Therefore it does not need any domain-specific knowledge or collection of vast amounts of data to give meaningful recommendations [4]. Furthermore, through user behavior, CF is able to unearth hidden trends or customer preferences that are not captured by traditional approaches. Such efficiency has been demonstrated in Amazon's and Netflix's commercial use [5–7].

Despite the progress made in modeling techniques, hyperparameter tuning in CF systems has largely remained a human activity. Often enough, researchers base their hyperparameter choice on experience and experimentally established criteria [8]. This is usually time-consuming and not always the best way; for instance, tuning hyperparameters was crucial to success in the Netflix Prize competition [9, 10].

Several effective applications for hyperparameter optimization have shown favorable outcomes using Bayesian optimization with Gaussian priors [11–13]. Nonetheless, recent advancements in intricate deep learning architectures [14, 15] have highlighted the expanding potential of neural networks for CF, although with heightened computing demands and considerable tuning complexity. Conversely, our methodology emphasizes hyperparameter optimization in two principal collaborative filtering algorithms using Bayesian optimization utilizing Gaussian processes. We want to automate the tuning process to reconcile manual approaches with resource-intensive deep learning models, minimizing human interaction while possibly improving performance on commonly used datasets. This study illustrates the effectiveness of our methodology, yielding competitive outcomes on the Netflix Prize,

MovieLens 1M, MovieLens 10M, Jester and Douban datasets (refer to our source code in the GitHub repository https://github.com/anhnt1/netflix).

The paper is organized as follows. In the next section, this paper shall review the extant literature focusing particularly on the manual tuning of hyperparameters and how previous research dealt with modeling users and items along with their interactions. The following part will then provide a rationale for employing Bayesian optimizations through Gaussian processes. Section 4 explains our methodology; while Section 5 presents results from running both CF algorithms on the Netflix Prize, MovieLens 1M, Movialens 10M, Jester and Douban datasets. Finally, we summarize the findings.

II. Literature Review

Notable has been the recommendation systems research with a major focus on modeling people, items and their interactions. Also, in this area, a significant study has been carried out into a number of ways to improve its efficiency leading to a large number of research works in this direction. CF, on the other hand, has been an overarching technique in such studies. An early paper by Koren et al. [9] among others was based on matrix factorization methods applied for user-item interaction. These models represent the interaction matrix between users and items as latent feature vectors which are then employed to predict future preferences of users towards different items. To build more complicated models, several research works have introduced hybrid techniques that combine content-based filtering with CF for exploiting both user behavior and item properties [16, 17]. In the CF industry, these are standard matrix factorization techniques, without implying they are easy – such decompositions have high computational complexity; yet they are scalable, for instance, common Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF). This necessitated the Truncated–ULV decomposition technique (T–ULVD) as an efficient computational alternative for scaling [18]. Data sparsity is overcome through knowledge-enriched CF systems reported by [19] – in order to represent interactions between users and items, they model user attributes as well as item attributes obtained by knowledge graphs (KGs). The methodology followed in [20] incorporates trust links between users, thus addressing cold-start issues as well as data sparseness in CF models. Deep learning methods have recently been used to improve the performance of CF systems since they allow for their use with data sparsity. Advanced complex user-item interactions and latent attributes can now be learned by employing methods like Collaborative Deep Forest Learning (CDFL) and Multi-model deep learning. These incorporate deep neural networks for improved handling of data sparseness and cold-start problems in addition to enhancing recommendation accuracy by combining traditional CF methods with deep neural networks [14, 15]. Attention mechanisms and recurrent neural networks enable the capture of relevant information and long-range dependencies, thereby enriching feature representation hierarchies with more contextual information [21]. To estimate missing values in a user-item interaction matrix for better prediction of users' preferences, some models use Bayesian statistics for matrix completion or imputation via matrix factorization [8, 22, 23]. Consequently, just like any other study that focuses on the estimation of parameters only for CF models, their main goal is still the same.

Even with the substantial progress made in developing advanced CF models over the last two decades, hyperparameter optimization remains crucial for their improvement. In general, traditional approaches usually entail manual tuning that often takes up time and may lead to suboptimal outcomes due to the high-dimensional search space as well as complex interactions between these parameters. This complicates the process of tuning for model correctness, generalization, and computational efficiency [8, 24]. They could be learning rates, regularization coefficients, embedding dimensions, or neural-networks-based considerations in architectures [25, 26]. The manual approach is particularly challenging for newcomers in the area who lack adequate expertise.

To address hyperparameter tuning difficulties, automated approaches such as Bayesian optimization provide an alternative that enhances model accuracy while reducing modeling time by doing all the hard work for one. These methods employ optimization and machine learning in exploring and evaluating hyperparameter space and are often more effective than manual nursing [27, 28]. Various practical examples demonstrated the effectiveness of automated hyperparameter tuning techniques [11–13] and this can be a fruitful area of research in recommender systems. This research work is more certain about the practical significance of recommendation systems and CF models in developing automatic parameter-tuning techniques with our access to actual performance results. This paper sets out to explore an alternative way of tuning hyperparameters that is simple and applicable in CF systems with the potential for good performance. This might reduce the dependence on human effort as well as improve the algorithm's performance across diverse scenarios. Making it easier for more practitioners to deploy and customize successful recommender systems could potentially drive innovation within this realm by simplifying model optimization efforts for all

involved parties.

III. Bayesian Optimization with Gaussian Processes

1. Bayesian Optimization for Hyperparameter Tuning in CF

Hyperparameter tuning in CF is a good match for Bayesian optimization using Gaussian processes (GPs) because it effectively manages those expensive evaluations at the first instance. In hyperparameter tuning especially for CF models, Bayesian optimization focuses on maximizing (or minimizing) functions that are costly to evaluate. For example, projecting the behavior of the function using less number of evaluations facilitates an effective exploration within the parameter space by using GPs [29]. Combining Bayesian optimization with GPs may instead offer a more realistic description of the underlying function which could result in increased expressiveness of surrogate functions approximating complex models like those of CFs [29].

Secondly, it gives an ability to measure uncertainty. Indeed, GPs themselves can indicate how uncertain they are about their own projection hence serving as the basis for choices on the next sample's future location. This is valuable when the main goal of tuning hyperparameters is finding the most suitable values with few evaluations [30]. Because acquisition functions based on uncertainty increase convergence rates and accuracy in surrogate models which lowers risks associated with overfitting and in turn enhances CF task performances [30].

Thirdly, it enables adaptive exploration and exploitation in a single breath. Natural exploration-exploitation tradeoff within Bayesian optimization can enable one to move effectively over the hyperparameter space. The use of acquisition functions in directing the search process helps to achieve this balance [29]. Common random numbers can enhance the efficiency of Bayesian optimization by reducing variance and making the optimization process more robust [31].

In conclusion, using Gaussian priors for Bayesian optimization provides an appropriate approach for tuning hyperparameters in CF models because it manages costly evaluations effectively, gauges uncertainty, and balances exploration versus exploitation leading to precise hyperparameter adaptation as well as improved CF performance metrics.

For subsequent paragraphs, this paper plans to provide an overview of the Gaussian process which is used as a prior in Bayesian optimization. This will assist one grasp how this method is implemented and certain techniques behind its capabilities.

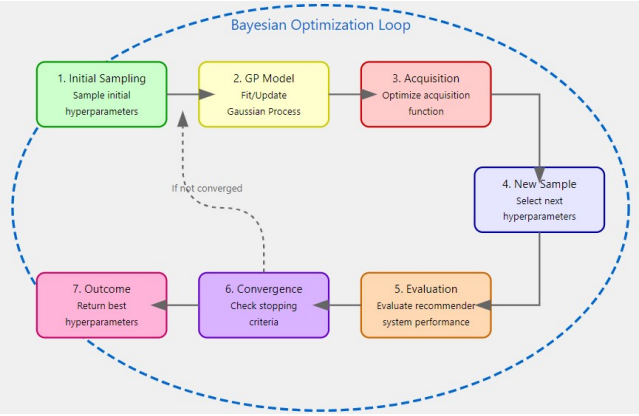

Figure 1. Bayesian Optimization Process for CFs.

2. High-Level Representation of Bayesian Optimization Process for Recommender Systems

Figure 1 illustrates the iterative characteristic of Bayesian optimization via a loop covering the whole process. Each iteration adjusts according to all prior assessments, gradually refining the best hyperparameters. When convergence has not been attained, the dashed arrow coming from the convergence check shows the iterative feature of the process.

1) Initial sampling: First, the method samples starting hyperparameters from the hyperparameter space. In this step, a limited number of hyperparameter combinations is either randomly or methodically chosen from the comprehensive search space. This first sample offers an initial insight into the model's potential responses to various parameter combinations.
2) Gaussian Process model: Second, Bayesian statistics is used in this step. It uses all available data (including the performance results from the initial samples) to fit a GP model, and new observations update this model in the next iterations. This step updates our beliefs to form a posterior distribution and uncertainty estimates. This GP serves as a proxy for the underlying (unknown) performance function, enabling us to estimate both the anticipated performance value (the mean) and the corresponding uncertainty at each location inside the hyperparameter space.
3) Acquisition function: Third, the present GP model is used to build an acquisition function that directs the selection of the subsequent hyperparameters for evaluation. This function addresses two objectives: (i) the exploitation of regions anticipated to provide high performance, and (ii) the investigation of areas characterized by significant model uncertainty. Typical acquisition functions include Probability of

Improvement (PI), Expected Improvement (EI), and Lower Confidence Bound (LCB). Readers should see the section III.3 for more details of these three acquisition functions.

4) New sample: Fourth, a new hyperparameter configuration is chosen from the search space based on the acquisition function's suggestions. This decision generally seeks to optimize possible improvement while also considering areas with significant uncertainty.
5) Evaluation: Fifth, the selected hyperparameter configuration is then assessed on the recommender system. This assessment entails training or refining the model using a dataset (e.g., Netflix Prize, MovieLens,...) and evaluating performance measures like as RMSE or MAE.
6) Convergence check: Sixth, subsequent to the new assessment, the algorithm ascertains if any stopping requirements are met – for example, if the enhancement in performance fails to surpass a threshold or a certain number of repetitions has been achieved.
7) Outcome: Seventh, upon satisfying the convergence conditions, the procedure concludes, yielding the optimal hyperparameter set found to date. Consequently, the loop resumes at Step 2, integrating the newly acquired data into the GP model, and a new iteration begins.

By depicting Bayesian optimization iteratively, each iteration enhances the surrogate GP model, aiming to quickly ascertain ideal hyperparameters while reducing the frequency of costly model assessments. This iterative technique is particularly advantageous for recommender systems, since hyperparameter optimization often requires substantial computation resources.

3. An Overview of Bayesian Optimization Utilizing Gaussian Processes

Assume that one has some function f that maps a set of hyperparameters $\mathcal{X}$ into a real value. However, there are instances when optimizing f is a daunting task, especially when function evaluations consume lots of time. In such situations, Bayesian optimization is a viable alternative. A detailed explanation of the basic principles underpinning Bayesian optimization can be found in [32, 33], below we summarize this explanation:

- The unknown function f is modeled as a probability distribution over functions. Probabilistically, this allows us to include prior knowledge about f. Usually, a Gaussian process serves as the commonest approach to use for this purpose.
- By evaluating the function at a few selected points $x_1, x_2, \ldots, x_D$, we get corresponding values $f(x_1), f(x_2), \ldots, f(x_D)$ which are regarded as observed data within the probabilistic model.
- By modeling f as a probability distribution, we may compute the conditional probabilities of $f(x)$ given observed function values at points $x_1, x_2, \ldots, x_D$ for example. Hence, in terms of computational costs, using a Gaussian process for this probabilistic model makes it cheaper to calculate $f(x)$ than to evaluate directly some of the desired points.
- The conditional distribution can assist in producing estimates of $f(x)$ at unseen coordinates while simultaneously guiding future choices during the optimization procedure.
- The most promising point for subsequent evaluation is determined by an acquisition function. This function relies on $f(x)$'s predicted mean and standard deviation, as well as y_{best}, which is presently the highest (or "best") observed value. Common acquisition functions include:

  1) Probability of improvement (PI): This function ranks points that have good chances of being above the current best. It emphasizes prioritizing points with a greater likelihood of enhancing the present optimal value. Demands low computation but may exhibit excessive caution in using established advantageous areas and neglect to investigate adequately when the likelihood of progress in uncharted territories seems minimal [34–36].
  2) Expected improvement (EI): This function works to maximize the possibility of improving the optimal value. It balances exploitation and exploration by assigning weights to each prospective enhancement based on its likelihood; often demonstrates effective performance in practice. It may incur further processing costs compared to PI when assessing the anticipated benefit at each potential location [34, 35].
  3) Lower confidence bound (LCB): It considers both mean and uncertainty thereby balancing exploration with exploitation. It directly regulates exploration by imposing more penalties on high-variability areas; straightforward formulation integrating mean and uncertainty. Its reactivity to the selected confidence parameter may result in excessively cautious or too aggressive searches, affecting convergence rates [34, 37].

- The primary advantage of Bayesian optimization is its ability to utilize all data available. It differs from methods that restrict themselves solely to recent or historical information by providing that each subse-

quent query incorporates every other one that occurred earlier. This makes it more efficient and accurate.

Finally, CF models' hyperparameter tuning using Bayesian optimization with GPs provides a solid and efficient framework. It typically balances exploration and exploitation through the use of probabilistic modeling as well as acquisition functions delivering more accurate and effective optimization. This means that it is an appropriate method for complex models that include high computational costs (expensive) while also enabling tracking of uncertainty that could help improve the overall performance of recommendation systems.

More information on the datasets under consideration will be provided in the next section. This will be followed by two well-known algorithms used in recommender systems and how they have been applied in our work to address the cold-start problem, limitations of using GP models for hyperparameter optimizations and sparsity in datasets.

IV. Experiments

1. Baselines and Rationale

To comprehensively assess the efficacy of our proposed method, we juxtapose it with many cutting-edge algorithms, the majority of which use neural network topologies. Tables I and II encapsulate these models, including Neural Collaborative Filtering, deep factorization machines, and several other neural-based methodologies. We have furthermore included a traditional but highly competitive baseline, SVD++, recognized for using both explicit and implicit input to enhance user preference modeling [9, 10].

The selection of these baselines fulfills many objectives:

1) Current trends representation: Neural-based algorithms have established themselves as standard in the realm of recommendations. Models like Neural CF use deep neural networks to comprehend intricate user-item interactions that conventional methods may neglect.
2) Comparative analysis: The inclusion of SVD++ highlights the disparities in performance between solely factorization-based approaches, which include minor changes to account for implicit feedback, and more contemporary neural solutions. This enables readers to discern if performance improvements arise from more complex structures, superior hyperparameter optimization, varied regularization techniques, or a confluence of variables.
3) Empirical significance: Research and contests (e.g., Netflix Prize) continuously underscore SVD++ and neural-based models as formidable challengers, with documentation from many sources (for instance, [8]) routinely attesting to their dependability and prevalence. This guarantees that the benchmarks represent acknowledged, real-world best practices.
4) Contextualizing our findings: By doing a comprehensive comparison – from traditional factorization to contemporary neural benchmarks – we provide a clear perspective on relative advantages and disadvantages, including tuning overhead, and scalability considerations. This thorough assessment offers an enhanced comprehension of how our methodology aligns with prominent and extensively used recommender system frameworks.

These baselines not only demonstrate the benefits and drawbacks of our strategy but also situate our study within the larger context of current collaborative filtering research.

2. The Datasets

The Netflix Prize competition included evaluating algorithms using an extensive dataset of over 100 million movie ratings submitted by anonymous consumers. Additionally, an extra dataset known as the Quiz set comprises 1.4 million current ratings for interim evaluation. A withheld Test set of comparable size was ultimately used to determine the competition's winner. This work uses identical datasets for training, testing, and validation, in contrast to other studies [38, 39] that adopt a modified approach by binarizing ratings and filtering users, or by randomly partitioning the original training set to generate supplementary datasets [40]. The MovieLens 1M and 10M datasets are also widely used for assessing the efficacy of recommender systems. For these two datasets, as well as the Jester dataset 2, we randomly partitioned the data in a 90:10 ratio for training and testing, a percentage often used in research, as referenced in [8]. For the Douban dataset, we use the preprocessed subsets and splits the same as the authors of [41] used. The density of the Netflix Prize, MovieLens 1M, MovieLens 10M, Jester and Douban datasets are 0.0118, 0.0447, 0.0131, 0.2595 and 0.0152 accordingly.

3. Two CF Algorithms

The two basic CF algorithms from [10] provided in this section serve as the basis for more elaborate models. In the first algorithm, users and items are embedded into a shared latent factor space. The idea is to obtain underlying preferences by mapping the users and the items on corresponding vectors. Specifically, every user u corresponds to a vector $\mathbf{p}_u \in \mathbb{R}^n$ called user-factor vector, whereas each item i correlates with a vector $\mathbf{q}_i \in \mathbb{R}^n$ named item-factor vector. The predicted ratings denoted as $\hat{r}_{ui}$ are calculated as the inner product of these vectors, namely $\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$.

Optimization of the latent factors aims at minimizing the difference between predicted ratings and actual ratings r_{ui} during the process:

minimize $\sum_{(u,i)\in\mathcal{K}} \left(r_{ui} - \mathbf{p}_u^T \mathbf{q}_i\right)^2 + \lambda \left(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2\right)$.

$\mathcal{K}$ is a set defined as $\mathcal{K} = \{(u,i) \mid r_{ui} \text{ is known}\}$; 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 $\mathcal{K}$. For every rating r_{ui}, we make changes to model parameters counter to gradients given by;

- $\mathbf{p}_u \leftarrow \mathbf{p}_u + l_r \cdot (e_{ui} \cdot \mathbf{q}_i - \lambda \cdot \mathbf{p}_u)$
- $\mathbf{q}_i \leftarrow \mathbf{q}_i + l_r \cdot (e_{ui} \cdot \mathbf{p}_u - \lambda \cdot \mathbf{q}_i)$

where l_r is the learning rate.

The second CF algorithm builds on top of the first CF algorithm by adding baseline estimation for handling the underlying user and item biases present in CF data. These biases occur as users or items with systematic tendencies toward higher or lower ratings. To tackle these problems, baseline estimates are utilized: μ symbolizes the general mean rating while b_u and b_i represent user and item deviations respectively from the unbiased part $\mu + \mathbf{p}_u^T \mathbf{q}_i$. The prediction can then be computed as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i$$

The associated loss is minimized to estimate parameters:

$$\text{minimize} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i - \mathbf{p}_u^T \mathbf{q}_i)^2 + \lambda(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 + b_u^2 + b_i^2)$$

Similar to the first algorithm, we loop over all known ratings in the data set $\mathcal{K}$. For every rating r_{ui}, model parameters are updated like this:

- $b_u \leftarrow b_u + l_r \cdot (e_{ui} - \lambda \cdot b_u)$
- $b_i \leftarrow b_i + l_r \cdot (e_{ui} - \lambda \cdot b_i)$
- $\mathbf{p}_u \leftarrow \mathbf{p}_u + l_r \cdot (e_{ui} \cdot \mathbf{q}_i - \lambda \cdot \mathbf{p}_u)$
- $\mathbf{q}_i \leftarrow \mathbf{q}_i + l_r \cdot (e_{ui} \cdot \mathbf{p}_u - \lambda \cdot \mathbf{q}_i)$

The algorithms iterate through $\mathcal{K}$ many times and terminate when the loss fails to decrease significantly after multiple iterations.

4. Limitations in GP Models for Hyperparameter Tuning

While GP models are helpful in balancing exploration and exploitation during hyperparameter optimization, we recognize many limitations intrinsic to this methodology. Initially, GP-based methodologies may encounter scaling challenges in high-dimensional parameter spaces, since matrix operations often become computationally unfeasible [29, 30]. The smoothness assumption in GPs may be too limiting for some CF algorithms, especially those exhibiting significant performance declines due to tiny hyperparameter adjustments [29]. Third, the optimization process is significantly affected by initial hyperparameter samples, and inadequate seeding may confine the search to local minima [42].

This work prioritized practical constraints – specifically time and computational resources – over a comprehensive exploration of advanced GP modifications (e.g., sparse approximations or customized GP models) that may alleviate some of these issues. Our future research agenda includes the investigation of scalable variations of GP models designed for high dimensions; alternative GP models that are more appropriate for sudden fluctuations in CF performance, and enhanced methods for initializing GP hyperparameter searches, particularly under cold-start scenarios.

Investigating these pathways may enhance the efficacy of GP-based Bayesian optimization in bigger and more complex CF contexts. Nonetheless, implementing these improvements requires substantial further investment in methodological advancement and computer capacity, which we will address in further research.

5. The Cold-Start Problem

The accuracy of recommendations and user contentment in collaborative filtering systems are particularly susceptible to cold-start scenarios, characterized by a lack of previous data for new users or products. Numerous sophisticated techniques, including meta-learning, deep learning and hybrid approaches [43–47] have been suggested to tackle this issue. Nevertheless, these methodologies often need considerable data and computing resources [48, 49].

Two CF algorithms used in this research utilize global averages to address cold-start cases during validation and testing. This strategy offers a rapid and computationally economical baseline, although it unavoidably reduces the intrinsic complexity of varied user behavior [44, 45]. This dependence on global averages neglects individual- or item-specific intricacies and may generate bias by presuming uniform user behavior, ignoring personal preferences. This limitation belongs to these two CF algorithms and not to the proposed technique in this research.

Owing to practical limitations – including restricted time, computer resources, and research scope – we have not explored more sophisticated cold-start alternatives. Our primary objective is to provide an efficient method instead of an all-encompassing solution that addresses every facet of the cold-start issue. Future study will investigate hybrid or meta-learning methodologies to enhance the personalization

of early-stage suggestions for new users and enhanced user/item initialization techniques that surpass global averages.

In conclusion, we recognize that the cold-start approach used in the two CF algorithms serves as an initial framework – it sacrifices detail for operational efficiency. Significant progress is achievable and will be sought in future endeavors as resources and project schedules permit.

6. Sparse Data Issues

This subsection outlines the impact of data sparsity – a prevalent issue in CF – on recommendation systems.

Data sparsity impact: this issue is considerably prominent in CF, impacting the stability of user-item interaction matrices with scarce data. With limited data points, there is not enough data that may accurately portray underlying preferences and latent characteristics. Consequently, this gives larger prediction errors with higher RMSE values, as shown by comparison tests for traditional and neural modes [50, 51].

Algorithm adapability: traditional means, for instance, the SVD++ method, otherwise manage sparsity well by integrating both the implicit and data inputs. However, with the most severe scarcity of user-item interactions, the predictive capability of these methods will similarly be significantly reduced [50]. Gaussian priors in the Bayesian hyperparameter tuning may improve estimates of their parameters, though the inherent scarcity of data will be blocking an array of possible applications [51].

Error propagation in latent factor models: when data sparsity prevents the learning of complex user-item interactions, the resultant prediction errors will propagate via the neural network. Deep learning methods like Neural Collaborative Filtering were designed to recognize more subtle patterns, but the advantage of these sophisticated models is not fully employed in sparse datasets, creating a mismatch between the theoretical potential and actual usefulness of the model [52].

In conclusion, sparse interaction matrices diminish the efficacy of even cutting-edge CF methods by hindering the learning process for latent user-item interactions. This leads to an increased RMSE and decreased suggestion accuracy.

The results of Bayesian optimization using Gaussian processes to choose the hyperparameters of the two methods on the experiment datasets will be shown in the next section.

V. Results

1. Experimental Setup and Results

In our experiments, this work used scikit-optimize [53] for hyperparameter optimization, which implements

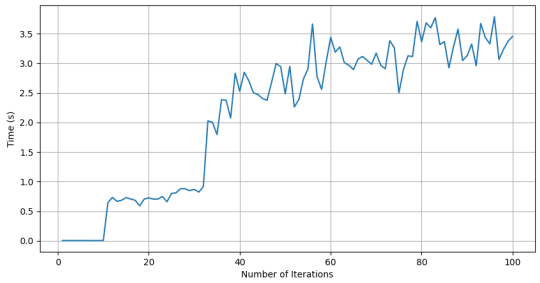

Figure 2. Bayesian optimization time.

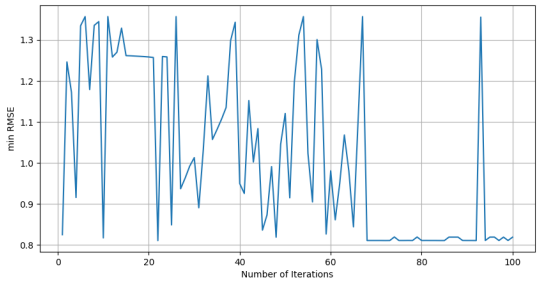

Figure 3. RMSE convergence of the first algorithm on the Netflix Prize dataset.

Bayesian optimization. Specifically, we use its gp_minimize function with default values for all possible parameters. Especially, we use the "gp_hedge" value (the default value) for the acquisition function parameter. This value will direct the gp_minimize function to probabilistically choose one of the three acquisition functions (e.g., PI, EI, LCB) at every iteration based on the gain provided by each function, with the trade-off being variability in convergence rates. Note that in this research, we do not try to achieve state-of-the-art performance as well as the effect of the chosen acquisition function on the performance of CF algorithms; instead, we focus on practical usages of Bayesian optimization with Gaussian priors in recommendation systems by experiment-

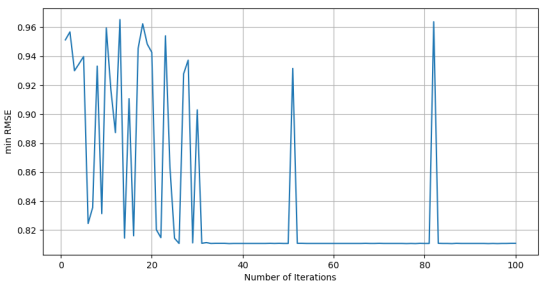

Figure 4. RMSE convergence of the second algorithm on the Netflix Prize dataset.

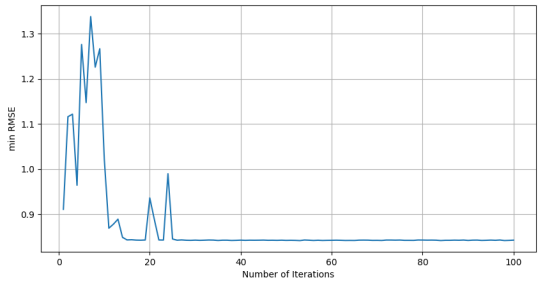

Figure 5. RMSE convergence of the first algorithm on the MovieLens 1M dataset.

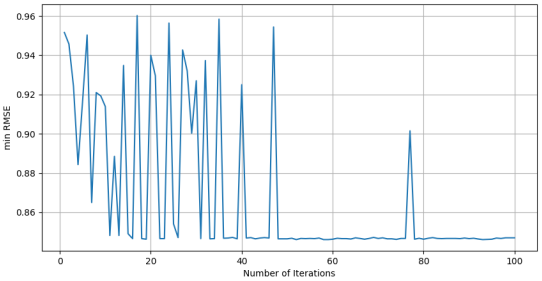

Figure 6. RMSE convergence of the second algorithm on the MovieLens 1M dataset.

ing with two simple and basic CF algorithms and some popular datasets. This aspect also signifies the simplicity of deployment and integration of the suggested approach into real-world systems.

In the experiments, we set the initial learning rate l_r to 5e-3, the range of the regularization parameter λ from 1e-4 to 1.0, and the dimension n range from 50 to 500. If the loss reduction is less than 1e-4 after two iterations, the learning rate is reduced by 10 times. The parameter optimization process, which optimizes $b_u, b_i, \mathbf{p}_u, \mathbf{q}_i$, stops if the loss reduction remains below 1e-4 after five iterations (stopping criteria).

Table I
PERFORMANCE COMPARISON OF VARIOUS ALGORITHMS ON THE MOVIELENS 1M DATASET.

Algorithm	**RMSE (MovieLens 1M)**
LLORMA	0.833
I-AutoRec	0.831
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**

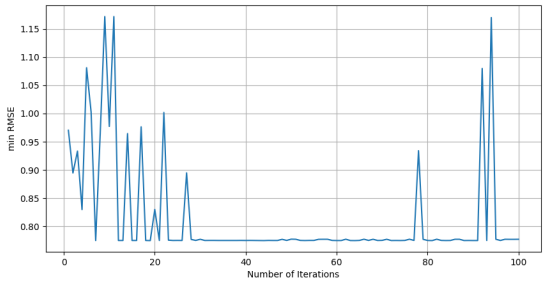

Figure 7. RMSE convergence of the first algorithm on the MovieLens 10M dataset.

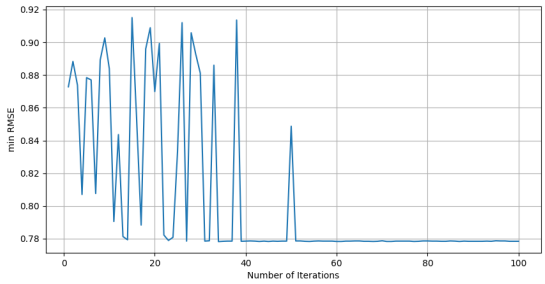

Figure 8. RMSE convergence of the second algorithm on the MovieLens 10M dataset.

Table II
PERFORMANCE COMPARISON OF VARIOUS ALGORITHMS ON THE MOVIELENS 10M DATASET.

Algorithm	**RMSE (MovieLens 10M)**
RSVD	0.8256
U-RBM	0.823
BPMF	0.8197
APG	0.8101
DFC	0.8067
Biased MF	0.803
GSMF	0.8012
SVDFeature	0.7907
ALS-WR	0.7830
I-AutoRec	0.782
LLORMA	0.7815
WEMAREC	0.7769
I-CFN++	0.7754
MPMA	0.7712
CF-NADE 2 layers	0.771
SMA	0.7682
GLOMA	0.7672
ERMMA	0.7670
AdaError	0.7644
MRMA	0.7634
SGD MF	0.7720
Bayesian MF	0.7633
Bayesian timeSVD	0.7587
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**

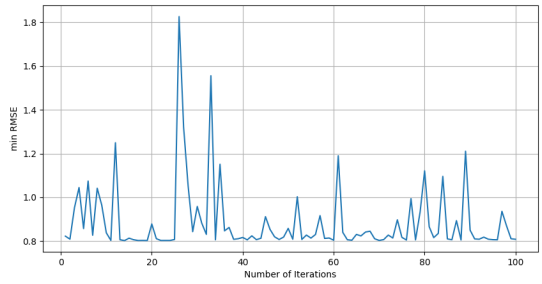

Figure 9. RMSE convergence of the first algorithm on the Douban dataset.

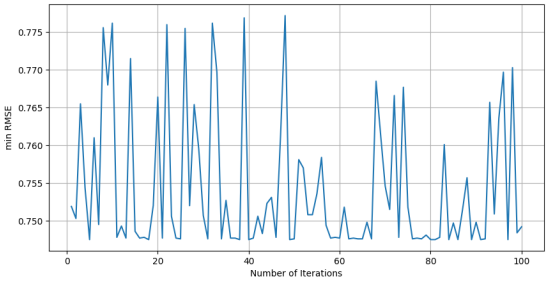

Figure 10. RMSE convergence of the second algorithm on the Douban dataset

Table III
PERFORMANCE COMPARISON OF VARIOUS ALGORITHMS ON THE DOUBAN DATASET.

Algorithm	RMSE (Douban)
GC-MC+Extra	0.734
GRAEM	0.732
SparseFC	0.730
IGMC	0.721
MG-GAT	0.727
GLocal-K	0.721
Our approach (first algo.)	**0.8055**
Our approach (second algo.)	**0.7499**

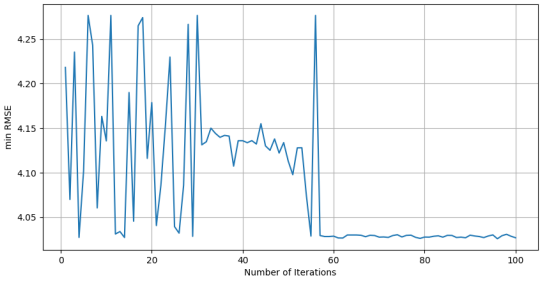

Figure 11. RMSE convergence of the first algorithm on the Jester dataset 2.

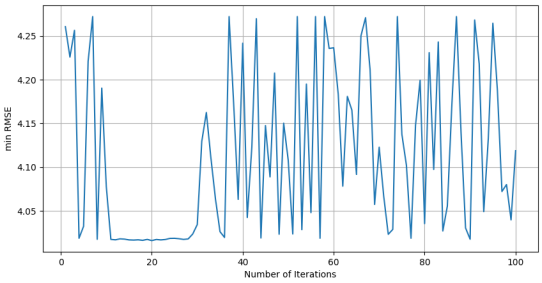

Figure 12. RMSE convergence of the second algorithm on the Jester dataset 2.

Table IV
PERFORMANCE COMPARISON OF VARIOUS ALGORITHMS ON THE JESTER DATASET 2.

Algorithm	RMSE (Jester)
SVD	4.497
SVD++	4.904
NMF	6.324
Slope One	4.253
Co-Clustering	4.292
Our approach (first algo.)	**4.036**
Our approach (second algo.)	**4.025**

Figures 3 and 4 illustrate the convergence of the minimum RMSE value after 100 iterations with the two algorithms, which achieved an RMSE 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 accordingly. As mentioned before, recent research works [38, 39] use a modified version of the datasets by binarizing ratings and filtering users, or randomly splitting the original training set to create additional datasets [40]. This prevents us from comparing the results, but the best-performing algorithm in the Netflix Prize contest had achieved an RMSE score of 0.8567 on the Test set [9], which incorporated SVD++ and many other techniques.

Table I, II, III and IV show the results from the two CF algorithms that this work used to calculate the errors on MovieLens 1M, MovieLens 10M, Douban and Jester datasets. For the MovieLens 1M dataset, other algorithms' performances are taken from [41] while those for the MovieLens 10M and Douban datasets are available from [8]. For the Jester dataset 2, we ran all possible algorithms on our computer using the scikit-surprise toolkit [54] because we could not find state-of-the-art results for this particular dataset. On the whole, the outcome is an indication that in hyperparameter tuning, Bayesian optimization using Gaussian priors is effective for both CF models.

2. Result Analysis

a) Factors Affecting Algorithm Performance

Most recommender algorithms are capable of high diversity concerning the nature of the datasets used for testing, thus letting several other factors become significant contributors to the reasons for the superiority of an algorithm over the lesser-performing ones:

- Sparsity and density of the dataset: the datasets vary to an enormous degree between sparsity. If we persist in calling MovieLens 10M dataset having higher sparsity (density = 0.0131) compared to Jester (more dense, with a density to approximate 0.2595), the algorithm using implicit feedback must handle this sparsity consideration. SVD++ or its variations, for instance, provide a good means for handling sparsity because they use both types of inputs, implicit and explicit, describing user preference in greater detail for opting for their sparser dataset. However, with denser datasets, simpler CF approaches – especially performance-optimized for sparseness – might be enough to predict user-item interactions with competitive error rates.
- Model complexity and feedback handling: advanced algorithms, mostly those based on neural networks (e.g., Neural Collaborative Filtering), are inherently very good at investigating intricate relationships between users and products. Yet, they hardly lead to significant performance improvements over traditional techniques when user interactions are scarce in the dataset. Both Netflix Prize and MovieLens 1M datasets, independently, could not raise a strong lead for neural management since the best-performing classical CF algorithms optimized via Bayesian optimization could reach RMSE levels that were competitive with these models. It, therefore, suggests that increasing model complexity using neural methods was required to model user-item interactions in the form of subtle patterns, often missed by less complex techniques.
- Hyperparameter adjustment and optimization techniques: hyperparameter tuning is a crucial step. The research revealed that using Bayesian optimization with GPs may efficiently fine-tune even fundamental CF methods. This is especially advantageous in practical situations where manual adjustment is laborious. The automatic exploration of hyperparameter space enables the model to adjust to the distinct noise and user-item interactions of each dataset. For instance, adjusting parameters allowed the simple CF models to exhibit strong performance across various datasets. The improvement in RMSE scores – exemplified by a result of 0.8103 on the Netflix Prize Test set – demonstrates the algorithm's capacity for self-adjustment based on dataset features.

The differentiation in overall performance of algorithms is based largely on the adequacy of the algorithm design to cope with implying sparsity, noise, and density of various datasets, and the inherent informative signaling using explicit and implicit feedback. This indicates that if algorithms offer mechanisms for efficient hyperparameter tuning – such as Bayesian optimization employing Gaussian processes – they may be rendered resilient to the specific obstacles presented by each dataset.

b) Emerging Trends in Algorithm Convergence and Performance

Several noteworthy features in convergence behavior and overall performance are demonstrated by the experimental results:

- Iterative convergence behavior:
  1) Through the use of GPs, the Bayesian optimization method propels the hyperparameters' progressive convergence across iterations. Up until a plateau is reached, the RMSE steadily improves as each iteration improves the model's predictions by updating the GP with fresh performance evaluations.
  2) In the studies that were reported, convergence was often seen in 50–90 repetitions. This shows that the hyperparameter tweaking rapidly focuses on efficient areas of the search space, even with the inherent unpredictability brought about by various acquisition functions.
  3) Convergence curves, which are shown in many figures, usually demonstrate a slow decline in RMSE over time. Both algorithms rapidly approach a comparable RMSE for some datasets, like the Netflix Prize dataset, indicating that even little hyperparameter changes can have a big effect when the tuning procedure is well calibrated.
- Performance trends across datasets:
  1) The performance results show competitive RMSE values across a variety of datasets, including Netflix Prize, MovieLens 1M, MovieLens 10M, Douban, and Jester.
  2) The main objective of this research work was to show the practical usability of Bayesian optimization in recommender systems. On the Netflix Prize dataset, both straightforward CF approaches tuned via Bayesian optimization achieved RMSE values around 0.81. This suggests that automated hyperparameter tuning

achieves competitive performance with a significant reduction in manual work.

3) The tuned models achieved RMSE levels in the range of 0.775 to 0.779 for bigger and more intricate datasets, such as MovieLens 10M. The studies demonstrate that even simple CF models gain a great deal from the optimization process, even though more complex approaches in the literature occasionally produce lower RMSE values.
4) The two algorithms' outputs differed more notably in the Douban and Jester datasets. For instance, both algorithms did well on the Jester dataset (RMSE approximately 4.02–4.04), but the second algorithm significantly improved on the Douban dataset (RMSE dropping from 0.8055 to 0.7499). This variety illustrates the ways in which dataset properties, including density, noise, and sparsity, impact convergence dynamics and overall performance.

In conclusion, the performance trends across datasets show that even simple CF algorithms can produce competitive results when combined with Bayesian optimization for hyperparameter tuning, while the convergence trends show an efficient iterative tuning process with quick progress in lowering RMSE. The importance of incorporating automated optimization techniques into recommender systems is highlighted by this convergence–performance link.

3. Scalability Analysis

a) Computational Impact of Bayesian Optimization with GPs

The complexity of Bayesian optimization with GPs is mainly defined by two tasks at each step: tuning Gaussian process hyperparameters and maximizing the acquisition function for new hyperparameter selection. The complexity of each task incur $O(\mathbb{N}^3)$ and $O(\mathbb{N}^2)$ computational cost respectively, where N is the number of data points [55]. Note that the optimization of acquisition functions (e.g., PI, EI, UCB) generally scales with the dimensionality d of the search space (not the size or the number of samples of datasets with which we conduct recommendations). For fixed or small values of d, this can be viewed as a lower-order term (e.g., $O(\mathbb{N}^2)$ in relation to GP inference [56]), and this characteristic is also mentioned in [12].

Our experience and results show that when you optimize parameters over one iteration for instance, it requires much more time than one iteration of the Bayesian optimization technique, hence the time taken searching for a new set of hyperparameters (a new exploration point) is almost negligible compared to the time taken while optimizing parameters for CF algorithm. Figure 2 displays the time taken to conduct our experiment on Bayesian optimization with Gaussian processes having two hyperparameters and lasting up to about 100 iterations. Instead, this time consumption is not influenced by the characteristics of the CF algorithms since the CF algorithms operate blindly within the Bayesian optimization process.

b) Scalability Strategies

While our present study does not examine parallelization or deployment strategies, we acknowledge their significance for large-scale implementations. Prospective solutions (e.g., Spark MLlib for Collaborative Filtering [57], containerized microservices, real-time streaming, and autoscaling infrastructure) signifies a future trajectory in enhancing our recommendation pipeline for more extensive, resource-demanding environments:

Parallelization Strategies

- Distributed matrix factorization: frameworks like Spark MLlib for Collaborative Filtering enhance the efficiency and memory management of large-scale factorization by distributing the user–item matrix over numerous nodes.
- Memory-based CF and partitioning: distributing users and/or items among computers facilitates partial similarity computations, which are then combined to provide comprehensive suggestions.

Deployment Approaches

- Hybrid batch and online learning: a dual-tier framework (bulk training combined with incremental updates) enables the continuous integration of fresh data without interrupting real-time predictions.
- Autoscaling infrastructure: cloud systems automatically augment computational resources as needed, hence ensuring low-latency suggestions despite a growth in users or requests.

4. Hyperparameter Analysis

In the experiment, we selected the hyperparameter ranges based on past experience; we made them quite wide so that optimizations could be made on them more freely (λ and n), and the starting value of the learning rate l_r. Broader search spaces enhance model capacity by allowing multiple aspects of the objective function to be explored, thus potentially improving locate optimal solutions; however, numerous evaluations and more complex management of exploration-exploitation trade-offs are required by such an expansion [58, 59]. In other words, most hyperparameter ranges are normally determined using known techniques

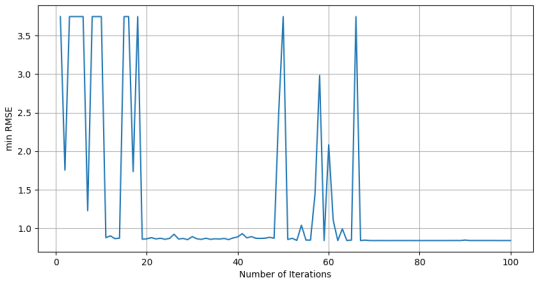

Figure 13. RMSE convergence of the first algorithm on the MovieLens 1M dataset (with hyperparameter space broaden).

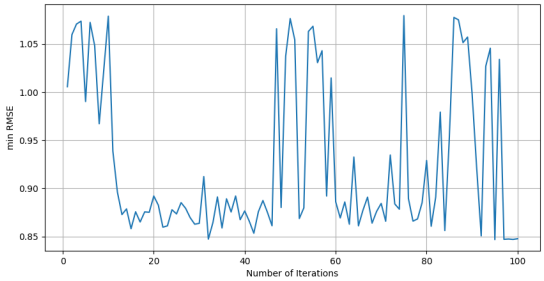

Figure 14. RMSE convergence of the second algorithm on the MovieLens 1M dataset (with hyperparameter space broaden).

or by observing various patterns (empirical data) observed over time. Secondly, but just like other previous research outcomes [11–13], it was found that the hyperparameter tuning process converges within 50–90 iterations. Lastly, all methods we used have only two hyperparameters in our experiment; however, using more than this number increases the cost of Bayesian optimization with Gaussian processes; though this cost does not considerably rise with the number of parameters as reported in [12, 60, 61].

5. Robustness Analyses

In this section, we present robustness analyses to investigate how robust and fine the method is. We delve more into the effect of hyperparameter changes and dataset variables on the performance of both CF methods.

a) Hyperparameter Sensitivity Analysis

In this analysis, we broaden the range of the two hyperparameters (λ and n) used in the Bayesian optimization process for the MovieLens 1M and 10M, Douban and Jester datasets (due to time and physical resource constraints, we could not be able to experiment further with the Netflix Prize dataset because of its size). This will allow us to empirically assess the convergence behavior and robustness of the proposed technique. This facilitates the examination of a broader spectrum of parameter values, possibly uncovering new optimum areas and elucidating the sensitivity and resilience of each method across diverse hyperparameter configurations.

In this new experiment, we set the range of the regularization parameter λ from 1e-6 to 10, and the dimension n ranges from 10 to 1000. The RMSE convergence of the two algorithms for the datasets are presented from Figure 13 to Figure 20. The performance of the two CF algorithms is presented in Table V.

The new figures indicate that a larger optimization area results in increased variability in convergence/performance trends. Certain hyperparameters, when let to fluctuate

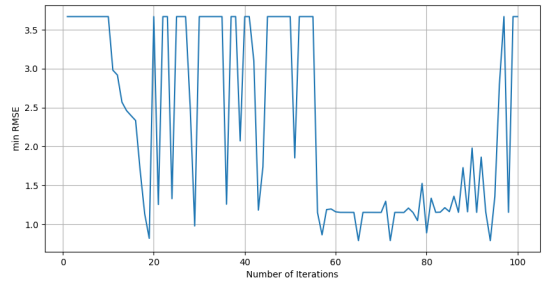

Figure 15. RMSE convergence of the first algorithm on the MovieLens 10M dataset (with hyperparameter space broaden).

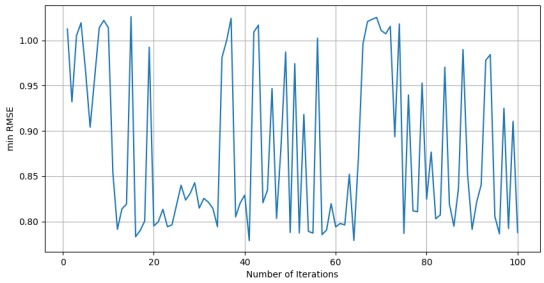

Figure 16. RMSE convergence of the second algorithm on the MovieLens 10M dataset (with hyperparameter space broaden)

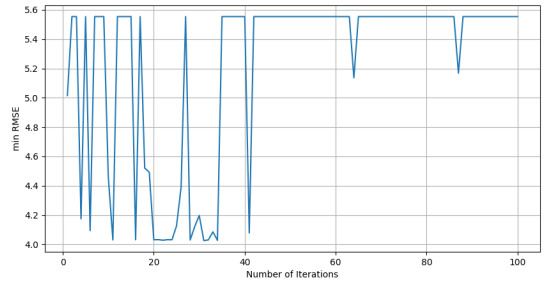

Figure 17. RMSE convergence of the first algorithm on the Jester dataset 2 (with hyperparameter space broaden).

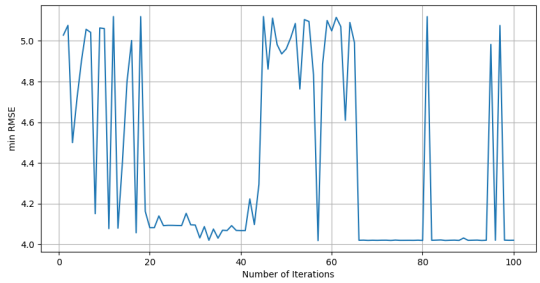

Figure 18. RMSE convergence of the second algorithm on the Jester dataset 2 (with hyperparameter space broaden).

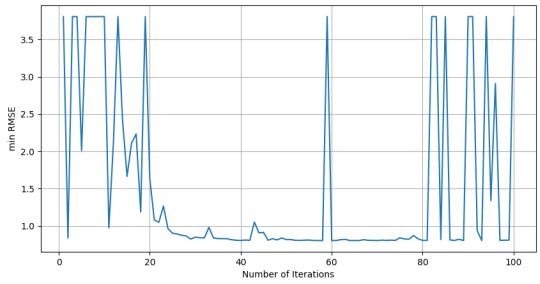

Figure 19. RMSE convergence of the first algorithm on the Douban dataset (with hyperparameter space broaden).

within a wider spectrum, lead to more significant alterations in RMSE, underscoring essential thresholds beyond which performance deteriorates. The previous, tightly defined space (Figures 5 to Figure 12) validated the baseline performance (”baseline“ means that is based on experience) of both methods, but the expanded optimization space (Figure 13 to Figure 20) exposes trade-offs. Specific hyperparameters may provide advantages just within a restricted domain, and broadening the scope might reveal weaknesses – such as areas where the model exhibits instability or overfitting. This contrast highlights the need of carefully balancing exploration and exploitation in hyperparameter optimization. The findings suggest that while fine-tuning within a limited scope might provide satisfactory performance, an expansive search may reveal additional ideal configurations that could further decrease RMSE. Nonetheless, this incurs the expense of heightened sensitivity in other instances, especially for the initial algorithm.

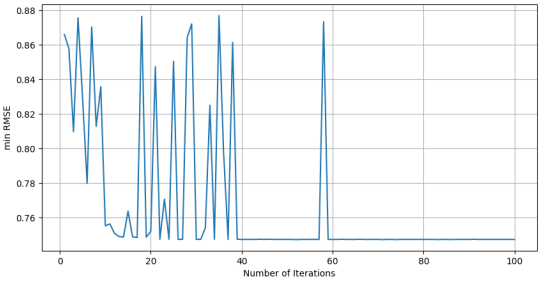

Figure 20. RMSE convergence of the second algorithm on the Douban dataset (with hyperparameter space broaden).

Table V
PERFORMANCE OF THE TWO CF MODELS ON THE EXPERIMENTAL DATASETS WITH THE HYPERPARAMETER SPACE BROADEN.

Algorithm	MovieLens 1M	MovieLens 10M	Jester	Douban
First algo.	0.8409	0.7917	4.034	0.8072
Second algo.	0.8461	0.7791	4.025	0.7498

For the minimum RMSE achieved in the two hyperparameter search spaces, both algorithms yielded generally stable RMSE values; however, these values sometimes plateaued at suboptimal levels owing to the constrained search range.

b) Robustness Across Datasets

This subsection summarizes the results of our tests in the datasets of the experiment.

Netflix Prize dataset: our experiments show that both CF models, when calibrated via Bayesian optimization, quickly converge toward competitive error rates. Specifically, the first algorithm achieved an RMSE of 0.8103, the second algorithm achieved a very similar performance with an RMSE of 0.8107. These results indicate that even with relatively simple CF architectures, automated hyperparameter tuning can yield results that are on par with more complex methods.

MovieLens 1M dataset: the two algorithms got the RMSE values of 0.8410 and 0.8459 accordingly. While a few state-of-the-art techniques typically achieve marginally lower values on this dataset, our methods demonstrate that with decreased manual intervention, competitive performance is attainable.

MovieLens 10M dataset: on the larger MovieLens 10M dataset –characterized by a lower density (0.0131) – our methods again proved effective with the RMSE values of 0.7757 and 0.7787 accordingly. These results emphasize that Bayesian hyperparameter search allows even straightforward CF models to adapt efficiently to larger and more sparse datasets, narrowing the performance gap with more intricate network-based approaches.

Douban dataset: for the Douban dataset, which poses additional challenges due to its user-item interactions and sparsity, our study shows the RMSE values of 0.8055 and 0.7499 accordingly. The significant performance improvement observed with the second algorithm on this dataset highlights how modeling strategies can be tailored to better

mitigate the adverse effects associated with higher data sparsity.

Jester Dataset: finally, in experiments conducted on the Jester dataset 2, we employed scikit-surprise for a consistent evaluation framework. The RMSE results are 4.036 and 4.025. On this denser dataset, both algorithms achieved very similar error rates, supporting the claim that our Bayesian optimization approach is robust across various data distributions.

Overall observations on convergence and robustness: across all datasets in our experiment, convergence curves (refer to Figures 3–12 in Section V.1) reveal rapid initial improvement, with diminishing returns after approximately 50–90 iterations – indicative of effective exploration within the hyperparameter space.

6. Practical Implications

Our work demonstrates that Bayesian optimization with Gaussian processes for tuning hyperparameters of two basic and simple CF algorithms on three popular datasets yields very competitive performance. We think that different issue sets dominated by each approach account for this enhancement. Although there are situations where minor improvements may occur if you switch algorithms in one approach, it is more efficient to use a combination of several approaches aimed at solving specific sets of issues than making changes within one approach; this way many different approaches could be developed to solve various issues that influence CF models. In terms of return on investment, reliable algorithms are preferable to sophisticate yet less reliable ones, according to [62–64]. Practitioners should choose more advanced algorithms over simplistic CF ones if they want better results; however, they must make sure that these other properties are balanced such as complexity, usability, and stability so that it will not become less efficient after all.

VI. Conclusion

This research analyzed the efficacy of two fundamental CF methods augmented by Bayesian optimization using GPs for automated hyperparameter tuning. Our experimental investigation of datasets including Netflix Prize, MovieLens (1M and 10M), Jester, and Douban has shown that even basic CF models may attain competitive performance with meticulous hyperparameter tuning, therefore reducing the need on human adjustments.

This study significantly contributes by explicitly demonstrating how Bayesian optimization systematically investigates the hyperparameter space, resulting in enhanced performance efficiency. As detailed in Section IV, our approach utilizes the iterative characteristics of GPs to achieve optimum configurations, thereby alleviating problems like data sparsity and model underfitting. The findings – exemplified by an RMSE of 0.8103 on the Netflix Prize Test set – highlight the efficacy of this method in improving model precision while conserving time and processing resources.

Moreover, the work has identified other obstacles that persist for future investigation. Our experimental setup tackled the cold-start issue using a global-averages strategy for new users and items; nonetheless, this approach is acknowledged as a baseline that reduces the intricacies of actual user behavior. As outlined in Section IV.5, advanced strategies – possibly using hybrid or meta-learning methodologies – may provide a more tailored initialization that reconciles efficiency with precision.

Likewise, the challenges posed by data sparsity (elaborated in Section IV.6) underscore that even cutting-edge optimization methods may not entirely surmount the inherent constraints of sparse interaction matrices. Future research should explore the implementation of scalable GP variations and sophisticated partitioning or distributed matrix factorization techniques (e.g., using frameworks such as Spark MLlib) to mitigate these limitations in bigger and more dynamic settings.

In conclusion, the article demonstrates that automated hyperparameter tweaking using Bayesian optimization utilizing GP models may substantially improve the efficacy of CF algorithms. The effectiveness of this method is apparent across several datasets; nonetheless, enhancements are required, especially concerning the cold-start issue and data sparsity. An explicit, cohesive narrative from the literature review to the experimental findings underscores the promise of integrating conventional CF approaches with contemporary optimization tools, paving the way for future research initiatives.

This paper fully links our methodology, experimental results, and theoretical insights, offering a comprehensive assessment of existing methodologies and a framework for future progress in recommender systems. The iterative and modular characteristics of our methodology allow ongoing enhancements, guaranteeing that as computer resources and algorithmic complexity advance, the robustness and relevance of CF models will similarly expand in more intricate settings.

REFERENCES

[1] K. Srikumar, "A framework of agent-based personalized recommender system for e-commerce," *South Asian journal of management*, vol. 11, p. 66, 2004.

[2] Y. H. Cho, J. K. Kim, and S. H. Kim, "A personalized recommender system based on web usage mining and decision tree induction," *Expert Systems with Applications*, vol. 23, no. 3, pp. 329–342, 2002.

[3] G. Adomavicius and A. Tuzhilin, "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*, vol. 17, no. 6, pp. 734–749, 2005.

[4] S. Yan, "A collaborative filtering recommender approach by investigating interactions of interest and trust," in *Knowledge Engineering and Management*, Berlin, Heidelberg, 2014, pp. 173–188.

[5] C. P. Lam, "Collaborative filtering using associative neural memory," in *Intelligent Techniques for Web Personalization*, Berlin, Heidelberg, 2005, pp. 153–168.

[6] R. Zhang, Q.-d. Liu, Chun-Gui, J.-X. Wei, and Huiyi-Ma, "Collaborative filtering for recommender systems," in *2014 Second International Conference on Advanced Cloud and Big Data*, 2014, pp. 301–308.

[7] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[8] S. Rendle, L. Zhang, and Y. Koren, "On the difficulty of evaluating baselines: A study on recommender systems," *ArXiv*, vol. abs/1905.01395, 2019.

[9] Y. Koren, "The bellkor solution to the netflix grand prize," 2009.

[10] ——, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08, New York, NY, USA, 2008, p. 426–434.

[11] M. Imani, M. Imani, and S. F. Ghoreishi, "Bayesian optimization for expensive smooth-varying functions," *IEEE Intelligent Systems*, vol. 37, no. 4, pp. 44–55, 2022.

[12] Y. Morita, S. Rezaeiravesh, N. Tabatabaei, R. Vinuesa, K. Fukagata, and P. Schlatter, "Applying bayesian optimization with gaussian process regression to computational fluid dynamics problems," *Journal of Computational Physics*, vol. 449, p. 110788, Jan. 2022.

[13] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'12, Red Hook, NY, USA, 2012, p. 2951–2959.

[14] S. Molaei, A. Havvaei, H. Zare, and M. Jalili, "Collaborative deep forest learning for recommender systems," *IEEE Access*, vol. 9, pp. 22 053–22 061, 2021.

[15] M. F. Aljunid and M. Doddaghatta Huchaiah, "Multi-model deep learning approach for collaborative filtering recommendation system," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 4, p. 268–275, nov 2020.

[16] Y. Huang and J. T. Kwok, "Collaborative filtering via co-factorization of individuals and groups," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[17] Z. Li, J. Huang, and N. Zhong, "Exploiting user and item embedding in latent factor models for recommendations," in *Proceedings of the International Conference on Web Intelligence*, ser. WI '17, New York, NY, USA, 2017, p. 1241–1245.

[18] F. Horasan, A. H. Yurttakal, and S. Gündüz, "A novel model based collaborative filtering recommender system via truncated ulv decomposition," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 8, p. 101724, 2023.

[19] Y. Chen, S. Mensah, F. Ma, H. Wang, and Z. Jiang, "Collaborative filtering grounded on knowledge graphs," *Pattern Recognition Letters*, vol. 151, pp. 55–61, 2021.

[20] N. Khaledian and F. Mardukhi, "Cfmt: a collaborative filtering approach based on the nonnegative matrix factorization technique and trust relationships," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 5, pp. 2667–2683, May 2022.

[21] H. Xia, Y. Luo, and Y. Liu, "Attention neural collaboration filtering based on gru for recommender systems," *Complex & Intelligent Systems*, vol. 7, no. 3, pp. 1367–1379, June 2021.

[22] S. Rendle, "Factorization machines with libfm," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, May 2012.

[23] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08, New York, NY, USA, 2008, p. 880–887.

[24] P. Szabó and B. Genge, "Hybrid hyper-parameter optimization for collaborative filtering," in *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2020, pp. 210–217.

[25] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag, "Collaborative hyperparameter tuning," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13, 2013, p. II–199–II–207.

[26] M. N. Volkovs and R. S. Zemel, "Collaborative ranking with 17 parameters," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'12, Red Hook, NY, USA, 2012, p. 2294–2302.

[27] H. H. Hoos, *Automated Algorithm Configuration and Parameter Tuning*, Berlin, Heidelberg, 2012, pp. 37–71.

[28] P. N. Koch, O. Golovidov, S. Gardner, B. Wujek, J. D. Griffin, and Y. Xu, "Autotune: A derivative-free optimization framework for hyperparameter tuning," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[29] Q. Lu, K. D. Polyzos, B. Li, and G. B. Giannakis, "Surrogate modeling for bayesian optimization beyond a single gaussian process," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 11 283–11 296, 2023.

[30] P. Lualdi, R. Sturm, A. Camero, and T. Siefkes, "An uncertainty-based objective function for hyperparameter optimization in gaussian processes applied to expensive black-box problems," *Applied Soft Computing*, vol. 154, p. 111325, 2024.

[31] M. A. L. Pearce, M. Poloczek, and J. Branke, "Bayesian optimization allowing for common random numbers," *Oper. Res.*, vol. 70, no. 6, p. 3457–3472, nov 2022.

[32] P. I. Frazier, "A tutorial on bayesian optimization," 2018.

[33] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[34] H. Wang, B. van Stein, M. Emmerich, and T. Back, "A new acquisition function for bayesian optimization based on the moment-generating function," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 507–512.

[35] M. Pearce, "Acquisition functions for simultaneous bayesian

optimisation of multiple problems," in *2017 Winter Simulation Conference (WSC)*, 2017, pp. 4618–4619.
[36] J. Berk, V. Nguyen, S. Gupta, S. Rana, and S. Venkatesh, "Exploration enhanced expected improvement for bayesian optimization," in *Machine Learning and Knowledge Discovery in Databases*, Cham, 2019, pp. 621–637.
[37] M. T. M. Emmerich, A. H. Deutz, and J. W. Klinkenberg, "Hypervolume-based expected improvement: Monotonicity properties and exact computation," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 2147–2154.
[38] D. Kim and B. Suh, "Enhancing vaes for collaborative filtering: flexible priors & gating mechanisms," in *Proceedings of the 13th ACM Conference on Recommender Systems*, ser. RecSys '19, Sep. 2019.
[39] H. Steck, "Embarrassingly shallow autoencoders for sparse data," in *The World Wide Web Conference*, ser. WWW '19, May 2019.
[40] I. Shenbin, A. Alekseev, E. Tutubalina, V. Malykh, and S. I. Nikolenko, "Recvae: A new variational autoencoder for top-n recommendations with implicit feedback," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, ser. WSDM '20, New York, NY, USA, 2020, p. 528–536.
[41] S. C. Han, T. Lim, S. Long, B. Burgstaller, and J. Poon, "Glocal-k: Global and local kernels for recommender systems," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM '21, New York, NY, USA, 2021, p. 3063–3067.
[42] Z. Chen and B. Wang, "How priors of initial hyperparameters affect gaussian process regression models," *Neurocomputing*, vol. 275, pp. 1702–1710, 2018.
[43] Y. Liu, S. Wang, X. Li, and F. Sun, "A meta-adversarial framework for cross-domain cold-start recommendation," *Data Science and Engineering*, vol. 9, no. 2, pp. 238–249, June 2024.
[44] H. Xu, C. Li, Y. Zhang, L. Duan, I. W. Tsang, and J. Shao, "Metacar: Cross-domain meta-augmentation for content-aware recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 8199–8212, 2023.
[45] M. Li, W. Que, Z. Geng, M. Li, Z. Kou, J. Chen, C. Guo, and B. Zhang, "Cold-start item recommendation for representation learning based on heterogeneous information networks with fusion side information," *Future Generation Computer Systems*, vol. 149, pp. 227–239, 2023.
[46] P. Magron and C. Févotte, "Neural content-aware collaborative filtering for cold-start music recommendation," *Data Mining and Knowledge Discovery*, vol. 36, no. 5, pp. 1971–2005, September 2022.
[47] N. Heidari, P. Moradi, and A. Koochari, "An attention-based deep learning method for solving the cold-start and sparsity issues of recommender systems," *Knowledge-Based Systems*, vol. 256, p. 109835, 2022.
[48] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," in *Interspeech*, 2017.
[49] K. Song, W. Gao, L. Zhao, J. Lin, C. Sun, and X. Liu, "Cold-start aware deep memory network for multi-entity aspect-based sentiment analysis," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 7 2019, pp. 5197–5203.
[50] B. Hu, Z. Li, and W. Chao, "Data sparsity: A key disadvantage of user-based collaborative filtering?" in *Web Technologies and Applications*, Berlin, Heidelberg, 2012, pp. 602–609.
[51] M. Grčar, D. Mladenič, B. Fortuna, and M. Grobelnik, "Data sparsity issues in the collaborative filtering framework," in *Advances in Web Mining and Web Usage Analysis*, Berlin, Heidelberg, 2006, pp. 58–76.
[52] F. Yin, Z. Wang, W. Tan, and W. Xiao, "Sparsity-tolerated algorithm with missing value recovering in user-based collaborative filtering recommendation," *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE*, vol. 10, no. 15, pp. 4939–4948, 2013.
[53] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbaty, "scikit-optimize/scikit-optimize (v0.9.0)," 2021, zenodo. https://doi.org/10.5281/zenodo.5565057.
[54] N. Hug, "Surprise: A python library for recommender systems," *Journal of Open Source Software*, vol. 5, no. 52, p. 2174, 2020.
[55] R. Garnett, *gaussian processes*, 2023, p. 15–44.
[56] S. Rana, C. Li, S. Gupta, V. Nguyen, and S. Venkatesh, "High dimensional bayesian optimization with elastic gaussian process," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, 2017, p. 2883–2891.
[57] "Collaborative filtering," https://spark.apache.org/docs/latest/ml-collaborative-filtering.html, accessed: 2025-01-28.
[58] H. Ha, S. Rana, S. Gupta, T. Nguyen, H. Tran-The, and S. Venkatesh, *Bayesian optimization with unknown search space*, Red Hook, NY, USA, 2019.
[59] V. Nguyen, S. Gupta, S. Rane, C. Li, and S. Venkatesh, "Bayesian optimization in weakly specified search space," in *2017 IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 347–356.
[60] M. Blum and M. A. Riedmiller, "Optimization of gaussian process hyperparameters using rprop," in *The European Symposium on Artificial Neural Networks*, 2013.
[61] R. Preuss and U. von Toussaint, "Parallel computation of Gaussian processes," *AIP Conference Proceedings*, vol. 1853, no. 1, p. 050004, 06 2017.
[62] N. T. Blog, "Netflix recommendations: Beyond the 5 stars," Apr 2012, accessed: 2024-Sep-15.
[63] B. Hallinan and T. Striphas, "Recommended for you: The netflix prize and the production of algorithmic culture," *New Media & Society*, vol. 18, no. 1, pp. 117–137, 2016.
[64] E. Campochiaro, R. Casatta, P. Cremonesi, and R. Turrin, "Do metrics make recommender algorithms?" in *2009 International Conference on Advanced Information Networking and Applications Workshops*, 2009, pp. 648–653.

Tuan-Anh Nguyen received his PhD in computer science from the University of Kent, UK, in 2010. His main research interest is in applying science to broader social activities. He currently works at Ho Chi Minh City University of Foreign Languages - Information Technology.
Email: anhnt1@huflit.edu.vn