Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

Interpretable User Behavioral Analysis and Personalized Recommendation with Side Information

Chen Feng

School of Industrial Engineering
Purdue University
Grissom Hall
West Lafayette, IN 47907, USA

Ruijiu Mao

Institute of Operations and Analytics
National University of Singapore
Innovation 4.0, 3 Research Link, #04-01
117602, SINGAPORE

ABSTRACT

Due to information overload, recommender system is developed and widely applied to better predict customer behaviors empowered by interaction data. In this paper, we develop an interpretable recommender system to improve click behavior prediction accuracy. Our proposed recommender system consists of i) designing a user-item interaction rating system defined as the weighted-sum of the multiple interaction variables, where the weight vector is regarded as hyperparameter, ii) using matrix factorization technique to capture the latent structure from the observable user-item interaction, and predict personalized ratings for items not shown to users, and iii) determining the weight vector by optimizing AUC which evaluates click behavior prediction accuracy on a validation set. To be specific, an algorithm with radial basis function surrogates is applied for hyperparameter optimization. We also develop a case study with user behavioral data from Netease music and observe a 15.5% improvement in AUC when incorporating multiple information sources.

1 INTRODUCTION

The Internet is flooding with content, and people could easily get overwhelmed and drown in content. More than 500 hours of video are uploaded to YouTube every minute (New York Times 2021a). Meanwhile, users' interest varies and changes from time to time. Practically every day or week, as quoted from (New York Times 2021b), "there is a contemporary piece of digital leisure or an internet superstar mania that comes and goes a lot sooner than quick vogue". Empowered by big data, recommender systems help anticipate the preference that a user would show to an item, and it performs well in industry. Ten years ago from today, already three-quarters of videos that people watch on Netflix are directed by their recommender system (Washington Times 2012).

A recommender system seeks to predict individual preference for an item that has not yet been discovered by the specific user. Such a problem can be illustrated as a matrix completion problem, where we regard the entry (u,i) of the matrix as the rating of item i by user u. The entry is observable if user u has been recommended on item i and is missing otherwise. Conventional wisdom indicates that users sharing similar opinions about some items, topics, or standards may react similarly towards other things. To fill in the missing entries of a partially observed matrix that has low rank, researchers have developed theories and algorithms. Among them, matrix factorization is one of the most popular and widely applied algorithms. In our work, we solve the L-2 matrix factorization regularization problem using a gradient descent method. (Koren 2008).

Since our objective is to predict personalized click behavior, it is natural to recover a matrix of click-through probability, in which observable entries equal zero when the user swipes away and equal one when the user clicks on. However, unlike the dataset including only one-dimensional rating, for example, the most popular dataset for the Netflix competition only contains movie rating (Bennett, Lanning, et al. 2007), more datasets provide a variety of user activities nowadays. For instance, in addition to click

behavior records, (Zhang, Hu, Liu, Wu, and Li 2022) also include user activities such as likes, shares, and comments subsequent to clicks. To make predictions with higher accuracy, we are encouraged to integrate a variety of side information. If a user on a video-sharing platform not only clicked on content, but also liked or commented, the preference rating should be higher. Therefore, we design a user-item interaction rating system defined as the weighted-sum of the multiple interaction variables, where the weight vector should be tuned carefully. Note that even though the matrix we are recovering is a rating matrix instead of click-through probability, we evaluate the preference rating matrix by click behavior on validation data. In other words, we hope to incorporate more interactive information to improve the accuracy of the click behavior prediction. We use AUC as our evaluation metric.

Hyperparameter optimization for the weight vector is very challenging due to the high complexity of the mapping from the weight vector domain to the validation error of the click behavior prediction. To avoid the curse of dimensionality, we apply a deterministic-surrogate-based hyperparameter optimization method (Ilievski, Akhtar, Feng, and Shoemaker 2017) which outperforms Gaussian process-based algorithms and tree-based algorithms. The hyperparameter tuning method shores up the efficiency of our proposed recommender system.

In this paper, we develop an interpretable and efficient recommender system to predict click behavior by incorporating multiple interaction data types. Our contribution is threefold.

- We develop an interpretable recommender system from user-item interactive data. Unlike sophisticated black-box models, our model allows everyone to discern the rating mechanics contributed from various types of interaction.
- We develop an efficient recommender system supported by a hyperparameter tuning method. Hyperparameter tuning on the weight vector avoids making arbitrary assumptions to forcibly incorporate information with variety.
- We conduct a case study with Netease music user-item interaction data (Zhang, Hu, Liu, Wu, and Li 2022), and observe a 15.5% improvement in AUC when incorporating multiple information. From the case study, we also shed managerial insights that video creators should value more on the quality of content than the effect of eye-catching, based on the observation that commenting, viewing comment, and liking contribute the most to preference rating.

The rest of this paper is structured as follows. In Section 2, we provide a literature review on related work. In Section 3, we describe the problem and present our methodology including the matrix factorization algorithm and hyperparameter tuning algorithm. In Section 4, we conduct a case study and illustrate our findings in comparison with a benchmark model. In Section 5, we draw conclusions and outline future work.

2 RELATED LITERATURE

Our work contributes to two streams of literature relating to recommender system empowered by multiple sources of user-item interaction data and hyperparameter optimization application.

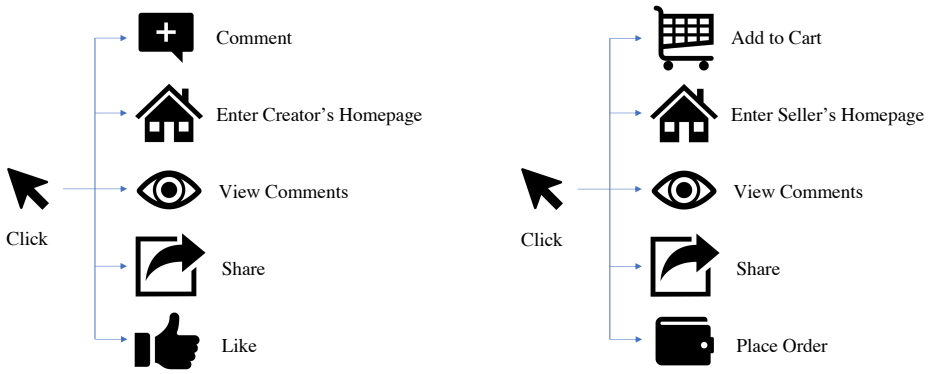
Researchers have developed theories and algorithms regarding low-rank matrix completion. Following the seminal work (Koren 2008), our matrix completion algorithm solves the L-2 matrix factorization regularization problem by stochastic gradient descent optimization. The entries of the rating matrix are sometimes directly provided like the movie-rating score data (Bennett, Lanning, et al. 2007), or set as binary values as implicit feedback (e.g., a purchase record) (Wang, Guo, and Du 2018), or defined by a given formulation incorporating all resources of data provided (Bugliarello, Jain, and Rakesh 2019). Our design of the rating matrix involves a carefully studied weight vector corresponding to the level of contribution from multiple types of activities with a sequential structure. To be specific, all other interactions are followed by one type of interaction. For example, clicking on the product web-page is precedent to adding to cart or placing an order in e-commerce. Clicking on the short video is precedent to giving it a "like" on

a video-sharing platform. Therefore, to predict the user-item propensities, collective matrix factorization (Singh and Gordon 2008), which studies matrix recovery using multiple matrices across multiple groups, or modeling into a tensor recovery problem (Farias and Li 2019) could not resolve data sparsity issue and can be simplified to calibrating weight vector as defined in our problem setting. Different from (Quan, Wang, and Guan 2021), our approach focuses on interactions between users and items without including user and item features, such as user demographics or product specifications. In fact, we are interested in interpreting individual preferences from different types of interaction information, and we aim to show the improvement by introducing multiple user-item interaction data sources.

The history of hyperparameter optimization problem can be traced back to 1990s (Feurer and Hutter 2019). Hyperparameter optimization for the weight vector is very challenging due to the high complexity of the mapping from the weight vector domain to the validation error of the click behavior prediction. To ensure time-efficiency, we apply a deterministic-surrogate-based hyperparameter optimization method (Ilievski, Akhtar, Feng, and Shoemaker 2017) for hyperparameter tuning. Surrogate-based optimization (Mockus, Tiesis, and Zilinskas 1978) is a global optimization strategy utilizing a surrogate model of the target function to determine the next promising point for validation. Choices of surrogates include probabilistic surrogates (Snoek, Larochelle, and Adams 2012; Hernández-Lobato, Hoffman, and Ghahramani 2014) and radial basis function (RBF) surrogates (Regis 2014; Regis and Shoemaker 2013). Probabilistic surrogates require accurate estimation of sufficient statistics of the error distribution and are thus time-consuming. The mixed-integer algorithm we apply with radial basis function (RBF) surrogates, on the other hand, searches the surrogate through dynamic coordinate search and takes less evaluation time. The hyperparameter tuning method shores up the time-efficiency of our proposed recommender system.

3 METHODOLOGY

In this section, we start with the definition of our recommendation problem. Let U be the user set, M be the item set, and $|U| = m, |M| = n$. Let K be the set of tuple (u,i) where the item $i \in M$ was recommended to the user $u \in U$ in the past, and the interactions between tuples in K are recorded. Our recommendation problem is to predict the click behavior of user $u \in U$ to the item $i \in M$ where $(u,i) \notin K$. The data consists of information of h categories of interaction behaviors including clicking, together with its subsequent behaviors defined as $isBehavior(2), \cdots, isBehavior(h)$. The sequential structure indicates that all other interactions are followed by one type of interaction. As shown in Figure 1a, clicking on the short video is precedent to giving it a "like" (or comment, etc.) on a video-sharing platform, and clicking on the product web-page is precedent to adding to cart (or placing an order, etc.) in e-commerce as depicted in Figure 1b.



(a) Video-sharing Platform (b) E-commerce

Figure 1: Sequential Structure of Interactions

3.1 Mapping from Interaction Behaviors to Preference Ratings

Typically, recommender systems make prediction of click behavior based on the partially observed user-item interaction matrix where the corresponding entry value is 1 if the user clicked the item and 0 otherwise. However, binary information cannot reflect the preference or clicking tendency of a user on an item very well. In addition to clicking information, there are other interaction behaviors that also reflect preference and clicking tendency. Therefore, we believe that it is more effective for a recommender system to learn and predict click behavior based on preference rather than binary click information only. Basically, we assume that there is an underlying function $f : \{0,1\}^h \rightarrow \mathbf{R}$ that maps interaction behaviors to a rating that measures user preference on items. As a starting point, we assume a linear function for f:

$$r_{u,i} = y_1 \cdot isClick_{u,i} + y_2 \cdot isBehavior(2)_{u,i} + \cdots + y_h \cdot isBehavior(h)_{u,i}, \tag{1}$$

where $\mathrm{y} = (y_1, \cdots y_h)$ is the weight vector for the set of behavior types that contribute to the preference rating. Here, we only consider interaction behaviors that reflect a positive attitude toward the item, so we assume $y_i \in [0,1]$ without loss of generality. Define $A \subset K$ as the training data set and $B = K - A$ as the validation data set. Given a choice of y, we can obtain a corresponding $m \times n$ partially observed interaction matrix $Z(\mathrm{y})$ for interaction information in A, where

$$Z(\mathrm{y})_{u,i} = \begin{cases} r_{u,i}(\mathrm{y}), & \text{if } (u,i) \in A, \\ N/A, & \text{Otherwise.} \end{cases} \tag{2}$$

3.2 Matrix Factorization Model

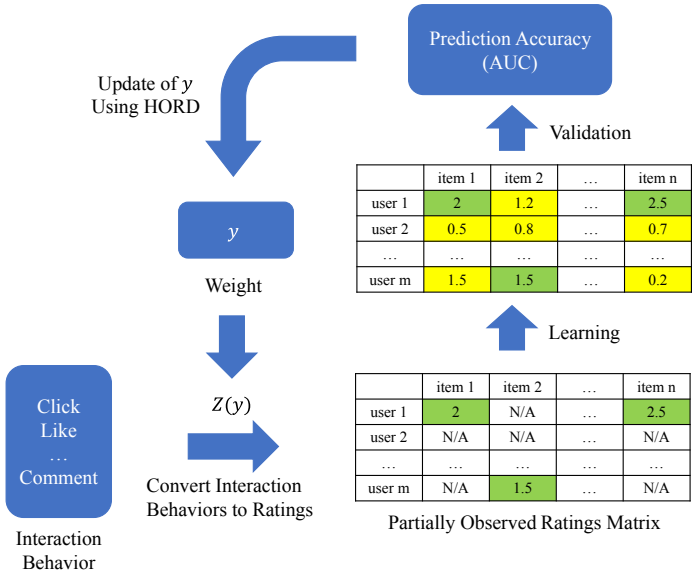


Figure 2: The framework of recommender system using HORD.

Given the partially observed interaction matrix $Z(\mathrm{y})$, we apply the classic Latent Factor Model (LFM) (Koren, Bell, and Volinsky 2009) to estimate the preference rating of each entry in the validation data set, which is denoted by $\hat{r}_{u,i}$ where $(u,i) \in B$. LFM is a model-based approach (Koren 2008) that generates the prediction rating matrix by multiplication of two smaller matrices:

$$\hat{Z} = QP, \tag{3}$$

where

$$\hat{Z}_{u,i} := \hat{r}_{u,i},\ Q \in \mathbb{R}^{m\times k},\ P \in \mathbb{R}^{k\times n}, \tag{4}$$

and k is the number of latent factors. The intuition behind this step is to use two matrices to measure the preference of each user for each latent factor and the relationship between each latent factor and each item. Each user u will be represented by a vector p_u in k-dimension, and each item i will be represented by a vector q_i in k-dimension. As a result, the dot product of p_u and q_i will be the prediction of the preference rating of user u for the item i:

$$\hat{Z}_{u,i} = \mathrm{q}_i^T \mathrm{p}_u. \tag{5}$$

The regularized squared error on the training dataset is minimized to learn p_u and q_i:

$$\min_{\mathrm{p,q,b}} \sum_{(u,i)\in A} (Z_{u,i} - \mu - b_u - b_i - \langle \mathrm{q}_i, \mathrm{p}_u \rangle)^2 + \beta(\|\mathrm{q}_i\|^2 + \|\mathrm{p}_u\|^2 + b_u^2 + b_i^2), \tag{6}$$

where $Z_{u,i}$ is the known rating of item i given by user u, μ is the average of all the ratings, b_u and b_i are the bias of user u and item i from μ respectively. For example, if user u is critical, he or she may have lower average ratings than other users and b_u will be negative; if a card is of high quality, it may receive higher average ratings than other items and b_i will be positive. b_u and b_i, together with q and q are variables of this optimization problem and will be learned from the known ratings.

A gradient descent method is used to minimize (6) as in (Koren, Bell, and Volinsky 2009). The system updates the variables through the following steps:

$$\begin{aligned}
\mathrm{b}_u &\leftarrow \mathrm{b}_u + \gamma(e_{ui} - \beta \mathrm{b}_u),\\
\mathrm{b}_i &\leftarrow \mathrm{b}_i + \gamma(e_{ui} - \beta \mathrm{b}_i),\\
\mathrm{p}_u &\leftarrow \mathrm{p}_u + \gamma(e_{ui} \cdot \mathrm{q}_i - \beta \mathrm{p}_u),\\
\mathrm{q}_i &\leftarrow \mathrm{q}_i + \gamma(e_{ui} \cdot \mathrm{p}_u - \beta \mathrm{q}_i),
\end{aligned} \tag{7}$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$, $\hat{r}_{ui} = \mu + b_u + b_i + \langle q_i, p_u \rangle$, and γ is the step size.

Let $\mathrm{q}^*, \mathrm{p}^*, \mathrm{b}^*$ denote the optimal solution of (6). The predicted rating $\hat{Z}_{u,i}$ has the following formula:

$$\hat{Z}_{u,i} = \mu + b_u^* + \mathrm{b}_i^* + \langle \mathrm{q}_i^*, \mathrm{p}_u^* \rangle. \tag{8}$$

3.3 Evaluation of the Model and Weight Optimization using HORD

In this section, we illustrate our approach to optimize the choice of behavior weights. Optimization of behavior weights is a global optimization of a black-box function F^{LFM} that maps a weight vector y to the validation accuracy of the model. In this sense, behavior weights play a similar role as hyperparameters of the recommender system. Therefore, we take advantage of Hyperparameter Optimization using RBF based surrogate and DYCORS (HORD), which is developed for hyperparameter optimization. Proposed in (Ilievski, Akhtar, Feng, and Shoemaker 2017), HORD is a type of surrogate-based optimization (Mockus 1994) and an efficient method for the global optimization of expensive black-box functions. In each iteration of HORD, radial basis function (RBF) is used as a surrogate to be the approximation of the black-box function at first, and then Dynamic coordinate search (DYCORS) (Regis and Shoemaker 2013) is used to find a near-optimal hyperparameter. The pseudocode of the application of HORD to behavior weight tuning is given in Algorithm 2, and the interpretation and choice of parameters for HORD are referred to (Ilievski, Akhtar, Feng, and Shoemaker 2017). We use the area under the ROC curve (AUC) to evaluate the accuracy of the prediction on the validation dataset, which is the output of the black box function F^{LFM}. The detailed definition of F^{LFM} is given in Algorithm 1.

Algorithm 1: Recommender System Evaluation Function

Function : $F^{\mathrm{LFM}}(\mathrm{y})$:
Input: weight (hyperparameter) $\mathrm{y} = (y_1, ..., y_h)$;
for ***each*** $(u,i) \in A$ **do**
  Obtain $r_{u,i}$ from Equation (1);
end
Construct partially observed matrix $Z(\mathrm{y})$ as Equation (2);
Solve $\mathrm{q}^*, \mathrm{p}^*, \mathrm{b}^*$ for (6) using Equation (7) and obtain prediction $\hat{Z}_{u,i}$ for $(u,i) \in B$ from Equation (8);
Calculate the area under the ROC curve (AUC) for the prediction of click behavior of $(u,i) \in B$;
Return AUC

Algorithm 2: Recommender system behavior weight tuning by HORD

Data: Number of interaction types h, $n_0 = 2(D+1)$, $l = 100h$, maximal evaluation number $N_{\max}$.
Result: $\mathrm{y}_{\mathrm{best}}$: optimal weight for each interaction type capturing the click tendency.
Use Latin hypercube sampling to sample n_0 points and set $\mathscr{Y} = \{y_i\}_{i=1}^{n_0}$;
Compute $F^{\mathrm{LFM}}(\mathrm{y})$, initialize $iter = n_0$, and we denote $\mathscr{A}_{iter} = \{(\mathrm{y}, F^{\mathrm{LFM}}(\mathrm{y}))\}_{i=1}^{n_0}$;
while $iter < N_{\max}$ **do**
  Use $\mathscr{A}_{iter}$ to fit or update the RBF surrogate model $S_{iter}(\mathrm{y})$ as $S_{iter}(\mathrm{y}) = \sum_{i=1}^{iter} \lambda_i \phi(\|\mathrm{y} - \mathrm{y}_i\|) + p(\mathrm{y})$;
  Set $\mathrm{y}_{\mathrm{best}} = \arg\min\{F^{\mathrm{LFM}}(\mathrm{y}) : i = 1, \ldots, iter\}$;
  Compute φ_{iter} by $\varphi_{iter} = \varphi_0 \left[1 - \frac{\ln(iter - n_0 + 1)}{\ln(N_{\max} - n_0)}\right]$, $n_0 <= iter < N_{\max}$, i.e, the probability of perturbing a coordinate;
  Populate Ω_{iter} with l candidate points, $t_{iter,1:l}$, where for each candidate $\mathrm{y}_j \in t_{iter,1:l}$, (a) Set $\mathrm{y}_j = \mathrm{y}_{\mathrm{best}}$ (b) Select the coordinates of y_j to be perturbed with probability φ_{iter} and (c) Add δ_i sampled from $\mathscr{N}(0, \sigma_{iter}^2)$ to the coordinates of y_j selected in (b) and round to nearest integer if required;
  Calculate $V_{iter}^{ev}(t_{iter,1:l})$ by

$$V^{ev}(t) = \begin{cases} \frac{S(t) - s^{\min}}{s^{\max} - s^{\min}}, & \text{if } s^{\max} \neq s^{\min} \\ 1, & \text{otherwise} \end{cases},$$

  $V_{iter}^{dl}(t_{iter,1:l})$ by

$$V^{dl}(t) = \begin{cases} \frac{\Delta^{\max} - \Delta(t)}{\Delta^{\max} - \Delta^{\min}}, & \text{if } \Delta^{\max} \neq \Delta^{\min} \\ 1, & \text{otherwise} \end{cases},$$

  and the final weighted score $W_{iter}(t_{iter,1:l})$ by $W(t) = wV^{ev}(t) + (1-w)V^{dl}(t)$;
  Set $\mathrm{y}^* = \arg\min\{W_{iter}(t_{iter,1:l})\}$;
  Evaluate $F^{\mathrm{LFM}}(\mathrm{y}^*)$;
  Adjust the variance σ_{iter}^2 by $\sigma_{iter+1}^2 = \min(\sigma_{iter}^2/2, 0.005)$;
  Update $\mathscr{A}_{iter+1} = \{\mathscr{A}_{iter} \cup (y^*, F^{\mathrm{LFM}}(\mathrm{y}^*))\}$;
  $iter = iter + 1$.
end
Return $\mathrm{y}_{\mathrm{best}}$.

4 CASE STUDY

4.1 Data Description

In this section, we conduct a case study using the Netease music user-item interaction dataset. Since our approach focuses on interactions between users and items, user and item features such as user demographics or item specifications are not included. The item refers to the music card in Netease music, which is a mini blog with music and video. The interaction between the user and item includes clicking, commenting, entering the creator's homepage, viewing comments, sharing, and liking. The original dataset contains many users and cards with few interaction data, so we focus on the interaction data between the 494 most active users, and 968 most popular music cards. The density of the observed data and statistics are given in Table 1 and Figure 3. The interaction matrix is still very sparse, with an observation density of 2.4% and overall click-through rate of 4.9%.

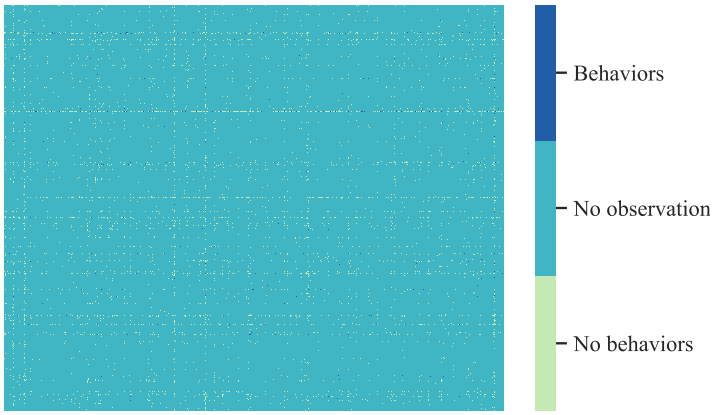


Figure 3: Density of Observed Data (x: card, y: user)

Table 1: Netease Dataset Statistics

Behavior	**Count**
Click	581
Comment	2
Enter creator's HPAGE	13
View Comments	182
Share	8
Like	16
Swipe Away	11,039
Total Pair	478,192

4.2 Experimental Design

To highlight the importance of introducing side information, we introduce a benchmark algorithm, the naive LFM, which recovers a matrix where the observable entries equal zero when the user swipes away and equal one when the user clicks on.

In our proposed framework, the weight vector is tuned according to Algorithm 1. However, the LFM hyperparameters β and γ, the number of latent variables, and the stopping criterion are yet to be decided. Recall that we aim to show the improvement of introducing multiple user-item interaction data sources. Therefore, we set the same β and γ which are equal to 0.02 and 0.05 for our framework and the benchmark algorithm, the naive LFM. We compare the performance of our framework against the naive LFM for $k = 20, 30, \cdots, 70$ respectively, where k is the number of latent factors. A summary of our hyperparameter settings for matrix factorization is presented in Table 2.

Table 2: Hyperparameter Settings for Matrix Factorization

Hyperparameter	**Setting**
k	20/30/40/50/60/70
β	0.02
γ	0.05
Stopping criterion	800 iterations

We stop the matrix factorization in our framework when 800 iterations of gradient descent are completed. We use the number of iterations instead of the squared error as the stopping criterion because the regularized squared error is in proportion to the scale of our weight vector. For example, weight vector $[0.1, 0.1, 0.1, 0.1, 0.1, 0.1]$ and $[1, 1, 1, 1, 1, 1]$ lead to the same prediction result. However, the regularized squared error from 1 generated by the former is 1/10 of the latter one.

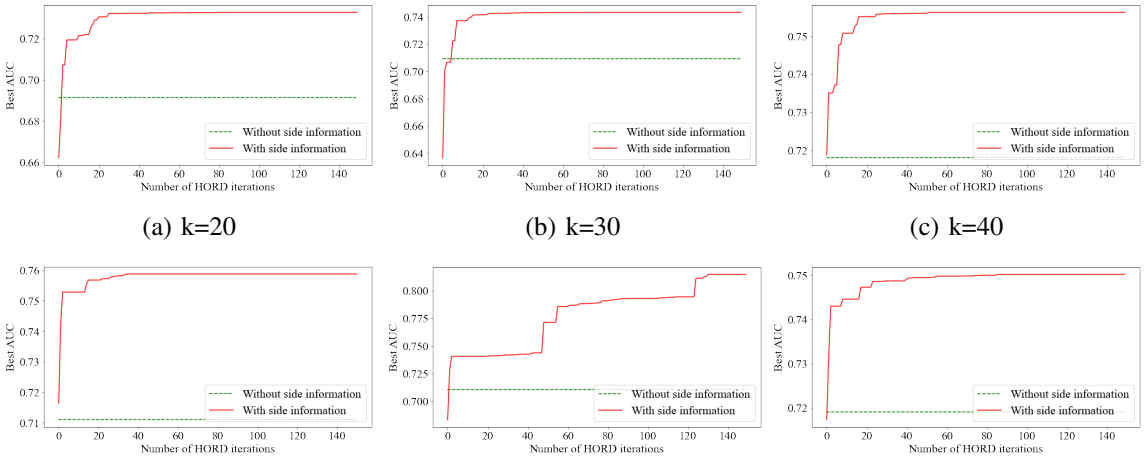
4.3 Results and Analysis

The experiment is run on a PC with Core i5-8400 CPU and 16 GB RAM. As listed in Table 3, our framework reaches a higher AUC compared with the naive LFM in each case.

Table 3: Result Comparison: AUC

k	**Without side information**	**With side information**	**Improvement**
20	0.692	0.733	5.9%
30	0.710	0.743	4.6%
40	0.720	0.756	5.0%
50	0.708	0.758	7.1%
60	0.710	0.819	15.5%
70	0.719	0.750	4.3%

Conducting 150 iterations of behavior weight tuning, Figure 4 plot the best AUC of prediction by our framework found versus the number of HORD iterations, for $k = 20, 30, \cdots, 70$ respectively. We also include the AUC output under our benchmark algorithm in the three figures.



(a) k=20 (b) k=30 (c) k=40

(d) k=50 (e) k=60 (f) k=70

Figure 4: Comparison of Our Proposed Algorithm with Naive LFM

It can be seen from the figures that AUCs of the model with side information surpass the naive Latent Factor Model at the very beginning for all cases. The case of $k = 60$ shows the most remarkable improvement (15.5%) brought by incorporating side information and the most outstanding performance on click behavior prediction. Note that AUC score achieves the highest when $k = 60$, we conclude that increasing latent factor number may not always lead to higher prediction accuracy. It also justifies the low-rank nature of the preference rating matrix.

The optimal weight with the highest AUC of 0.819 is also presented in Table 4. The click behavior has a low weight compared with some other behaviors such as commenting and liking. This result seems

reasonable because only users who are really interested in the content of the music card may have subsequent behaviors after clicking. In addition, commenting and viewing comments contribute the most to preference ratings. From the observation, we shed managerial insights video creators should become more aware and value the quality of content instead of the effect of eye-catching.

Table 4: Optimal weight with the highest AUC

Behavior	Click	Comment	Enter homepage	Share	View comments	Like
Weight	0.10	0.99	0.10	0.05	1.00	0.52

We also record the running time for our experiments. The time spent for the case of $k = 60$ is 18,346 seconds, which is efficient considering the limited capacity of our CPU. Compared with the complexity of the deep neural network architecture, the simplicity of our framework can significantly reduce the training and tuning time.

5 CONCLUSIONS AND FUTURE WORK

In this work, we present an interpretable recommender system to predict click behavior by incorporating multiple interaction data types, and conduct a case study with real data to illustrate the superiority of wisely utilizing multiple interaction data. In our model, we design a user-item interaction rating system defined as the weighted-sum of the multiple interaction variables, and use matrix factorization technique to capture the latent structure from the ratings of observable user-item interaction. By carefully tuning the weight vector assisted by deterministic-surrogate-based hyperparameter optimization method, we discern the rating mechanics contributed from various types of interaction, and achieve excellent discrimination without inclusion of user and item features such as user demographics or product specifications. The hyperparameter tuning method shores up the efficiency of our proposed recommender system.

In our case study, we observe a 15.5% improvement in AUC when incorporating multiple information. We also justify the low-rank nature of the preference rating matrix and shed managerial insights. Interpretability is important for informed decision making. Calibrating the weight vector allows the decision makers on video-sharing platforms to better measure the importance of various behaviors and value the quality of content.

The hyperparameter optimization method borrows the mixed-integer algorithm with radial basis function (RBF) surrogates proposed by (Ilievski, Akhtar, Feng, and Shoemaker 2017), we are interested in other hyperparameter optimization algorithms and hope to develop some other efficient heuristics that fits better to matrix factorization and recommender system applications.

Our approach focuses on interactions between users and items without the inclusion of user and item features such as user demographics or product specifications, and we emphasize the improvement of introducing multiple user-item interaction data sources. Although an AUC score exceeding 0.8 is already considered excellent, we are motivated to explore interpretable and efficient approaches to incorporate user and item features. In the future, we will adaptively aggregate more information, attempt some graph-based learning approaches such as graph neural network (GNN) and knowledge graph, and contribute to the field of interpretable AI.

REFERENCES

Bennett, J., S. Lanning et al. 2007. "The netflix prize". In *Proceedings of KDD Cup and Workshop*, Volume 2007, 35. Citeseer.

Bugliarello, E., S. Jain, and V. Rakesh. 2019. "Matrix Completion in the Unit Hypercube via Structured Matrix Factorization". *arXiv preprint arXiv:1905.12881*.

Farias, V. F., and A. A. Li. 2019. "Learning Preferences with Side Information". *Management Science* 65(7):3131–3149.

Feurer, M., and F. Hutter. 2019. "Hyperparameter Optimization". In *Automated Machine Learning*, 3–33. Springer, Cham.

Hernández-Lobato, J. M., M. W. Hoffman, and Z. Ghahramani. 2014. "Predictive Entropy Search for Efficient Global Optimization of Black-Box Functions". *Advances in Neural Information Processing Systems* 27.

Ilievski, I., T. Akhtar, J. Feng, and C. Shoemaker. 2017. "Efficient Hyperparameter Optimization for Deep Learning Algorithms Using Deterministic RBF Surrogates". In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 31.

Koren, Y. 2008. "Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model". In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 426–434.

Koren, Y., R. Bell, and C. Volinsky. 2009. "Matrix Factorization Techniques for Recommender Systems". *Computer* 42(8):30–37.

Mockus, J. 1994. "Application of Bayesian Approach to Numerical Methods of Global and Stochastic Optimization". *Journal of Global Optimization* 4(4):347–365.

Mockus, J., V. Tiesis, and A. Zilinskas. 1978. "The Application of Bayesian Methods for Seeking the Extremum". *Towards Global Optimization* 2(117-129):2.

New York Times 2021a. "How a Mistake by YouTube Shows Its Power over Media". https://www.nytimes.com/2021/10/29/business/youtube-novara.html.

New York Times 2021b. "Netflix and the Internet of Fads". https://www.nytimes.com/2021/12/07/technology/netflix-tiktok-fads.html.

Quan, Y., K. Wang, and S. Guan. 2021. "NetEase Cloud Dataset: Active User Identification and Deep Neural Network Based CTR". In *2021 5th International Conference on Cloud and Big Data Computing (ICCBDC)*, 39–45.

Regis, R. G. 2014. "Particle Swarm with Radial Basis Function Surrogates for Expensive Black-Box Optimization". *Journal of Computational Science* 5(1):12–23.

Regis, R. G., and C. A. Shoemaker. 2013. "Combining Radial Basis Function Surrogates and Dynamic Coordinate Search in High-dimensional Expensive Black-Box Optimization". *Engineering Optimization* 45(5):529–555.

Singh, A. P., and G. J. Gordon. 2008. "Relational Learning via Collective Matrix Factorization". In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 650–658.

Snoek, J., H. Larochelle, and R. P. Adams. 2012. "Practical Bayesian Optimization of Machine Learning Algorithms". *Advances in Neural Information Processing Systems* 25.

Wang, Z., Y. Guo, and B. Du. 2018. "Matrix Completion with Preference Ranking for Top-N Recommendation.". In *IJCAI*, 3585–3591.

Washington Times 2012. "Key to Netflix's Future: Better Recommendations". https://www.washingtontimes.com/news/2012/apr/9/key-to-netflixs-future-better-recommendations/?utm_source=RSS_Feed&utm_medium=RSS.

Zhang, D. J., M. Hu, X. Liu, Y. Wu, and Y. Li. 2022. "NetEase Cloud Music Data". *Manufacturing & Service Operations Management* 24(1):275–284.

AUTHOR BIOGRAPHIES

CHEN FENG is a current Ph.D. student from the School of Industrial Engineering at Purdue University. His research interests include game theory, mechanism design, and reinforcement learning. His email address is feng219@purdue.edu.

RUIJIU MAO is a current Ph.D. student from the Institute of Operations and Analytics at National University of Singapore. Her research interests include applications of data analytics in revenue management and

supply chain management. She is the corresponding author and her email address is maoruijiu@u.nus.edu.