AdaF^{2}M^2: Comprehensive Learning and Responsive Leveraging Features in Recommendation System

Yongchun Zhu, Jingwu Chen✉, Ling Chen, Yitan Li,
Feng Zhang, and Zuotao Liu

ByteDance, China
zhuyongchun.zyc,chenjingwu,chenling.nb,liyitan,
feng.zhang,michael.liu@bytedance.com

Abstract. Feature modeling, which involves feature representation learning and leveraging, plays an essential role in industrial recommendation systems. However, the data distribution in real-world applications usually follows a highly skewed long-tail pattern due to the popularity bias, which easily leads to over-reliance on ID-based features, such as user/item IDs and ID sequences of interactions. Such over-reliance makes it hard for models to learn features comprehensively, especially for those non-ID meta features, e.g., user/item characteristics. Further, it limits the feature leveraging ability in models, getting less generalized and more susceptible to data noise. Previous studies on feature modeling focus on feature extraction and interaction, hardly noticing the problems brought about by the long-tail data distribution. To achieve better feature representation learning and leveraging on real-world data, we propose a model-agnostic framework AdaF^{2}M^2, short for **A**daptive **F**eature **M**odeling with **F**eature **M**ask. The feature-mask mechanism helps comprehensive feature learning via multi-forward training with augmented samples, while the adapter applies adaptive weights on features responsive to different user/item states. By arming base models with AdaF^{2}M^2, we conduct online A/B tests on multiple recommendation scenarios, obtaining +1.37% and +1.89% cumulative improvements on user active days and app duration respectively. AdaF^{2}M^2 has been widely deployed on both retrieval and ranking tasks in multiple applications of Douyin Group, indicating its superior effectiveness and universality.

Keywords: Recommendation, Feature Representation

1 Introduction

Video platforms like Douyin and TikTok are extremely hot nowadays, with billions of users. The progress of personalized recommendation systems has made a significant contribution to the success of those platforms, which helps expose attractive items to users according to their interests. While tracing the source

of personalization, features play an essential role by providing the original information of users, items, and interactions between them. Actually, feature engineering has been very important in industrial recommendation systems from past to present.

Conventional matrix factorization (MF) methods [4] work by decomposing the user-item interaction matrix into the product of two lower dimensional matrices, where each row represents a user/item ID embedding. After MF, Factorization Machines (FMs) [9] leverage more features in addition to ID and conduct pair-wise interactions via the inner product of two embeddings. With more computing firepower and the growing scale of data, recommendation models in industry have migrated from FM-based models to Deep Neural Networks (DNN) [2], which significantly improve model performance. DNNs have the ability to model more complex feature interactions [1, 15, 12], thus being able to handle more types of feature inputs, e.g., image and text. Reviewing the development history of recommendation models, the utilization of features is getting better, and more complex features are becoming acceptable, varying from ID to multimodal features.

In practice, recommendation models usually take lots of features as inputs. With diverse features engaged in, feature modeling is crucial for final performance. Well-learned feature representations can help models generalize better, handle noise and uncertainty in the data, and uncover hidden patterns. However, the data distribution in industrial recommendation systems usually follows a highly skewed long-tail pattern due to the popularity bias, e.g., 5% of users/items account for over 80% of samples. Head users probably dominate the model learning with a large proportion of samples. Since the user/item ID is the most fine-grained and the most informative feature for users with a large amount of behavioral data, the model also tends to attribute more knowledge to IDs. This easily leads to over-reliance on ID-based features, such as user/item IDs and ID sequences of interactions. Such over-reliance will seriously affect the comprehensive learning of features, making the model less generalized and more vulnerable to noise interference. Taking data noise as an example, user behavior is often interfered with unobserved factors. For instance, a user might skip a favorite video just because they received a phone call at that time, but such noisy pattern can be wrongly fitted by the user ID.

Further, the above over-reliance also limits the learning of non-ID meta features, causing poor performance in the user/item cold-start stage. For example, recommending a new video with few samples highly relies on its attribute features such as theme and category. The weakening of feature representation learning will make the feature leveraging less responsive to different user/item states.

In this paper, to tackle problems brought about by the long-tail distribution, we propose a simple but powerful framework Adaptive Feature Modeling with Feature Mask (**AdaF^{2}M^2**), which consists of a feature-mask mechanism and a state-aware adapter. We generate multiple augmented samples with the feature-mask mechanism and learn features comprehensively via multi-forward task-oriented training on these samples. To adaptively model users/items of dif-

ferent states, we propose the state-aware adapter, which takes empirical state signals as input, to apply adaptive weights on features responsive to different user/item states. Note that it is convenient to deploy AdaF^{2}M^2 with different base recommendation models. AdaF^{2}M^2 has been widely deployed on both retrieval and ranking tasks in multiple applications of Douyin Group, indicating its superior effectiveness and universality.

2 Related Work

Feature Representation Learning. Researchers have investigated feature representation learning for a long time. Many researchers pay attention to feature interaction which aims at capturing associations among different features and producing more diverse combinations of features to improve feature representations [9, 11, 14, 15, 3, 10, 19, 5, 12]. Besides, many methods focus on improving representations of some specific features, e.g., item ID [8], user ID [7], sequential features [20], multi-modal features [16]. In addition, some researchers make efforts to enhance representations with novel training frameworks. DropoutNet [13] applies dropout to ID embeddings, which improves the recommendation performance for cold-start users and items. Yao et al. [18] exploits self-supervised learning to improve item representation learning as well as serve as additional regularization to improve generalization. However, these methods ignore the importance of effective feature representation leveraging for users/items of different states and lack additional supervised signals.

Feature Representation Leveraging. The useless features may introduce noise and complicate the training process. Feature representation leveraging aims at identifying useful feature interactions through model training. The attention mechanism is convenient to model importance of features, and many methods [10, 19] utilize it for the feature selection. AFN [1] takes learnable parameters for feature selection. AutoFIS [6] exploited a two-stage framework, which identifies the feature importance and retrains the model without redundant feature interactions. However, these methods detect beneficial features by learning from the features themselves, which could be dominated by the head users and items, and the ID-based features are usually assigned with high weights. In this paper, we propose a state-aware adapter taking empirical state signals as input responsive to different user/item states.

3 Proposed Framework

3.1 Recommendation Task Setup

First, we consider the common setup for a binary classification task, such as CTR predicting in recommendation systems. Each sample consists of the input raw features $\boldsymbol{x} = [x_1, \cdots, x_n]$ and a label $y \in \{0, 1\}$, where n indicates the number of raw features. In binary classification, a deep recommendation model approximates the probability $\hat{y} = Pr(y = 1|\boldsymbol{x})$ for the sample with input $\boldsymbol{x}$.

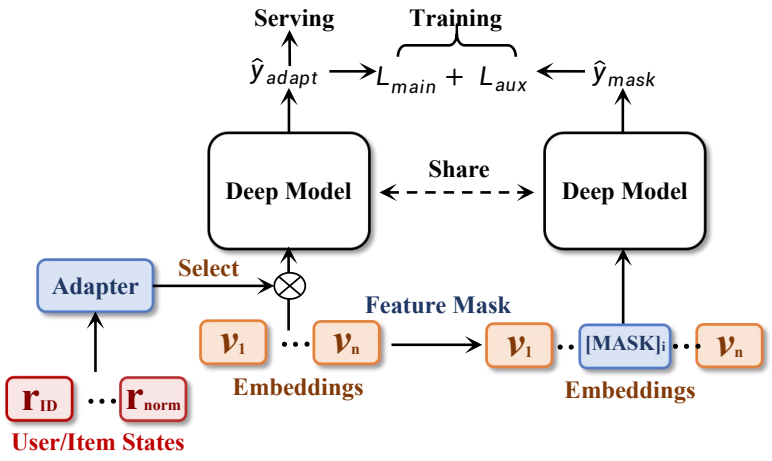

Fig. 1. Adaptive Feature Modeling with Feature Mask Framework (AdaF^{2}M^2).

Generally, a deep recommendation model consists of a feature embedding layer, a feature interaction layer, and a deep network. The feature embedding layer aims to transform raw features $[x_1, \cdots, x_n]$ into low-dimensional representations, named feature embeddings, denoted as $[\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n]$. Then, the feature interaction layer takes the embeddings as input to generate high-order cross-feature representations, and most existing methods focus on this layer, e.g., FM [9], DCN [15]. Finally, the cross-feature representations are fed into a deep network for the prediction. In this paper, we focus on improving the representation of the embedding layer, and the interaction layer and the deep network are denoted as $g(\cdot)$. The prediction of a recommendation model is formulated as $\hat{y}$.

3.2 Feature Mask

We propose a feature-mask mechanism to enhance representations via multi-forward task-oriented training with augmented samples, which enables more comprehensive feature learning. The key idea is to get rid of the over-reliance on important features by randomly masking part of all features. We create multiple augmented samples by feature-mask and train on these samples with a task-oriented loss. The augmented samples can force the model to make predictions based on diverse combinations of features, which enables all features to be learned well.

First, for each instance, we sample k times randomly from a range $[\beta, \gamma]$ and obtain k values, where β and γ represent a range of sampling probability (setting as 0.1 and 0.5 in this paper). Each value p indicates the probability of replacing a feature embedding $\boldsymbol{v}$ with a default mask embedding $[MASK]$. Note that the default mask embedding of each feature is different. Given the masking probability p, we randomly mask some feature embeddings of a sample

with corresponding default mask embeddings to generate an augmented sample. The feature embeddings of an augmented sample are formulated as:

$$[\boldsymbol{v}_1, \cdots, [MASK]_i, \cdots, [MASK]_j, \cdots, \boldsymbol{v}_n], \tag{1}$$

where $[MASK]_i$ indicates the default embedding of the i-th feature. With k times random masking, k augmented samples are generated from a given sample.

The second step is learning with the augmented samples. Inspired by recent natural language processing (NLP) and computer vision (CV) techniques, most existing recommendation methods [18] also use self-supervised learning for the augmented samples. The main idea of these methods is that let augmented data from the same sample be discriminated against others, which can improve the discriminability of the representations of users/items. However, the recommendation tasks are different from NLP and CV in two ways: (1) Most tasks of CV and NLP lack sufficient labeled samples, but there are a large amount of unlabeled samples in the real world. However, in recommendation systems, there are billions of labeled samples every day (the feedbacks of users are utilized as labels), and the number of labeled samples is more than the unlabeled samples (the items with no interaction). (2) The prerequisite for many tasks of CV and NLP is understanding the content (text and image), so they need to differentiate between different samples/contents. However, the goal of recommendation models is to make accurate predictions, rather than distinguish different users/items.

Along this line, we propose multi-forward training with a task-oriented optimization procedure to learn the augmented samples. In detail, the concatenated embeddings of an augmented sample as Equation (1) are fed into the deep network $g(\cdot)$ to generate the predicted result $\hat{y}$, formulated as:

$$\hat{y}_{mask} = g([\boldsymbol{v}_1, \cdots, [MASK]_i, \cdots, \boldsymbol{v}_n]). \tag{2}$$

Then, we feed k augmented samples into deep networks to obtain k prediction $\hat{y}_{mask\cdot}$ A task-oriented optimization procedure (directly computing the target loss) is adopted, which applies additional supervision over the prediction of the augmented samples, denoted as:

$$\mathcal{L}_{aux} = \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \sum_{i=1}^{k} -y \log(\hat{y}^i_{mask}) - (1-y)\log(1-\hat{y}^i_{mask}), \tag{3}$$

where $\mathcal{D}$ indicates the training dataset and $\hat{y}^i_{mask}$ indicates the prediction of the i-th random augmented sample. Then, we apply stochastic gradient descent to update all parameters of the deep model and all embeddings.

The feature-mask mechanism which generates multiple augmented samples can help comprehensive feature learning via multi-forward training with task-oriented optimization. It has two main advantages: (1) In real-world recommendation systems, it is common that some features of a sample are missing or mislabeled. The augmented samples are used to simulate the noisy condition of the real-world recommendation systems, which can improve the robustness and

generalizability of the model. (2) With the feature-mask mechanism, any features are possible to be masked, including ID-based personalized features. Feeding inputs without personalized features to the deep network can force the model to pay attention to the remained non-ID meta features, e.g., age and gender. Since we adopt task-oriented optimization, the model directly makes predictions based on these unmasked features, which can learn them better.

3.3 Adaptive Feature Modeling

In this section, we propose adaptive feature modeling with a state-aware adapter, which can assign adaptive feature weights for users and items with different states. The deep model takes all feature embeddings $[\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m]$ as input, and the adaptive feature modeling aims to assign adaptive feature weights, and the input with weights can be formulated as $[a(\boldsymbol{v}_1), \cdots, a(\boldsymbol{v}_n)]$, where $a(\boldsymbol{v}_i) = w_i \boldsymbol{v}_i$, where w_i indicates the adaptive weight for the i-th feature.

How to generate adaptive weights is the most important key of adaptive feature modeling. The existing methods [17, 3, 10, 19, 1] mainly generate the weights from the feature themselves, which could be formulated as: $[w_1, \cdots, w_n] = h([\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n])$, where the $h(\cdot)$ represents the weight generator. However, the learning process is dominated by the head users/items, and the generator would assign bigger weights for the important features of the head users and items. Generally, ID-based personalized features, e.g., ID and sequential features, are important in these methods.

To tackle the problem of over-reliance on ID-based features in existing methods, we propose a state-aware adapter, which can assign adaptive weights to features according to the states of users/items. To enable the adapter to perceive the different states, we propose three kinds of empirical state signals: (1) **ID embedding**: We concatenate all ID embedding (user, item, and artist ID) into a vector, denoted as $\boldsymbol{r}_{ID}$. (2) **Norm of ID embedding**: Generally, the norm of ID embedding can indicate the quality of ID embedding, e.g., the norm of old users' ID embeddings is bigger than the new users'. We concatenate the norm of various ID embeddings and use different non-linear functions (log, sqrt, square) to enhance the representations of the norm, denoted as $\boldsymbol{r}_{norm}$. (3) **Interaction count**: The number of interactions is a strong signal to distinguish the states of users and items. Thus, we utilize the numbers of 'impression', 'add comment', 'click like button' of users and items, indicated as $\boldsymbol{r}_{count}$.

Thus, the state-aware adapter takes the concatenated state embeddings as input to assign adaptive weight for users/items with different states, formulated as: $[w_1, \cdots, w_n] = \sigma(h([\boldsymbol{r}_{ID}, \boldsymbol{r}_{norm}, \boldsymbol{r}_{count}]))$, where $\sigma(\cdot)$ indicates the Sigmoid function. With the guide of the strong empirical signals, the adapter has better ability to generate suitable distributions of weights for different states of users and items.

In addition, the adaptive mechanism in existing methods raised another problem it can seriously influence the learning process of feature embeddings. In detail, the prediction with adaptive weights can be formulated as:

$$\hat{y} = g([a(\boldsymbol{v}_1), \cdots, a(\boldsymbol{v}_n)]). \tag{4}$$

By computing the gradient and performing a gradient descent step, we obtain a new embedding parameter:

$$\boldsymbol{v}' = \boldsymbol{v} - \lambda \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = \boldsymbol{v} - \lambda \frac{\partial \mathcal{L}}{\partial g} \frac{\partial g}{\partial a} \frac{\partial a}{\partial \boldsymbol{v}} = \boldsymbol{v} - \lambda w \frac{\partial \mathcal{L}}{\partial g} \frac{\partial g}{\partial a}, \tag{5}$$

where λ indicates the learning rate. Equation (5) shows the gradient of embedding $\boldsymbol{v}$ is related to the feature weight w. In other words, a feature assigned with a low weight would be learned with a small gradient (vanishing gradients). The non-ID meta features which are important for long-tail users/items could be assigned with low weights for high-active users that accounts for most samples. Thus, it is more difficult to learn these features, while the long-tail users/items only accounts a small number of samples.

To alleviate the problem of vanishing gradients in existing methods, we propose multi-forward training with different augmented samples. In detail, we train the state-aware adapter using the samples without random masking. In addition, for comprehensive feature representation learning, the embedding is learned with augmented samples generated by random masking. Thus, the adaptive weights have no influence on feature representation learning.

3.4 Overall Framework

The overall framework of AdaF^{2}M^2 is shown in Figure 1, which consists a state-aware adapter and a feature-mask mechanism. The deployment of the AdaF^{2}M^2 framework consists of two stages: the training stage and the serving stage.

Training stage: The final probability prediction with the weights generated by the state-aware adapter is formulated as $\hat{y}_{adapt} = g([a(\boldsymbol{v}_1), \cdots, a(\boldsymbol{v}_n)])$, where the $g(\cdot)$ indicates the prediction function. We train the framework, including embeddings, the deep network, the adapter, with the cross-entropy loss:

$$\mathcal{L}_{main} = \sum_{(\boldsymbol{x},y)\in\mathcal{D}} -y \log(\hat{y}_{adapt}) - (1-y)\log(1-\hat{y}_{adapt}), \tag{6}$$

where $\mathcal{D}$ indicates the training dataset. For comprehensive feature learning, we utilize the auxiliary loss as shown in Equation (3). Note that the adaptive weights are not one of the input of the Equation (2). The overall loss function can be formulated as $\mathcal{L} = \mathcal{L}_{main} + \alpha \mathcal{L}_{aux}$, where the α denotes a hyper-parameter, which is set as 0.2 in this paper. With the training procedure, AdaF^{2}M^2 can comprehensively learn features and apply adaptive feature weights responsively to different user/item states.

Serving stage: The serving stage aims at delivering attractive items for the right users, and the model needs to predict the probability that the user might click/like/share each item. $\hat{y}_{adapt}$ is directly used as the final prediction in the serving stage. Note that the additional forwards with randomly masked features as input are only utilized in the training stage. Thus, the feature-mask mechanism has no impact on the latency of the serving stage.

Table 1. Online A/B testing results of the ranking task. Each row indicates the relative improvement with our AdaF^{2}M^2 framework over the baseline (a DCN-V2 based multi-task model). Statistically-significant improvement is marked with bold font in the table.

State	Main Metrics		Constraint Metrics			
	Active Days	Duration	Like	Finish	Comment	Dislike
New users	**+0.333%**	**+0.540%**	**+0.562%**	**+0.418%**	**+0.402%**	-2.246%
Old users	**+0.206%**	**+0.434%**	**+0.387%**	**+0.157%**	**+0.230%**	-1.394%
Overall	**+0.212%**	**+0.442%**	**+0.418%**	**+0.224%**	**+0.262%**	**-1.594%**

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate AdaF^{2}M^2 with baselines on both public and large-scale industrial recommendation datasets.

DouyinMusic: Douyin provides a music recommendation service, with over 10 million daily active users. We collect from the impression logs and get two datasets with different sizes. Besides, there is more than a month gap between the two sampled datasets to test the effectiveness of the proposed framework in different periods. The small dataset contains more than 4 billion samples, denoted as *DouyinMusic-4B*. The large dataset contains more than 20 billion samples, denoted as *DouyinMusic-20B*. Each sample of the industrial datasets contains more than one hundred features, including both non-ID meta features (gender, age, genre, mood, scene, and so on) and ID-based personalized features (user ID, item ID, artist ID, interacted ID sequence), which can represent the real-world scenarios. We use 'Finish' as the label.

MovieLens-1M[1]: It is one of the most well-known benchmark datasets. The data consists of 1 million movie ranking instances over thousands of movies and users. Each movie has features including its title, year of release, and genres. Titles and genres are lists of tokens. Each user has features including the user's ID, age, gender, and occupation. We transform ratings into binary (The ratings of at least 4 are turned into 1 and the others are turned into 0). The dataset is randomly divided into train / validation / test sets in the ratio of 6:2:2.

Baselines. We categorize our baselines into two groups according to their approaches. The first group includes the popular retrieval methods, including: *Factorization Machines(FM)* [9], *YouTube DNN* [2]. The second group includes state-of-the-art methods for ranking tasks in recommendation systems, including *Adaptive Factorization Network(AFN)* [1], *Deep & Cross Network V2(DCN-V2)* [15], and *EulerNet* [12]. The proposed AdaF^{2}M^2 is a model-agnostic framework that can be applied upon various models, and we apply AdaF^{2}M^2 on the above five base models to demonstrate its effectiveness and universality.

[1] http://www.grouplens.org/datasets/movielens/

Table 2. Offline results (RelaImpr of AUC and UAUC) on the industrial datasets DouyinMusic-20B and DouyinMusic-4B.

	DouyinMusic-20B		DouyinMusic-4B	
Method	AUC	UAUC	AUC	UAUC
FM	**+0.41%**	**+0.26%**	**+0.46%**	**+0.37%**
YouTube DNN	**+0.15%**	**+0.17%**	**+0.18%**	**+0.22%**
AFN	**+0.25%**	**+0.13%**	**+0.23%**	**+0.15%**
DCN-V2	**+0.21%**	**+0.15%**	**+0.45%**	**+0.71%**
EulerNet	**+0.19%**	**+0.17%**	**+0.36%**	**+0.43%**

Table 3. Offline results (AUC, RelaImpr) on the public dataset MovieLens-1M.

Method	AUC	AUC (w/ AdaF^{2}M^2)	RelaImpr
FM	0.7767	**0.7808**	+0.53%
YouTube DNN	0.7857	**0.7871**	+0.18%
AFN	0.7878	**0.7889**	+0.14%
DCNV2	0.7886	**0.7905**	+0.24%
EulerNet	0.7911	**0.7921**	+0.13%

4.2 Online A/B Testing

To verify the real benefits AdaF^{2}M^2 brings to our system, we conducted online A/B testing experiments for more than two weeks for the ranking and retrieval tasks in Douyin Music App respectively.

Online Metrics. We evaluate model performance based on two main metrics, Active Days and Duration, which are widely adopted in practical recommendation systems. The total days that users in the experimental bucket open the application are denoted as Active Days. The total amount of time spent by users in the experimental bucket on staying in the application is denoted as Duration. We also take additional metrics, which evaluate user engagement, including Like/Dislike (clicking the like/dislike button on the screen), Finish (hearing the end of a song), and Comment (leaving comments on a song), which are usually used as constraint metrics. We calculate all online metrics per user.

Ranking Tasks. We apply the proposed AdaF^{2}M^2 on a DCN-V2-based multi-task model which is deployed in the online ranking tasks and conduct an online A/B testing to demonstrate the effectiveness of AdaF^{2}M^2 to improve ranking models. The online A/B results of new users (the registration time is less than X days from now, and X is a predefined threshold), old users (the registration time is greater than X days), and whole users are shown in Table 1. For the main metrics Active Days and Duration, the proposed AdaF^{2}M^2 achieves a large improvement of +0.212% and +0.442% for all users with statistical significance, which is remarkable given the fact that the average Active Days and Duration improvement from algorithms is around 0.05% and 0.1%, respectively.

Table 4. Online and offline ablation experiments of the retrieval tasks. We use a two-tower model as the base model and show the relative improvement.

Method	Online Metrics		Offline Metrics	
	Active Days	Duration	AUC	UAUC
w/ Feature Mask	+0.006%	+0.078%	+0.05%	+0.03%
w/ Adapter	+0.022%	+0.091%	**+0.12%**	**+0.11%**
w/ AdaF^{2}M^2	**+0.073%**	**+0.184%**	**+0.15%**	**+0.17%**
w/ MaskNet	+0.033%	+0.048%	**+0.14%**	**+0.12%**

Retrieval Tasks. We deploy AdaF^{2}M^2 on two online retrieval models, including an FM-based model and a two-tower model that is similar to YouTube DNN [2]. We find the proposed AdaF^{2}M^2 framework achieves a significant improvement on Active Days (+0.066% and +0.073%) and Duration (+0.195% and +0.184%), against the two base retrieval models. Especially, an upgrade of a retrieval model can increase about 0.07% Active Days, which is a very significant improvement. In addition, with different base models, AdaF^{2}M^2 is effective in improving the recommendation performance, which demonstrates that AdaF^{2}M^2 has satisfying universality.

4.3 Offline Experiments

In this section, we conduct extended offline experiments with one public dataset and two industrial datasets. Specifically, we apply the proposed AdaF^{2}M^2 framework upon five base models to testify the effectiveness and universality.

Experimental Details. For all methods, the initial learning rate for the Adam optimizer are tuned by grid searches within {0.001, 0.005, 0.01, 0.02, 0.1}. In addition, we tune the dimension of embeddings in {16, 32, 64, 128, 256}. For all methods, we set a mini-batch size of 256. Statistically significant improvement is marked with bold font in the tables.

Offline Metrics. For binary classification tasks, AUC is a widely used metric. It measures the goodness of order by ranking all the items with prediction, including intra-user and inter-user orders. Besides, AUC is a common metric for recommendation [20]. In addition, we introduce another metric in our business, named UAUC, which calculates AUC per user and then averages scores with weights proportional to the user's sample size. A larger UAUC suggests better model performance. For the industrial datasets, we report the relative improvement (RelaImpr) of AUC and UAUC over base models (the limitation of our company). The public dataset only contains 6,000 users and the user-level AUC(UAUC) is not very confident. Thus, following most existing methods [12], we report AUC and RelaImpr.

Offline Results. The experimental results on industrial and public datasets are shown in Table 2 and Table 3, respectively. The results further reveal several insightful observations. With the help of the proposed AdaF^{2}M^2, most base

models show a significant improvement, which demonstrates the effectiveness and universality of the proposed framework. The improvement mainly comes from the better feature representation learning and leveraging mechanisms. In addition, it also testifies that AdaF^{2}M^2 is a model-agnostic framework that can be applied upon various base models.

4.4 Ablation Study

To test the effectiveness of each module in AdaF^{2}M^2, we present an ablation study. We introduce three kinds of models, w/ Feature Mask, w/ Adapter, and w/ AdaF^{2}M^2, which add the feature-mask mechanism, the adapter, and the overall AdaF^{2}M^2 based on a two-tower model, respectively. The online and offline results are shown in Table 4. The results show that the base model with AdaF^{2}M^2 achieves better performance than the model only with Feature Mask or Adapter, which demonstrates each module is effective. In addition, we find that the model with the adapter can improve the offline performance, but there is no improvement in the online results. The main reason could be the gap between the offline and online metrics. The offline metrics (AUC and UAUC) are dominated by the performance of the head users, while the online metrics average scores with the same weights for all users. In addition, to demonstrate the proposed AdaF^{2}M^2 is better than existing mask methods with self-supervised learning (SSL), we conduct another online A/B testing that replaces the AdaF^{2}M^2 with MaskNet [18]. The results show that AdaF^{2}M^2 is better than MaskNet with the self-supervised loss in recommendation systems.

5 Conclusion

In this paper, for better feature representation learning and leveraging in recommendation systems, we propose Adaptive Feature Modeling with Feature Mask (AdaF^{2}M^2), a novel framework consisting of feature-mask mechanism and a state-aware adapter. AdaF^{2}M^2 can comprehensively learn features and adaptively leverage features responsive to different user/item states. We demonstrated the superior performance of the proposed AdaF^{2}M^2 in offline experiments. In addition, we conducted online A/B testing in both ranking and retrieval tasks on multiple recommendation scenarios, obtaining +1.37% and +1.89% cumulative improvements on user active days and app duration respectively, which demonstrates the effectiveness and universality of AdaF^{2}M^2 in online systems. Moreover, AdaF^{2}M^2 has been deployed on both ranking and retrieval tasks in multiple applications of Douyin Group.

References

1. Cheng, W., Shen, Y., Huang, L.: Adaptive factorization network: Learning adaptive-order feature interactions. In: AAAI. vol. 34, pp. 3609–3616 (2020)

2. Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: RecSys. pp. 191–198 (2016)
3. Huang, T., Zhang, Z., Zhang, J.: Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In: RecSys. pp. 169–177 (2019)
4. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
5. Liu, B., Xue, N., Guo, H., Tang, R., Zafeiriou, S., He, X., Li, Z.: Autogroup: Automatic feature grouping for modelling explicit high-order feature interactions in ctr prediction. In: SIGIR. pp. 199–208 (2020)
6. Liu, B., Zhu, C., Li, G., Zhang, W., Lai, J., Tang, R., He, X., Li, Z., Yu, Y.: Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In: KDD. pp. 2636–2645 (2020)
7. Man, T., Shen, H., Jin, X., Cheng, X.: Cross-domain recommendation: an embedding and mapping approach. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 2464–2470 (2017)
8. Pan, F., Li, S., Ao, X., Tang, P., He, Q.: Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings. In: SIGIR. pp. 695–704 (2019)
9. Rendle, S.: Factorization machines. In: ICDM. pp. 995–1000 (2010)
10. Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., Tang, J.: Autoint: Automatic feature interaction learning via self-attentive neural networks. In: CIKM. pp. 1161–1170 (2019)
11. Sun, Y., Pan, J., Zhang, A., Flores, A.: Fm2: Field-matrixed factorization machines for recommender systems. In: WWW. pp. 2828–2837 (2021)
12. Tian, Z., Bai, T., Zhao, W.X., Wen, J.R., Cao, Z.: Eulernet: Adaptive feature interaction learning via euler's formula for ctr prediction. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (2023)
13. Volkovs, M., Yu, G., Poutanen, T.: Dropoutnet: addressing cold start in recommender systems. In: NIPS. pp. 4964–4973 (2017)
14. Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. In: ADKDD, pp. 1–7 (2017)
15. Wang, R., Shivanna, R., Cheng, D., Jain, S., Lin, D., Hong, L., Chi, E.: Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In: WWW. pp. 1785–1797 (2021)
16. Wei, Y., Wang, X., Nie, L., He, X., Hong, R., Chua, T.S.: Mmgcn: Multi-modal graph convolution network for personalized recommendation of micro-video. In: ACMMM. pp. 1437–1445 (2019)
17. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.S.: Attentional factorization machines: learning the weight of feature interactions via attention networks. In: IJCAI. pp. 3119–3125 (2017)
18. Yao, T., Yi, X., Cheng, D.Z., Yu, F., Chen, T., Menon, A., Hong, L., Chi, E.H., Tjoa, S., Kang, J., et al.: Self-supervised learning for large-scale item recommendations. In: CIKM. pp. 4321–4330 (2021)
19. Zhang, K., Qian, H., Cui, Q., Liu, Q., Li, L., Zhou, J., Ma, J., Chen, E.: Multi-interactive attention network for fine-grained feature learning in ctr prediction. In: WSDM. pp. 984–992 (2021)
20. Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., Yan, Y., Jin, J., Li, H., Gai, K.: Deep interest network for click-through rate prediction. In: KDD. pp. 1059–1068 (2018)