Boosting Factorization Machines via Saliency-Guided Mixup

Chenwang Wu, Defu Lian, Yong Ge, Min Zhou, Enhong Chen, Senior Member, IEEE, and Dacheng Tao, Fellow, IEEE

Abstract—Factorization machines (FMs) are widely used in recommender systems due to their adaptability and ability to learn from sparse data. However, for the ubiquitous non-interactive features in sparse data, existing FMs can only estimate the parameters corresponding to these features via the inner product of their embeddings. Undeniably, they cannot learn the direct interactions of these features, which limits the model’s expressive power. To this end, we first present MixFM, inspired by Mixup, to generate auxiliary training data to boost FMs. Unlike existing augmentation strategies that require labor costs and expertise to collect additional information such as position and fields, these augmented data are only by the convex combination of the raw ones without any professional knowledge support. More importantly, if non-interactive features exist in parent samples to be mixed respectively, MixFM will establish their direct interactions. Second, considering that MixFM may generate redundant or even detrimental instances, we further put forward a novel Factorization Machine powered by Saliency-guided Mixup (denoted as SMFM). Guided by the customized saliency, SMFM can generate more informative neighbor data. Through theoretical analysis, we prove that the proposed methods minimize the upper bound of the generalization error, which positively enhances FMs. Finally, extensive experiments on seven datasets confirm that our approaches are superior to baseline methods. Notably, the results also show that “poisoning” mixed data benefits the FM variants.

Index Terms—Recommender systems, factorization machines, sparse data.

I. INTRODUCTION

The rapid advancement of information technology has exacerbated information overload. As an effective way to alleviate information overload, recommender systems have attracted many researchers to study [1], [2], [3], [4]. In the practical recommendation task, the sample’s features are mostly categorical, making them highly sparse and difficult to learn after one-hot encoding. For this reason, factorization machines (FMs) [5] are proposed. On the one hand, FMs can learn model parameters and task predictions in linear time, which is necessary for large-scale systems. On the other hand, FMs can effectively estimate highly sparse features’ interactions and generalize to unobserved ones. The flexibility and effectiveness make them widely used in e-commerce [6], location services [7], and social networks [8].

Despite the pleasant expression capability for sparse data, the performance of FM is heavily dependent on real-world datasets with complex and nonlinear underlying structures [9] (e.g., overfitting to the infrequently-occurring features or underfitting to the frequent features [10]). Accordingly, it has remained challenging to perform high-quality predictions when facing data outside the learned discrete feature space. For this purpose, some work attempted to enhance the models by incorporating additional information (beyond the basic interactive features), e.g., Fields [11], time information [12], multi-view [13], and position information [14]. Apparently, such methods must ensure that additional information is available or easy to collect. Apart from the time-consuming labor costs, they also need the support of professional knowledge, which limits their applicability. Therefore, it is necessary to study efficient data augmentation without additional knowledge or manual intervention.

In addition, the key of FM is feature interaction, but it is pretty common for some feature interactions to be absent or not observed under limited sparse data. For instance, in the Frappe dataset, 24.84% of feature interactions in the test set are unobserved in the training set. If the feature interaction is not previously encountered in the model, how can we ensure that FMs can comprehend the behavior of the sample containing such unknown feature interaction? For example, if an e-commerce recommender system is uninformed about the opinion of users who purchase books (Feature A) regarding bookmarks (Feature B), on what grounds can we predict the behavior of a book-loving user concerning bookmarks? However, if the system recurrently observes that customers purchasing books concurrently do (or do not) appreciate buying bookmarks, then the model can discern this collaboration, thus facilitating predictions for novel users exhibiting similar behavior. So intuitively, Constructing more
explicit feature interactions may help learn from ignored users outside the training set, thereby enhancing the recommendation generalization.

Given the aforementioned deficiencies and inspired by the augmentation capability of Mixup [15] in computer vision, we first propose Mixup powered Factorization Machine (MixFM) to enhance FMs. It is the first attempt of Mixup at sparse recommendations, which can simultaneously consider the knowledge limitations and explicit interaction problem of non-interactive features. Specifically, through the raw samples’ convex combination, MixFM generates many neighbor data to enrich the dataset without professional knowledge. These auxiliary data enable MixFM to process the regions between natural sparse samples in a continuous linear manner, thereby alleviating the dependence on raw data. Besides, if the non-interactive feature pair exists respectively in the parent samples, MixFM will inherit these features and establish their direct interaction. Second, seeing that the randomness of neighbor data generated by MixFM may produce meaningless data, we further develop a novel Saliency-guided Mixup powered Factorization Machine (SMFM). In SMFM, we provide a solver to efficiently calculate the sample’s saliency, guiding SMFM to generate more informative data. We theoretically prove that our proposals hold a smaller generalization than FM. Besides, we give the first generalization bound of FM and provably reveal the impact of embedding and data on recommendation generalization.

The contributions are multi-fold:

- We give the first generalization bound of factorization machines. The bound reveals that the generalization requires more data, and a smaller embedding size under the sufficient representation capability of FM, which sheds light on further enhancing FMs.
- We propose two novel approaches, MixFM and SMFM, to boost FMs by generating auxiliary data. They do not need additional knowledge and can establish direct interaction with original non-interactive features. Besides, we contribute an efficient solution for calculating the sample’s saliency, guiding SMFM to generate more informative data.
- We deliver the generalization bound of MixFM, which incorporates auxiliary data, and theoretically prove that our methods have a better generalization guarantee than FM.
- Through extensive experiments with seven real-world datasets, we empirically demonstrate the superiority of MixFM and SMFM in improving performance. Notably, the results indicate that our methods are also effective in FM variants.

The rest of this paper is organized as follows. Section II introduces the related work. Section III introduces the necessary preliminaries. Section IV introduces two novel approaches, MixFM and SMFM, to boost FMs by generating auxiliary data. Section V delivers theoretical analyses of the proposed method for improving FMs. Section VI presents extensive experimental comparisons and analyses. The final section concludes the paper and looks forward to future research.

II. RELATED WORK

Given that traditional linear models deal with features independently and ignore their interactions, Rendle et al. [5] proposed factorization machines (FMs), which use a simple dot product to achieve arbitrary feature interaction. Such flexibility makes it widely used in various practical scenarios [3], [7], [16], [17], [18]. However, these specific features require expert knowledge support and expensive labor, which limit their applicability. This encourages us to investigate more general augmentation algorithms.

Manual Architecture Design: In recent years, much work has been devoted to the design of FM structures. HOFM [19] carried out third-order or higher-order extensions of feature interaction. Consequently, Field-Aware Interaction Layer [20], tensor product spline model [21] and Dual Attention Mechanism [22] were proposed to capture high-order feature interactions. Considering that FM treats the contribution of each pair feature interactions equally and ignores its differences, Juan et al. [11] proposed Field-aware factorization machines (FFM). They introduced the concept of field and learned its embedding representation for each feature in each field. Recently, the Field-weighted Factorization Machine (FwFM) [23], [24] modeled the field interaction intensity more parameter-efficiently. The irrational assumption that traditional FMs regard the importance of high-order terms indifferently has prompted extensive research on Attentional Factorization Machines [22], [25], [26], as they can mitigate the impact of unimportant feature interactions. Inspired by adversarial training, Punjabi et al. [27] proposed a robust factorization machine by adding perturbations to the first-order and second-order terms during training. Recently, multi-view machines [13], [28], [29] that model high-order feature interactions based on multi-views have begun to receive attention, and they can design feature interactions from different perspectives. Benefiting from the potential of deep neural networks (DNN) in feature representation, some work [4], [9], [30], [31], [32] utilized them to model high-order feature interactions, including DeepFM [31], NFM [9], Deep & Cross network (DCN) [32], and XDeepFM [4]. Although high-order interactions enhance the model representation, directly learning unobserved feature interactions is still difficult.

Automated Architecture Search: Admittedly, these methods require considerable human intervention, such as model selection, parameter tuning, etc., making them challenging to generalize to new environments. Therefore, Automated Machine Learning (AutoML) [33], which primarily focuses on embedding size search, has shown potential on FM and received increasing attention. Given that different features may require varying embedding sizes, PEP [34] pruned embeddings by dynamically learning thresholds, while AutoDim [35] utilized DARTS [36] to design multiple optional embedding tables for each field for network search. AutoSrh [37] relaxed the discrete dimension search to continuous and optimized using gradient descent. SSEDs [38] introduced a criterion that can identify the importance of each embedding with just one propagation. Additionally, adaptive denoising instance selection [39] and neural architecture search [40] have also been shown to enhance...
FM). However, neural network searches typically employ many different models and parameters to fit datasets, thereby increasing the risk of overfitting, which may lead to poor performance on unobserved feature interactions.

**Feature Selection:** These methods often incorporate all features into the model learning, disregarding the potential negative impact of redundant features. As a result, feature selection has gained increasing attention. AutoField [41] adopted a two-stage design, where a differentiable controller network searches for the required fields, and the recommended model is retrained based on the selected fields. In contrast, AdaFS [42] employed a more flexible approach, adaptively jointly optimizing the feature selection and model training. BH-FIS [43] used Bayesian variable selection to choose feature interactions and decompose them into independent variables, ensuring the learning of both interaction selection and interaction weight independently. OptFS [44] focused on selecting feature interactions and transformed it into selecting two relevant features to accommodate various interaction operations while incorporating a learnable gate to determine whether a feature should appear in the feature set. Recently, graph-based feature crossing has gained increasing attention, where features are treated as nodes in a graph, and the interaction relationships between features are automatically learned through gradient descent, which infers the graph structure. GCFM [45] employed a multi-filter graph-convolved feature crossing layer to establish cross features with neighboring embeddings, expanding feature interactions from pairwise to proximity-based to capture more comprehensive information. GraphFM [46] utilized a multi-head attention mechanism to accommodate the polysemy of feature interactions in different semantic spaces. DAGFM [47] asserted that the graph information propagation process corresponds strictly to a dynamic programming algorithm for constructing arbitrary-order interactions, enabling graph-based FM to achieve lossless compression for arbitrarily complex feature interaction models. However, feature selection strategies focus on observable feature interactions in the training set and still cannot address the issue of unobserved feature interactions.

### III. PRELIMINARIES

This section will give the necessary preliminaries to facilitate a clear introduction to the proposed methods.

#### A. Factorization Machines

In the practical recommendation, data can be described by categorical and numerical features. Below we give an example.

| Categorical Features | Numerical Features |
|---------------------|-------------------|
| 1, 0, ..., 0       | [0, 1, ..., 0]    |
| User ID            | Item ID           |
| 0.1, 0.29, ..., 0.01 | Historical Items |

For category features, User ID and Item ID use one-hot encoding, while the historical item list employs multi-hot encoding. Numerical features are normalized to [0,1]. Clearly, even if there are only three types of categorical features in this example, the feature vector will become highly sparse after encoding.

For this reason, Rendle et al. [5] proposed factorization machines for prediction tasks of extremely sparse data. Its key idea is to consider the interrelationships between sparse features and combine them. In the paper, we only study the 2-way interactive FM. Specifically, assuming a real-valued feature vector $x \in \mathbb{R}^m$ (most values are 0), the learning paradigm of FM can be defined as

$$f(x) = w_0 + \sum_{i=1}^{m} w_i x_i + \sum_{i=1}^{m} \sum_{j=i+1}^{m} \langle v_i, v_j \rangle x_i x_j,$$

(1)

where $m$ is the number of features after encoding, $w_0$, $w_i$ are the weights of the bias term and the linear term, respectively, $v \in \mathbb{R}^{m \times d}$, and $v_i \in \mathbb{R}^d$ represents the embedding vector of the $i$th feature $x_i$, where $d$ is the embedding size. The inner product $\langle v_i, v_j \rangle$ parameterizes the interaction between $x_i$ and $x_j$. It is worth noting that the computational complexity of (1) is $O(d_m^2)$. To efficiently work, the second-order term is usually reformulated as

$$\sum_{i=1}^{m} \sum_{j=i+1}^{m} \langle v_i, v_j \rangle x_i x_j = \frac{1}{2} \sum_{k=1}^{d} \left( \sum_{i=1}^{d} v_{i,k} x_i \right)^2 - \sum_{i=1}^{d} v_{i,k}^2 x_i^2.$$

(2)

The complexity is reduced to $O(dm)$.

#### B. Saliency Map

With the rapid development of machine learning today, even in the face of a model with good performance, we will still be wary as the model is not interpretable, and we fail to know the working principle. Studying the interpretability may help clarify the model’s internal mechanisms and working principles [48], [49], [50]. One of the most widely studied tools is the saliency map [51], which generates a saliency vector with the same size as input to determine which features dominate the decision. For example, for an image $x$ in the field of computer vision, $f(x)$ represents the prediction of $x$ by the model $f$, then the saliency map $S$ of any image $x_0$ can be calculated by

$$S(x_0, f) = \frac{\partial f(x)}{\partial x} \bigg|_{x_0},$$

where the ith value of $S(x_0, f)$ represents the saliency of the ith pixel of $x_0$.

### IV. METHODOLOGY

This section will detail the proposed boosting strategies: MixFM and its improved version SMFM.

#### A. MixFM

First, we present a domain-knowledge-independent method, namely Mixup powered Factorization Machine (MixFM), for enhancing the FM. It is mainly based on the following considerations: (1) learning extremely sparse data will be highly dependent on (overfit or underfit) the dataset [9]; (2) existing information assistance methods require expensive workforce collection or assurance of information availability, which may limit the practicality; (3) FM and its variants cannot directly
establish the explicit relationship between the non-interactive features that are not observed or even absent, which limits their performance to generalize to unseen data.

The first two considerations are intuitive, and we verify the benefits of constructing such explicit interactions between the non-interactive features. Specifically, in the Frappe dataset, we remove samples that simultaneously contain features “5” and “20” from the training set. "w/o interaction" denotes the test set will not contain these two features at the same time, and "w interaction" is the opposite.

like traditional FM, which performs empirical risk minimization (ERM) on the original dataset, MixFM optimizes based on the principle of vicinal risk minimization (VRM). Specifically, it is trained on both the standard dataset and the neighbors of natural samples. Suppose \( D \) is the standard dataset. For any sample pair \((x, y) \sim D\), its neighbor data \((\tilde{x}, \tilde{y})\) is defined as follows:

\[
\tilde{x} = \lambda \cdot x + (1 - \lambda) \cdot r_x, \\
\tilde{y} = \lambda \cdot y + (1 - \lambda) \cdot r_y,
\]

where \( \lambda = \max(\lambda', 1 - \lambda') \), and \( \lambda' \sim \text{Beta}(\alpha, \beta) \). \( \alpha \) and \( \beta \) are controllable parameters for Beta distribution, and \((r_x, r_y)\) is a data pair sampled from the dataset \( D \). Unlike Mixup, we limit \( \lambda \geq 0.5 \) (indicated by \( \lambda = \max(\lambda', 1 - \lambda') \)). In this way, we can customize the first parent sample \((x, y)\) to make the nearest neighbor of the generated data controllable, thereby avoiding the generated samples being biased towards specific ones and increasing the data diversity. Notably, although our strategy and Mixup all adopt a similar linear interpolation strategy to generate new samples, the sparsity of recommendation data makes us pay more attention to enriching interactive features than just data.

The left part of Fig. 1 illustrates the process of generating new samples in MixFM. As can be seen, the generated mixed data inherits the features and behaviors of the two parents: (1) If the parent instances have the same behavior (feedback or not), then the mixed samples inherit the features of both and exhibit the same behavior, e.g., \( \tilde{x}^3 \) and \( \tilde{x}^2 \) in Fig. 1. (2) If the parents show different behaviors, then the mixed sample makes a compromise, e.g., \( \tilde{x}^3 \) in Fig. 1. Intuitively, such compromise of reducing weights avoids overconfident decisions for uncertain feature interactions.

Definition 4.1 (MixFM): Suppose the original dataset is \( D = \{(x^i, y^i)\}_{i=1}^n \) of size \( n \), \( n' \) mixed samples \( \tilde{D} = \{ (\tilde{x}^i, \tilde{y}^i) \}_{i=1}^{n'} \) are generated according to (3). Let \( L(f(x), y) \) be the training loss with respect to \((x, y)\), where \( f \) represents the standard FM, then MixFM performs empirical vicinal risk minimization on \( D \cup \tilde{D} \)

\[
\arg\min_f \frac{1}{n + n'} \sum_{(x, y) \in D \cup \tilde{D}} L(f(x), y). \tag{4}
\]

Definition 4.1 gives the formal definition of MixFM. The detailed process of MixFM is shown in Algorithm 1. To make the mixed samples cover the entire data space as possible (become all users’ neighbors), we perform sampling without replacing the first parent sample (Line 3 in Algorithm 1). For the special case where the number of generated samples exceeds the dataset size (i.e., \( n' > n \)), \( (n' \mod n) \) data are sampled without replacement, and copy the entire dataset \( \lfloor n'/n \rfloor \) times, where \( \lfloor \cdot \rfloor \) is rounded down.

Relationship With Noise Learning: On the one hand, it is undeniable that the introduced augmented data is indeed inconsistent with the original data from the data content perspective (adding a maximum of \( n' \) features and \( 1/2 \times n' \times (3n' - 1) \)

Fig. 1.Overview of our methods. Left part: An example of mixed data in FM, where \( \{x^1, \ldots, x^4\} \) represent the natural samples, and \( \{\tilde{x}^1, \tilde{x}^2, \tilde{x}^3\} \) are the mixed samples. Right part: the framework of SMFM.

Fig. 2. Test curves (AUC and LogLoss on Frappe) for 50 training epochs. We removed all samples containing the interaction of features “5” and “20” from the training set. “w/o interaction” denotes the test set will not contain these two features at the same time, and “w interaction” is the opposite.
second-order feature interactions, where \( n' \) is the number of non-zero features in origin data. Therefore, they can be considered as “noise”. On the other hand, we cannot deny that the “noise” originates from the introduction of additional interactive features, which allows FM to learn a broader range of feature interactions and achieve more effective adaptation to unknown data, enhancing its generalization capabilities. In summary, from the perspective of data content, the model benefits from noise learning. Taking a deeper look at the form of the noise, we can also consider that the model benefits from the introduced interactive features. They just look at the same viewpoint from different perspectives.

Analysis of MixFM: Corresponding to the motivations at the beginning of this section, we can reconsider MixFM from the following four aspects:

- **Overfitting Issue:** MixFM focuses on augmenting the data residing on the line connecting the raw data. This means that it trains the model to minimize the vicinal risk of the original data rather than directly minimize their empirical risks, which may alleviate the model’s high dependence on raw data.

- **Additional Knowledge:** According to (3), MixFM generates augmented data by performing linear interpolation between any two raw samples, which does not require any additional information (e.g., fields [11] and user mood [5]) beyond the basic interactive features.

- **Explicit Feature Interactions:** For any non-interactive feature, as long as they respectively exist in the parent samples, the mixed data can establish their direct interaction by inheriting all parent samples’ features. For instance, in Fig. 1, the third feature of user id and the first component of item id have no intersection in natural data \( \{x^i\}_{i=1}^4 \), while the mixed sample \( \hat{x}^3 \) establishes their interaction. More specifically to our experiments, MixFM’s 50 epochs of training can learn 99.22% of the interactive features of the test set in Frappe (75.16% in FM); that is, MixFM greatly reduces the unknowable interactive features in the test set from 24.84% to 0.78%. Intuitively, the extensive learning of feature interactions could improve the model’s performance.

- **Scalability:** We only need to mix non-zero features for the sparse data, which is very small, even for large-scale datasets. This allows the generation time of each instance to be linearly dependent on the number of non-zero features, and the complexity is approximately \( O(1) \), which makes it very scalable. We will also experimentally verify it regarding computation time (Section VI-K) and performance gains (Sections VI-B and VI-C).

### B. SMFM

The randomness of the samples generated by MixFM makes it inevitable to produce some redundant or even detrimental samples. For instance, if the \( \lambda \) is infinitely close to 1, MixFM is similar to copying raw samples, which only increases the training cost under sufficient training. Therefore, it is vital to generate more meaningful samples. Based on MixFM, we further propose a Saliency-guided Mixup powered Factorization Machine (SMFM).

For a well-trained model, if adding a mild noise to a sample can cause a significant change in the decision, then the sample is likely to be in the blind zone of the model. Given that MixFM tries to model the discrete data space in a continuous manner, the samples corresponding to such blind domain are likely to be informative samples, and incorporating them into the dataset may be beneficial for training. Formally, suppose \( \eta \) is the noise, where \( ||\eta|| < \epsilon \). Then, the change of loss \( \mathcal{L}(f(x), y) \) by adding noise \( \eta \) to \( x \) can be described as

\[
\Delta \mathcal{L}(f(x), y) \approx \frac{\partial \mathcal{L}(f(x), y)}{\partial x} \cdot \eta,
\]

where \( \Delta \mathcal{L}(f(x), y) = \mathcal{L}(f(x + \eta), y) - \mathcal{L}(f(x), y) \). The \( i \)-th component in \( \frac{\partial \mathcal{L}(f(x), y)}{\partial x} \) measures the saliency of the \( i \)-th feature of \( x \) in the model decision. Unlike the saliency map described in Section III-B, we measure the significance of the loss rather than that of the prediction score. In addition, considering that the smaller the feature in FM, the lower its contribution to the prediction (there is no contribution when the feature is 0). Therefore, we define the weighted saliency of any sample \( x \) in the paper

\[
\text{Saliency}(x, y, f) = \left( \frac{\partial \mathcal{L}(f(x), y)}{\partial x} \right)^T \cdot x,
\]

This is easy to implement in automatic differentiation systems like PyTorch [52] and TensorFlow [53].

With the means of calculating the sample’s saliency, we can use it to guide the generator to explore more significant neighbor data, which is the core idea of SMFM. Specifically, in SMFM, we first generate \( p \) candidate neighbors for each selected sample by (3). Second, according to (5), the most salient neighbor is selected from the \( p \) neighbors as the final mixed sample. The framework of SMFM is illustrated in the right part of Fig. 1, and the detailed algorithm is shown in Algorithm 2. Clearly, SMFM builds upon MixFM by introducing the operation of saliency calculation and data greedy selection, which also does not involve any additional knowledge beyond the basic interactive features.

It is worth noting that the recent work [54] proposed Maxup, but the proposed SMFM is quite different from them. On the
Algorithm 2: SMFM.

**Input:** The dataset \( \mathcal{D} = \{(x^i,y^i)\}_{i=1}^n \) with size \( n \); training epochs \( T \); the size of mixed data \( n' \); the number of candidate neighbors per sample \( p \).

1: for \( t = 1 \ldots T \) do
2: \( \mathcal{D} = \emptyset \).
3: Select \( n' \) instances \( \hat{\mathcal{D}} \) from \( \mathcal{D} \) without replacement.
4: for each \((x,y)\) in \( \mathcal{D} \) do
5: Generate \( p \) neighbors of \((x,y)\) through (3), denote them as \( \hat{\mathcal{D}}(x,y) \).
6: \( (\tilde{x}, \tilde{y}) = \arg \max_{(x',y') \in \hat{\mathcal{D}}(x,y)} \text{Saliency}(x',y',f) \).
7: \( \hat{\mathcal{D}} = \hat{\mathcal{D}} \cup \{ (\tilde{x}, \tilde{y}) \} \).
8: end for
9: Do standard training on \( \mathcal{D} \cup \hat{\mathcal{D}} \) according to (4).
10: end for
11: return \( \mathcal{D}' \)

On one hand, the mixed data in Maxup is selected based on the loss, whereas we utilize the saliency of the sample. On the other hand, Maxup modifies the natural training data; in contrast, we “poison” new data without changing the original ones. We also perform detailed comparisons in the experiments.

**Analysis of Saliency-Guided Mixup:** We can understand the Saliency-guided Mixup from two aspects:

- **Feature Section:** In real-world datasets, there are often numerous features that may contain redundant or irrelevant information. It may lead to suboptimal recommendation performance and slower model optimization [41], [42]. Therefore, feature selection plays a crucial role in recommender systems. Considering that the saliency map could highlight the most crucial and influential parts of the input data (i.e., features in sparse data) [55], the proposed Saliency-guided Mixup allows us to greedily select feature interactions that the model deems important.

- **Adversarial Training:** The saliency map emphasizes which features the model focuses on. This knowledge is beneficial for crafting effective adversarial examples, as proposed in [56], namely the adversarial saliency map. Therefore, in each training iteration, constructing saliency-guided mixed samples can be seen as generating adversarial examples (maximization). Then, the model is trained based on these saliency-guided examples (minimization). This min-max training strategy is essentially the learning paradigm for adversarial training. Indeed, adversarial training has been widely confirmed to enhance models’ generalization capabilities, enabling them to perform better on unseen data [57], [58].

V. THEORETICAL RESULTS

This section theoretically analyzes the positive role of our methods in improving the generalization of FM. For the output hypothesis \( f \) of the learning algorithm, the generalization error \( \hat{R}_\mathcal{D}(f) \) we focus on is the predictive ability on unseen data, which cannot be directly observed, but its empirical error \( \mathcal{R}(f) \) on the training set can be directly observed. To this end, we explore the generalization error upper bound concerning empirical error, which gives the worst generalization performance.

Notably, although related work [59] has delivered the generalization bound of Mixup, it is limited to the typical linear model with input constraints and does not apply to FM with second-order or higher interactions. Nevertheless, we can benefit from their work by ignoring the linear term \( \sum_{i=1}^m w_i x_i \) in FM for simplification. All proofs can be found in Supplementary Material.

A. Generalization Bound for Factorization Machines

We give the necessary notations that will be used in proofs. \( R(x,k) \) denotes repeating \( x \) \( k \) times (e.g., \([R(3,2), R(2,1)] = [3,3,2] \)). Besides, we define \( V_{i=1}^m x_i = x_1, \ldots, x_n \) (e.g., \([V_{i=1}^3 x_i = [x_1, x_2, x_3] \)). Then, we can reformulate FM as follows.

**Definition 5.1 (Linear representation of FM):** For an FM that omits the first-order term, define a feature vector \( u = \sum_{i=1}^{m} V_{i=1}^{d-2} R(x_i, x_j, d), R(0, d^2 m - dm(m - 1)/2)^T \in \mathcal{R}^{d^2 m^2}, \theta = [V_{i=1}^{m} V_{i=1}^{j=1} V_{k=1}^{m} V_{j=1}^{d=1} V_{k=1}^{d=1} V_{j=1}^{m} V_{k=1}^{d=1} V_{j=1}^{m} V_{k=1}^{d=1} V_{j=1}^{m} V_{k=1}^{d=1} V_{j=1}^{m} V_{k=1}^{d=1} V_{j=1}^{m} V_{k=1}^{d=1} V_{j=1}^{m} V_{k=1}^{d=1} V_{j=1}^{m} V_{k=1}^{d=1} V_{j=1}^{m} V_{k=1}^{d=1} V_{j=1}^{m} V_{k=1}^{d=1}]^T \in \mathcal{R}^{d^2 m^2} \), where \( x \) is consistent with the expression in Section III-A.

Then, FM can be redefined as

\[
\hat{f}(u) = \theta^T u + w_0. \tag{6}
\]

Comparing (6) with (2), it is easy to find that they are equivalent. Following [60], we study the following function class \( \mathcal{H}_\gamma \) closely related to FM:

\[
\mathcal{H}_\gamma := \left\{ f : x \mapsto f(x), \sum_{i=1}^{m} \sum_{j=1}^{d} v_{i,j}^2 \leq \gamma \right\}. \tag{7}
\]

By Definition 5.1 and \( (\sum_{i=1}^{m} \sum_{j=1}^{d} v_{i,j}^2) = ||\theta||^2 \), this is equivalent to

\[
\mathcal{H}_\gamma := \{ f : u \mapsto \theta^T u + w_0, ||\theta||^2 \leq \gamma \}. \tag{8}
\]

In classical learning theory, the model generalization is closely related to the function complexity, and the typical complexity measurements are VC-Dimension [61], Covering number [62], and Rademacher complexity [63]. We adopt Rademacher complexity to analyze the complexity of FM in (8).

**Theorem 5.2.** For an FM that omits the first-order term, suppose its hypothesis space is \( \mathcal{H}_\gamma \) and \( \mathcal{D} = \{(x^i, y^i)\}_{i=1}^n \) is the dataset of size \( n \), where \( ||x||_\infty \leq 1 \), and \( ||x||_0 \leq \tau \). Then the empirical Rademacher complexity \( \hat{R}_\mathcal{D}(\mathcal{H}_\gamma) \) satisfies

\[
\hat{R}_\mathcal{D}(\mathcal{H}_\gamma) \leq \sqrt{\frac{\gamma^2 d\tau(\tau - 1)}{2n}}.
\]

This theorem gives the empirical Rademacher complexity of FM. Furthermore, we derive the generalization bound of FM as follows.

**Corollary 5.3:** Suppose that for the function \( f \) in \( \mathcal{H}_\gamma \), \( f \) maps any sample to \([0,1]\). Sample \( x \) satisfies \( ||x||_\infty \leq 1 \), and \( ||x||_0 \leq \tau \). Then, the generalization bound of FM is as follows:

\[
\mathcal{R}(\mathcal{H}_\gamma) \leq \sqrt{\frac{\gamma^2 d\tau(\tau - 1)}{2n}}.
\]
\[ R_D(f) \leq \hat{R}_D(f) + 2\sqrt{\frac{\gamma^2 d(\tau - 1)}{2n}} + 3\sqrt{\frac{\ln(2/\delta)}{2n}}, \]  
where \( R_D(f) \) and \( \hat{R}_D(f) \) represent the generalization error and empirical error of the model, respectively.

**Remark 1:** Eq. (9) indicates the FM’s generalization error is related to the \( O\left(\frac{\sqrt{d}}{n}\right) \) term. On the one hand, a smaller embedding size \( d \), more data \( n \) will benefit model generalization. On the other hand, the generalization error is related to the empirical error \( \hat{R}_D(f) \). A smaller \( d \) reduces model complexity, which in turn may increase empirical error. In summary, the model’s generalization needs more data, but the embedding size \( d \) requires a compromise between model complexity and empirical error. Significantly, this is the first generalization bound of FM, lacking in the previous work.

**B. Generalization Bound for MixFM**

After giving the generalization bound of the standard FM, we will prove that MixFM has a smaller generalization error. In MixFM, the operation of injecting mixed samples \( \hat{D} \) into the dataset can be decomposed into two stages: (1) inject all first parent samples of \( \hat{D} \) and define them as \( \hat{D} \); (2) Modify \( \hat{D} \) to its neighbor data \( D \) by (3). From the mathematical expectation, stage 1 increases the training epochs, so we have \( R_{\hat{D},\hat{D}}(f) = R_D(f) \) under sufficient training (it will also be evidenced in the experiments). Therefore, we only need to analyze stage 2: that is, as long as performing Mixup on the original dataset provably reduces the generalization error, it will be concluded that the effectiveness of MixFM in boosting generalization. In the following analysis, we adopt logistic loss for the classification tasks. Note that the following analysis is also applicable to regression tasks and even has a more concise form. We first provide a related finding to simplify the form of Mixup.

**Lemma 5.4 ([59]):** Suppose \( D = \{(x^i, y^i)\}_{i=1}^n \) is the dataset of size \( n \). Let \( \hat{D} = \{\hat{x}^i, \hat{y}^i\}_{i=1}^m \), where \( \hat{x}^i = \lambda x^i + (1 - \lambda) r_x \), \( \hat{y}^i = \lambda y^i + (1 - \lambda) y_g \), \( \lambda \sim \text{Beta}(\alpha, \beta) \), and \( (r_x, y_g) \sim D \). If the loss function has the form similar to \( l(x, y) = h(f(x)) - y f(x) \), where \( f(x) \) is the predict of \( x \), and \( h(\cdot) \) is an arbitrary function, then

\[
E_{\lambda, \text{Beta}(\alpha, \beta)} f(\hat{x}^i, \hat{y}^i) = E_{\lambda, \text{Beta}(\alpha + 1, \beta)} f(x^i, y^i) + \frac{\beta}{\alpha + \beta} \text{Beta}(\beta + 1, \alpha) f(x^i, y^i). \tag{10}
\]

Given the difficulty of analyzing the Mixup in \( x \) and \( y \) simultaneously, Lemma 5.4 points out that if the loss function is \( l(x, y) = h(f(x)) - y f(x) \), then performing Mixup on the data pair \( (x, y) \) can be simplified just to perform Mixup on the \( x \) but \( \lambda \) follows \( \text{Beta}(\alpha + 1, \beta) + \frac{\beta}{\alpha + \beta} \text{Beta}(\beta + 1, \alpha) \) instead of \( \text{Beta}(\alpha, \beta) \).

For the logistic loss \( l(x, y) = \log(1 + \exp(f(x))) - y f(x) \) that we focus on, let \( h(x) = \log(1 + \exp(f(x))) \), then we can directly apply Lemma 5.4, and only need to consider the Mixup on \( x \) by setting \( \lambda \sim \frac{\alpha}{\alpha + \beta} \text{Beta}(\alpha + 1, \beta) + \frac{\beta}{\alpha + \beta} \text{Beta}(\beta + 1, \alpha) \).

For simplicity, we define it as \( \tilde{\lambda} \). Similar to the analysis of Section V-A, we next focus on the linear form of FM in Definition 5.1; that is, \( f(u) = u^T u + w \).

Since \( \hat{u} = [\{m^{-1} \sum_{j=1}^m R(x, x_j, d), R(0, d^2 m^2 - dm(m - 1)/2)\}]^T \), according to (3), we have the mixed data as follows:

\[
\hat{u} = \begin{bmatrix} m^{-1} \sum_{i=1}^j R(x_i x_j, d), R(0, d^2 m^2 - dm(m - 1)/2) \end{bmatrix}^T.
\]

Considering each term \( x_i x_j \) in \( \hat{u} \), we have \( \hat{x}_i \hat{x}_j = \lambda^2 x_i x_j + \tilde{\lambda}(1 - \tilde{\lambda}) x_i (x_j + r_x) + (1 - \tilde{\lambda})^2 (r_x) i (r_x) j \).

Notice that when we centralize data \( E_{\hat{u}_{\sim D}(x_i x_j)} = 0 \), so \( \hat{u} = \tilde{\lambda}^2 u + (1 - \tilde{\lambda})^2 r_u \),

where \( r_u = \begin{bmatrix} m^{-1} \sum_{i=1}^j R((x_i x_j), d), R(0, d^2 m^2 - dm m - 1)/2 - 2 \end{bmatrix}^T \).

Therefore, performing traditional Mixup on \( x \) can be regarded as performing a mixture on \( u \) to get \( \hat{u} \), and the mixing rule is \( \hat{u} = \tilde{\lambda}^2 u + (1 - \tilde{\lambda})^2 r_u \).

Since the model \( f(\hat{u}) = u^T \hat{u} + w_0 \), the prediction is invariant to the shifting and scaling of \( \hat{u} \), so we study the generalization on the model on the data \( u = \frac{1}{\tilde{\lambda}} [\hat{\lambda}^2 u + (1 - \tilde{\lambda})^2 r_u] \).

By these transformations, we have the following result.

**Theorem 5.5:** For an FM that omits the first-order term, \( \hat{D} = \{(\hat{x}^i, \hat{y}^i)\}_{i=1}^m \) is the dataset of size \( n \), and \( D = \{\hat{x}^i, \hat{y}^i\}_{i=1}^m \) is the dataset after Mixup, where \( (\hat{x}^i, \hat{y}^i) \) is calculated by (3). Besides, \( u \) and \( \theta \) are defined in Definition 5.1. Let \( \mathcal{L} \) be the logistic loss, then the effect of Mixup in FM can be regarded as a data-related regularized term, that is,

\[
\mathcal{L}(\hat{D}) = \mathcal{L}(D) + \frac{1}{n} \sum_{i=1}^n g(u^i)(1 - g(f(u^i))) E_{\hat{u}_x} \left( \frac{1 - \tilde{\lambda}}{\tilde{\lambda}^4} \theta^T \hat{u}_x \right) \tag{11}
\]

where \( g(\cdot) \) is a sigmoid function, \( f(u^i) = u^T u^i + w \), \( \hat{u}_x = \frac{1}{n} \sum_{i=1}^n u^i u^i T \), where \( u^i \) is transformed from \( x^i \) by Definition 5.1. \( \lambda \sim \frac{\alpha}{\alpha + \beta} \text{Beta}(\alpha + 1, \beta) + \frac{\beta}{\alpha + \beta} \text{Beta}(\beta + 1, \alpha) \).

Theorem 5.5 suggests that using Mixup on the original dataset is equivalent to adding a regularization term for standard training. Adhering to the similar approach in [5], we consider the following function class closely related to the dual problem of Theorem 5.5

\[
\mathcal{H}_\gamma := \{f : u \mapsto \theta^T u + w_0, E_0 g(f(u)) \}
\]

\[
(1 - g(f(u))) \theta^T \hat{u}_x \theta \leq \gamma), \tag{12}
\]

where \( f(u) = \theta^T u + w_0, \hat{u}_x = E_0 u^i u^i T \). Next, we provide the generalization error bound of FM trained by mixed data.

**Theorem 5.6:** Suppose that for the function \( f \) in \( \mathcal{H}_\gamma \), \( f \) maps any sample to \([0,1]\). the sample \( x \) satisfies \( \|x\|_\infty \leq 1 \), and
\[ \|x\|_0 \leq \tau. \] Then, for any \( \delta > 0 \), with probability at least \( 1 - \delta \) over the draw of an i.i.d. dataset \( D \) of size \( n \), the following holds for all \( f \in \mathcal{H}_\gamma \):

\[ \tilde{R}_D(f) \leq \tilde{R}_D(f) + 2 \sqrt{\frac{(1+\epsilon)^2 \gamma \cdot \tau (\tau-1)}{2en}} + 3 \sqrt{\frac{\ln(2/\delta)}{2n}}, \]

where \( \tilde{R}_D(f) \) and \( \tilde{R}_D(f) \) represent the generalization error and empirical error, respectively.

**Remark 2:** Comparing Corollary 5.3 and Theorem 5.6, it is easy to conclude that when \( \gamma \geq \frac{(1+\epsilon)^2}{\epsilon d} \approx \frac{5.6}{d} \), MixFM has a smaller generalization error bound. In the worst case, \( \gamma \geq 2.54 \) when \( d = 2 \), which is still easily met because \( \gamma \) is often large to enhance the expressive capability of the model. In our experiments, \( \gamma \) takes the minimum value of 106.1 in the Frappe dataset, while in larger datasets, they are much larger than this value. Therefore, the expected risk of MixFM is closer to the empirical prediction, and the generalization ability is stronger [64].

**Overfitting and Underfitting Considerations:** If we shift the empirical error \( \tilde{R}_D(f) \) to the left of the inequality, Corollary 5.3 and Theorem 5.6 serve as the upper bound to measure model overfitting. Furthermore, the overfitting bound inspired us that more data \( n \), smaller \( \gamma \), and smaller embedding size \( d \) are needed to alleviate overfitting risk. Regarding model underfitting, the generalization implies that an FM with a small generalization error tends to be accompanied by small empirical risks; that is, there is usually no underfitting issue. Undeniably, for underfitting, the generalization theory cannot provide improvement suggestions refined to each parameter similar to overfitting. This is because generalization explores the predictive ability of unknown data while underfitting focuses on the fitting ability of training data.

As we discussed in Section IV-B, saliency data can be regarded as adversarial examples. Therefore, SMFM, which creates saliency neighbor samples for training, essentially performs adversarial training on MixFM. Recent studies [57], [58] have proved that adversarial training helps to improve generalization, so the generalization of SMFM is not lower than MixFM, let alone FM. Thus far, we have proven the effectiveness of the proposed methods in enhancing FM’s generalization.

### VI. EXPERIMENTS

In this section, we first introduce the experimental settings, such as datasets, baselines, and parameters, in Section VI-A. Then, we conduct extensive experiments concerning generalization, robustness, and sensitivity to the parameters of the proposed algorithm, see Sections VI-B to VI-I.

#### A. Experiments Settings

1) **Datasets:** We use seven real-world datasets to validate the effectiveness of our methods. Table I lists the detailed statistics.

- **Frappe** [65]: This is a context-aware mobile app dataset containing 288,609 records of 957 users on 4,082 apps. Each record includes contextual information such as weather, time, and city. We use one-hot encoding and multi-hot encoding to encode each record into a 5,382-dimensional vector.
- **MovieLens** [66]: This is a movie rating dataset, which includes 2,006,859 ratings of 23,743 movies by 17,045 users, and 668,953 movie tags are assigned to these movies. One-hot encoding is adopted to decompose each rating into 90,445 features.
- **Pet Supplies (Pet), Electronics, Books:** They are composed of the review information collected by Amazon customers in three different domains, namely pet supplies, electronic products, and books. These datasets include 11,400,788 users’ reviews on 103,287 pet items, 476,000 electronics, and 2,330,065 books. Each review has the user ID, produce ID, user status, brand, and other information.
- **Avazu:** It collects 11 consecutive days of user interaction data on mobile ads, including 40,428,968 user interaction records, 22 feature fields, and 2,018,012 features.
- **Criteo:** This is an industrial dataset that collects users’ ad clicks over one week. It comprises 45,840,617 user click records, 39 feature fields, and 1,086,810 features.

Since these datasets only contain positive instances, similar to [9], we randomly sample two negative instances paired with one positive instance. The augmented dataset is randomly divided into the training set, validation set, and test set with a ratio of 7:2:1.

2) **Comparison Methods:** The proposed strategies are compared to the following state-of-the-art feature interaction models and non-FM models.

- **FM** [5]: This is the benchmark model, factorization machines, using first-order feature information and pairwise feature interactions for prediction tasks.
- **NFM** [9]: NFM pioneers incorporating FM into a neural network and utilizes a neural network to replace the second-order feature interactions to model richer interactions. Specifically, it first used a Bi-Interaction layer to encode all the feature interactions in the embedding space without introducing additional model parameters. Next, it stacked multiple non-linear layers on top of the Bi-Interaction layer to capture higher-order feature interactions.
- **AFM** [25]: FM assumes equal weights for all second-order feature interactions, but some redundant feature interactions...
may negatively impact the model. To this end, AFM’s core idea is to differentiate the importance of different interaction features. Specifically, it employs an MLP and a softmax function to parameterize the attention factor $a_{ij}$ between each interaction feature $i$ and $j$. Subsequently, AFM corrects the second-order interaction term in FM $(v_i \odot v_j)x_ix_j$ to $a_{ij}(v_i \odot v_j)x_ix_j$.

**DCN** [32]: Its core idea lies in the two networks: (1) The Cross network. It consists of cross layers, where the $l$th layer is formulated as $x_{t+1} = x_t x_l^T w_l + b_l + x_t$. This structure increases the degree of feature interaction with the number of network layers. (2) The Deep network is a fully-connected feed-forward neural network, represented as $h_{l+1} = f(W_l h_l + b_l)$, which is capable of capturing non-linear feature interactions.

**DAFM** [22]: It introduces a dual-attentional mechanism: (1) Attentional Interaction. It incorporates a contextual attention mechanism in each MLP layer. This allows DAFM to capture the contributions of different high-order feature interactions. (2) Attentional Pooling. The output of the attentional interaction layer is then passed through an attentional pooling layer to ensure that the current high-order feature interactions include the lower-order terms.

**FINT** [20]: It proposes a field-aware interaction layer to capture the semantic information of high-order feature interactions while preserving the lower-order ones. Specifically, FINT maintains a feature interaction vector for each field-aware interaction layer, which includes interaction information of the field and each other field within the k-order. Finally, FINT employs a DNN to concatenate all the field embeddings and applies a non-linear transformation to further explore the nonlinear high-order relationships between fields.

**DeepFM** [31]: It improves the Wide&Deep [67] by replacing the Logistic Regression part with traditional FM to enhance the information extraction capability on the Wide side. Additionally, the Deep network employs an MLP to capture non-linear interactions and higher-order features.

**CopyFM, CopyNFM, CopyAFM, CopyDCN**: They use the same FM-based models (FM, NFM, AFM, and DCN) but randomly double some raw samples. Since our strategies enrich the dataset, they are introduced to make a fair comparison.

**MixNFM, SMNFM, MixAFM, SMAFM, MixDCN, SMDCN**: It can be seen from Algorithms 1 and 2 that the two proposed ideas are decoupled from factorization machines. Therefore, applying strategies similar to MixFM and SMFM, we introduce these models to validate the effectiveness of our ideas on FM variants.

**MaxFM, MaxNFM**: They respectively represent that Maxup [54], a variant of Mixup, is used in FM and NFM.

**LightGCN** [68]: It treats the user-item interaction as a bipartite graph and uses a graph convolutional network customized for collaborative filtering for prediction.

**Multi-VAE, Multi-DAE** [69]: Multi-VAE is an extension of variational autoencoder in collaborative filtering, in which Multi-DAE is the denoising version. Note that they are not suitable for Books dataset with millions of items.

3) **Evaluation Metrics**: Given that the FM-based models are widely applied in the re-ranking stage [4], [70], we adopt two metrics commonly used in this stage, LogLoss (logistic loss) and AUC (Area under the ROC Curve), which are also used in existing FM-based models [4], [5], [11], [12], [19]. LogLoss measures the fitting degree between the estimated distribution and the real one, and AUC can well reflect the model performance even with unbalanced data. The smaller LogLoss indicates better performance, while AUC is the opposite. We did 30 independent repeated experiments and reported the average results. Besides, the completed 30 rounds of results will be used for significance tests, and we employ the paired T-test. The null hypothesis in significance tests is that there is no significant difference between the performance of the two methods. During the calculation, the corresponding positions of the two paired comparison results share the same random seed.

4) **Parameter Settings**: For all models, we use Adam to optimize logistic loss. The learning rate is selected from $[0.0001, 0.0002, \ldots, 0.001, 0.005, 0.01]$, the batch size is searched from $[128, 256, 512, \ldots, 16384]$, and the embedding size is tuned from $[2, 4, 8, \ldots, 64, 128]$. Concretely, in Frappe and MovieLens, we set the embedding size to 64 and set it to 2 for the other datasets. In the experiments, we noticed that the larger the dataset, the smaller the learning rate needed to ensure training stability. Therefore, related to the dataset size, we set the learning rate of the seven datasets to $[0.0001, 0.0001, 0.0001, 0.0002, 0.0002, 0.0002, 0.00005]$. Besides, we use the learning rate decay, which is set to $[0.94, 0.94, 0.96, 0.98, 0.98, 0.96, 0.8]$. To balance training efficiency and convergence rate, the batch size is set to 2,048 in the first two datasets, and the last five ones are set to 16,384. Unless otherwise specified, we set the candidate neighbors $p$ to 10 in the first five datasets and set it to 2 in the last two large-scale datasets. Besides, the number of mixed data $n'$ is the same as the raw dataset size. The source codes are available at GitHub: https://github.com/Daftstone/SMFM.

**B. Performance Comparison**

We evaluate the performance concerning AUC and LogLoss, and report the overall results in Table II. Notably, even 0.1 basis-point improvements are significant in CTR tasks because millions of impressions will magnify it daily.

1) **Performance of Augmented FM-Based Models**: The first four rows of results in Table II focus on the standard FM, and we have the following findings. First, for CopyFM, despite the doubled training samples, compared with FM, the performance is only slightly fluctuating and even drops in some datasets (e.g., AUC in Pet). This finding aligns with the analysis in Section V-B that copying samples cannot improve model generalization under sufficient training. Second, MixFM has a significant improvement compared to FM. The performance has an average improvement of 9.5% for LogLoss, and in Frappe, the improvement even reaches 33.2%. The gratifying result echoes Theorem 5.6 that MixFM has a smaller generalization error. Third, SMFM achieves the best performance among all methods with respect to AUC, which implies the significant effect of using informative data. However, its LogLoss is not as good as MixFM, and in Books, it is even comparatively the worst. We reasonably suspect that SMFM essentially performs adversarial training on MixFM (in line with the analysis in
Section V-B, while adversarial training is often difficult to enjoy both prediction accuracy and model loss [71]. Given the above findings, if the task requirements tend to prediction accuracy, we suggest using SMFM; otherwise, MixFM. Finally, although the proposed methods could still significantly improve the model’s recommendation performance, it is undeniable that this improvement gradually diminishes when the dataset scale increases (from Frappe to Books). This reflects a key advantage of the proposed strategy: when the available training data is limited, it can lead to more substantial performance gains.

Furthermore, we illustrate the test curve of AUC and LogLoss during training, as shown in Figs. 3 and 4. Consistent with the results in Table II, our methods are in the lead. Besides, at the end of the training, overfitting appears in CopyFM but does not appear in MixFM and SMFM using the same amount of data. We reasonably believe that our methods add a data-adaptive regularization term, as shown in Theorem 5.5, which helps to alleviate overfitting.

We also migrate the proposed strategies to NFM, AFM, DCN, DAFM, and FINT to validate the effectiveness on FM variants, as shown in the middle part of Table II. Evidently, the performance of these models, configured with our methods (denoted as MixNFM, SMNFM, MixAFM, SMAFM, MixDCN, SMDCN, MixDAFM, SMDAFM, MixFINT, SMFINT), can still be further enhanced under most cases. Specifically, in an extensive comparison with 50 settings in five datasets, our methods surprisingly improve the performance in 94% of the settings, among which 76% of the settings are statistically significant. Such broad augmentation is exceedingly valuable and compelling.
Notably, although our methods do not show advantages w.r.t. LogLoss in some cases, the performance concerning AUC is improved. This is because FINT trained with hard labels tends to make overconfident predictions. In contrast, the proposed methods through a Mixup strategy can be considered label smoothing, which prevents overconfident predictions and may lead to a decrease in LogLoss. However, such soft label provides more subtle differences between samples, which helps to distinguish positive and negative samples better, thus getting better AUC. It is worth noting that FMs such as FINT are widely used in the re-ranking stage of recommendation [72], which makes the improvements of AUC (focuses on ranking performance) more important in actual business.

2) Comparison With Other Models: The lower part of Table II shows the performance of other FMs and non-FM models. First, as a variant of Mixup, although Maxup has some results close to our methods regarding AUC, LogLoss is significantly inferior. This is because Maxup performs adversarial training with a min-max optimization, challenging to enjoy both prediction performance and loss. Second, non-FM models such as LightGCN cannot use additional interactive features, resulting in poor performance in our experiments. Especially in MovieLens, where more than half of users do not have positive user-item interaction. Moreover, FM-based models studied in the paper are widely used in the re-ranking stage (metrics: AUC, LogLoss), while non-FM models such as Mul-VAE mainly focus on the matching stage (metrics: Recall, NDCG). Although the different emphases make a fair comparison impossible, displaying these results reflects the advantage of FM-based models in the re-ranking stage.

C. Performance Under Different Training Data

In this section, we test the performance of models under different training dataset sizes. Here, we conduct experiments on two large-scale datasets, Criteo and Avazu, and the results are shown in Fig. 5. It is evident that the proposed approaches remain competitive across differing quantities of training instances. Concurrently, this result offers an additional advantage - the lesser the training data, the more pronounced gain conferred by our proposed strategy to the backbone model, consistent with our findings in Section VI-B1. The finding is highly meaningful. This is because, from the data perspective, when the model performance is already considerable, even 0.1 basis-point improvement often requires support from several times more data. However, in many scenarios, high-quality samples usually need time-consuming human effort and possibly expert knowledge, thereby increasing the difficulty of data collection. The aforementioned advantage may mitigate data pressure, enabling FMs to achieve satisfactory performance with an acceptable amount of data.

D. Sensitivity w.r.t. Interpolation Coefficient $\lambda$

This section conducts sensitivity analysis about the interpolation coefficient $\lambda$, which determines the strength of feature mixing. Note that $\lambda$ is sampled from the Beta distribution, Beta($\alpha$, $\beta$); therefore, the sensitivity analysis on $\lambda$ is equivalent to the one on $\alpha$ and $\beta$. Following the existing strategies [15], [59], we set them to be the same and evaluate the sensitivity concerning $\alpha$ (or $\beta$, denoted as $\alpha(\beta)$), as shown in Fig. 6. It is worth noting that setting $\alpha(\beta)$ to 0 corresponds to $\lambda = 0$ or 1, which results in the algorithm degenerating into CopyFM. We can find that as the interpolation coefficient $\lambda$ increases (larger $\alpha(\beta)$), the model’s performance steadily improves until it reaches saturation. Considering CopyFM corresponds to $\alpha(\beta) = 0$, the positive correlation trend confirms that the proposed feature mixing strategy contributes to improving the model. Furthermore, the significant improvement near 0 indicates that even with a small mixing strength, the improvement effect of the proposed...
Fig. 5. Performance comparison under the different training dataset sizes.

Fig. 6. Performance under different $\alpha(\beta)$ in the Beta distribution.

Fig. 7. Performance under the different number of mixed samples $n'$.

algorithm remains substantial. This good property weakens the need to meticulously select parameters (defaulted to 0.5), since it yields good performance in most cases.

E. Sensitivity w.r.t. Mixed Data $n'$

In this section, we explore the effects of “poisoning” different numbers of mixed users ($n'$ in Algorithm 1) on performance, and Fig. 7 reports the experimental results in MixFM and MixNFM. The absissa represents the ratio of the injected neighbor data to the original sample; thus, 0 in absissa means a standard FM. The ordinate denotes the improvement of AUC over standard FM. Overall, the model performance improves with the increase of mixed data. However, the improvement is not infinite, and the additional information seems to be saturated when the mixed data reaches a threshold, manifested in performance fluctuations (e.g., MixNFM on MovieLens) or decline (e.g., MixNFM on Electronics). In addition, the performance improvement of MixNFM is more significant than that of MixFM. We reasonably believe that NFM uses DNN to capture high-level interactions between features, which can mine the auxiliary information of the mixed data better than FM. Comparing different datasets, we find that the performance improvement will decrease as the data increases. Despite this, MixFM still has an AUC improvement of at least 0.004 when the data is doubled.

F. Sensitivity w.r.t. The Number of Candidate Neighbors $p$

This section studies the performance sensitivity when using different numbers of candidate neighbors ($p$ in Algorithm 2) in SMFM and SMNFM, as shown in Fig. 8. The absissa denotes the number of candidate neighbors generated for each raw sample, and the ordinate is the performance improvement compared to MixFM and MixNFM, respectively. First, compared to generating one neighbor (MixFM or MixNFM), the performance of generating multiple neighbors to select the most significant one is improved, which stresses the effectiveness of the proposed Saliency-guided Mixup. Second, the sample’s saliency (i.e., the number of candidate neighbors) and performance are not entirely positively correlated. For example, in MovieLens, when the number of neighbors generated is 8, the performance of SMNFM reaches saturation, and more neighbors even reduce the performance. This is mainly because excessively strong saliency samples will cause serious crossover mixture problems between raw and mixed data, thereby increasing the difficulty of training. Notably, a similar conclusion can be found in [73].
Fig. 9. Performance comparison (AUC) under different embedding sizes $d$.

Fig. 10. FM’s generalization validation on Avazu. (1) The effect of the number of data $n$ on generalization. (2) The effect of the number of non-zero features $\tau$ on generalization. (3) The effect of the $\gamma$ on generalization. (4) The effect of the embedding size $d$ on generalization.

Fig. 11. MixFM’s generalization validation on Avazu. (1) The effect of the number of data $n$ on generalization. (2) The effect of the number of non-zero features $\tau$ on generalization. (3) The effect of the $\gamma$ on generalization.

G. Sensitivity w.r.t. The Embedding Size $d$

The embedded size $d$ is vital in capturing the features’ interaction. Besides, we analyze that the embedding size needs to compromise between model complexity and empirical error in Corollary 5.3. The performance comparison concerning AUC under different embedding sizes is illustrated in Fig. 9.

First, MixFM and SMFM under various settings are still leading, and SMFM achieves the best performance in all experimental models. The result emphasizes the effectiveness of our methods. Second, in Frappe and MovieLens, the performance positively correlates with the embedding size, but it is the opposite in Amazon’s three datasets. The possible reason is that the data volume of Frappe and MovieLens is small, so the empirical error is dominant compared to the model complexity, which requires a larger $d$ to ensure a smaller empirical error. For the last three datasets with sufficient data, a small embedding size may already guarantee a small empirical error, and the model complexity is the major concern. Therefore, we suggest using a smaller embedding size for the dataset with sufficient data to enjoy both generalization and computational efficiency.

H. Quantitative Analysis w.r.t. Generalization

To validate the generalization bound, we perform a quantitative analysis of the generalization gap (performance gap in training and testing sets) under various effect factors to experimentally verify the correctness of the generalization bound, similar to existing theoretical methodologies [59], [74]. The results are depicted in Figs. 10 and 11. According to the theoretical results of Corollary 5.3 and Theorem 5.6, in FM and MixFM, the generalization gap ($\overline{R}_D(f) - \hat{R}_D(f)$) is proportional to $\gamma$ and $\tau$ while inversely proportional to $n$. Besides, in FM, the generalization gap is proportional to $d$. It is evident that the generalization curve presented in Figs. 10 and 11 aligns with this theoretical finding (note that generalization is inversely proportional to weight decay, and weight decay is inversely proportional to $\gamma$, so the generalization gap is directly proportional to $\gamma$ in experiments). In summary, the consistency between the quantitative analysis findings and the theoretical findings further verified the correctness of the generalization bound.

I. Performance Under Different Amounts of Information

To verify the effectiveness of the proposed methods under different amounts of information, we construct a new dataset, Mini-Avazu, which randomly removes 4 fields of features from Avazu. We add these features as auxiliary information to Mini-Avazu for training and compare the performance of different methods. The results are shown in Table III, where “Origin” means no
TABLE III
PERFORMANCE UNDER DIFFERENT AMOUNTS OF INFORMATION

| Metric  | Method   | Origin | Feat+1 | Feat+2 | Feat+3 | Feat+4 |
|---------|----------|--------|--------|--------|--------|--------|
| Dataset |          |        |        |        |        |        |
| Mini-Avazu | FM   | 0.7363 | 0.7359 | 0.7297 | 0.7302 | 0.7375 |
|          | CopyFM  | 0.7362 | 0.7357 | 0.7295 | 0.7300 | 0.7374 |
|          | MixFM   | 0.7405 | 0.7385 | 0.7345 | 0.7351 | 0.7398 |
|          | SMFM    | 0.7386 | 0.7403 | 0.7366 | 0.7359 | 0.7356 |

| t-test | ***    | ***    | ***    | ***    |

| Metric  | Method   | Origin | Feat+1 | Feat+2 | Feat+3 | Feat+4 |
|---------|----------|--------|--------|--------|--------|--------|
| Dataset |          |        |        |        |        |        |
| Mini-Avazu | FM   | 0.3925 | 0.3915 | 0.3834 | 0.3921 | 0.3890 |
|          | CopyFM  | 0.3926 | 0.3917 | 0.3834 | 0.3922 | 0.3891 |
|          | MixFM   | 0.3908 | 0.3905 | 0.3915 | 0.3906 | 0.3886 |
|          | SMFM    | 0.3844 | 0.3840 | 0.3860 | 0.3856 | 0.3866 |

| t-test | ***    | ***    | ***    | ***    |

Fig. 12. Performance under different perturbation sizes.

additional information and “Feat+n” indicates incorporating n additional features. First, we find that not all information incorporation can enhance FMs, such as the comparison between Feat+1 and Feats+2. This demonstrates their limitations that additional information incorporation requires careful selection. Second, our methods are better than baselines, which verifies that our augmentation strategies do not lose out on the possible benefits of auxiliary information. This can also be further verified from our theoretical results. Since we only make assumptions about the 0 norm of data x, different numbers of features mean different \( \tau \). However, as can be seen from Remark 2 in Section V, the performance comparison is independent of \( \tau \). Therefore, the proposed strategies do not destroy the benefits of additional information.

J. Robustness Evaluation

Recent studies have shown that recommender systems are vulnerable [75]: adding small perturbations to the dataset can significantly reduce the recommendation quality. In this section, we test the performance of MixFM facing perturbations with different sizes. Fig. 12 shows the model’s performance degradation when random noises of different sizes are added to the Frappe and Electronics (we have similar results on other datasets). First, we observe that the FM is fragile, and its performance will reduce significantly with the increase in noise intensity. Second, the AUC reduction on Electronics is smaller than that of Frappe. The possible reason is that Electronics has more data, corresponding to the finding that model robustness requires more data [76]. Finally, compared with FM, MixFM has a smaller reduction concerning AUC, revealing that using additional mixed data helps to improve FM’s robustness.

K. Training Runtime

Since the proposed strategies do not change models, no delay in inference occurs, which is critical in real-world applications. However, it is undeniable that the training time complexity is linearly proportional to the dataset size. Table IV shows the 50-round training time (in seconds) when MixFM and SMFM double the data, so does the training time, which is consistent with the analysis. Notably, Fig. 7 shows that a considerable improvement requires fewer mixed samples (e.g., Frappe only supplemented 10% augmented data in NFM), while Fig. 4 shows that the proposed augmentation approaches are easier to converge. In summary, the actual training delay is much smaller, which is tolerable.

VII. Conclusion

In this paper, we proposed two novel approaches, MixFM and SMFM, to boost FMs by generating auxiliary data. They do not need additional knowledge and can establish direct interaction with original non-interactive features. Besides, we delivered the first generalization bound for FM, and we theoretically proved that our methods have a better generalization guarantee than FM. Finally, we conducted extensive experiments on multiple real-world datasets, and the results demonstrated the effectiveness of our proposals.

While the proposed exhibits promising results across various FMs, akin to other methods, it also harbors several potential limitations worth consideration: (1) different datasets might possess different optimal coefficients, necessitating careful experiments for selection; (2) the generated augmented samples are inconsistent with the original data, which potentially reduce interpretability; (3) reducing the training time to an insignificant increase is worth investigating.

Notably, in theory, the proposed strategies can be applied to any sparse data (may also apply to continuous data if we treat each dimension as a feature). We present some potential applications using sparse coding:

- Natural Language Processing (NLP): The traditional mixing strategy is based on the embedding level. In contrast, the proposed strategies can be applied to inputs encoded by Bag-of-Words or Term Frequency-Inverse Document Frequency. It directly combines two samples at the input
level to generate more diverse and enriched samples. Notably, compared to mixing at the high-dimensional and abstract embedding level, input-level mixing may offer greater interpretability.

- Anomaly Detection: In e-commerce user behavior data, ad traffic data, and financial transaction logs, the category features often exhibit a wide range of discrete values, resulting in high-dimensional and sparse data after encoding (usually one-hot encoding). The proposed methods may generate data that includes more diverse anomalous behaviors, thereby enhancing the ability to generalize to unknown anomalies.

In genetics, gene expression study can often involve analyzing thousands of genes. However, only a few genes may be active in a given context, resulting in sparse data. Utilizing the proposed methods, it is possible to synthesize more diverse functional gene products, which may provide insights for drug design and synthetic biology applications. Indeed, it is an interesting area for research, and we consider it a promising direction for future studies.

**REFERENCES**

[1] B. Yang, Y. Lei, J. Liu, and W. Li, “Social collaborative filtering by trust,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 8, pp. 1633–1647, Aug. 2017.

[2] D. Lian, H. Wang, Z. Liu, J. Lian, E. Chen, and X. Xie, “LightRec: A memory and search-efficient recommender system,” in Proc. Web Conf., 2020, pp. 695–705.

[3] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas, “Gaussian process factorization machines for context-aware recommendations,” in Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2014, pp. 63–72.

[4] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, “xDeepFM: Combining explicit and implicit feature interactions for recommender systems,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 1754–1763.

[5] S. Rendle, “Factorization machines,” in Proc. IEEE Int. Conf. Data Mining, 2010, pp. 950–1000.

[6] J. Chen et al., “Automated creative optimization for e-commerce advertising,” in Proc. Web Conf., 2021, pp. 2304–2313.

[7] R. J. Oentaryo, E.-P. Lim, J.-W. Low, D. Lo, and M. Finegold, “Predicting response in mobile advertising with hierarchical importance-aware factorization machine,” in Proc. 7th ACM Int. Conf. Web Conf., 2018, pp. 669–678.

[8] C.-T. Lu, H. He, H. Ding, B. Cao, and P. S. Yu, “Learning from multi-view multi-way data via structural factorization machines,” in Proc. World Wide Web Conf., 2018, pp. 1593–1602.

[9] B. Cao, H. Zhou, G. Li, and P. S. Yu, “Multi-view machines,” in Proc. 9th ACM Int. Conf. Web Search Data Mining, 2016, pp. 427–436.

[10] X. Zhao et al., “AutoField: Field-aware interaction neural network for click-through rate prediction,” in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2017, pp. 701–709.

[11] Y. Wang, X. Zhao, Y. Zhu, and W. Wang, “AutoDenoise: Automatic data instance denoising for recommendations,” in Proc. 2nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 255–262.

[12] B. Gao, R. Tang, Y. Ye, Z. Li, and X. He, “DeepFM: A factorization-machine-based neural network for CTR prediction,” in Proc. WWW, 2017, arXiv:1708.04617.

[13] Z. Wang, J. Ma, Y. Zhang, Q. Wang, J. Ren, and P. Sun, “Attention-over-attention field-aware wide and deep learning,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 6233–6330.

[14] X. Zhao et al., “Field-aware embedding dimension Searchin,” in Proc. Web Conf., 2021, pp. 2304–2313.

[15] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, “Attentional factorization machines: Learning the weight of feature interactions via attention networks,” in Proc. 16th ACM Conf. Recomender Syst., 2022, pp. 378–388.

[16] Z. Wang, J. Ma, Y. Zhang, Q. Wang, J. Ren, and P. Sun, “Attention-over-attention field-aware wide and deep learning,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 6233–6330.

[17] M. Yamada et al., “Convex factorization machine for toxicogenomics prediction,” in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2017, pp. 1215–1224.

[18] D. Lian, Z. Zhang, Y. Ge, F. Zhang, N. J. Yuan, and X. Xie, “Regularized content-aware tensor factorization meets temporal-aware location recommendation,” in Proc. IEEE Int. Conf. Data Mining, 2016, pp. 1029–1034.

[19] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, “Higher-order factorization machines,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3351–3359.

[20] Y. Zhao, S. Yang, G. Liu, D. Feng, and K. Xu, “FINT: Field-aware interaction neural network for click-through rate prediction,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2022, pp. 3913–3917.

[21] A. Sarkar, D. Das, V. Sembium, and P. Mandayam Comar, “Dual attentional higher order factorization machines,” in Proc. 16th ACM Conf. Recomender Syst., 2022, pp. 378–388.

[22] J. Pan, Y. Mao, A. L. Ruiz, Y. Sun, and A. Flores, “Predicting different types of conversions with multi-task learning in online advertising,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2019, pp. 2689–2697.

[23] Y. Sun, J. Pan, A. Zhang, and A. Flores, “FM2: Field-matrixed factorization machines for recommender systems,” in Proc. Web Conf., 2021, pp. 2828–2837.

[24] Y. Zhao, M. Zhang, B. Fu, G. Fu, and M. Wang, “Deep & cross network for ad click predictions,” in Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining, 2016, pp. 1754–1763.

[25] X. Zhao et al., “AutoField: Field-aware interaction neural network for click-through rate prediction,” in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2017, pp. 701–709.

[26] C.-T. Lu, H. He, H. Ding, B. Cao, and P. S. Yu, “Learning from multi-view multi-way data via structural factorization machines,” in Proc. World Wide Web Conf., 2018, pp. 1593–1602.

[27] D. Rügamer, “Additive higher-order factorization machines,” 2022, arXiv:2205.14515.

[28] Y. Wang, X. Zhao, T. Xu, and X. Wu, “AutoField: Automating feature mixing,” 2021, arXiv:2104.08070.

[29] D. Lian, Z. Zhang, Y. Ge, F. Zhang, N. J. Yuan, and X. Xie, “Regularized content-aware tensor factorization meets temporal-aware location recommendation,” in Proc. IEEE Int. Conf. Data Mining, 2016, pp. 1029–1034.

[30] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, “Higher-order factorization machines,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3351–3359.
[42] W. Lin, X. Zhao, Y. Wang, T. Xu, and X. Wu, “AdaFS: Adaptive feature selection in deep recommender system,” in Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining, 2022, pp. 3309–3317.

[43] Y. Chen, Y. Wang, P. Ren, M. Wang, and M. de Rijke, “Bayesian feature interaction model for factorization machines,” Artif. Intell., vol. 302, 2022, Art. no. 103589.

[44] F. Lyu, X. Tang, D. Liu, L. Chen, X. He, and X. Liu, “Optimizing feature set for click-through rate prediction,” 2023, arXiv:2301.10909.

[45] Y. Zheng, P. Wei, Z. Chen, Y. Cao, and L. Lin, “Graph-convolved factorization machines for personalized recommendation,” IEEE Trans. Knowl. Data Eng., vol. 35, no. 2, pp. 1567–1580, Feb. 2023.

[46] Z. Li, S. Wu, Z. Cui, and X. Zhang, “GraphFM: Graph factorization machines for feature interaction modeling,” 2021, arXiv:2105.11866.

[47] Z. Tian et al., “Directed acyclic graph factorization machines for CTR prediction via knowledge distillation,” in Proc. 16th ACM Int. Conf. Web Search Data Mining, 2023, pp. 715–723.

[48] Q. Zhang, L. Rao, and Y. Yang, “Group-CAM: Group score-weighted visual explanations for deep convolutional networks,” 2021, arXiv:2103.13859.

[49] X. Xu, Z. Wang, C. Deng, H. Yuan, and S. Ji, “Towards improved and interpretable deep metric learning via attentive grouping,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 1, pp. 1189–1200, Jan. 2023.

[50] B. Zhou, D. Bau, A. Oliva, and A. Torralba, “Interpreting deep visual representations via network dissection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 9, pp. 2311–2314, Sep. 2019.

[51] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” in Proc. Int. Conf. Learn. Representations, 2014, pp. 1–8.

[52] A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” in Proc. Adv. Neural Inf. Process. Syst., 2019, Art. no. 721.

[53] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous distributed systems,” 2016, arXiv:1603.04467.

[54] C. Gong, T. Ren, M. Ye, and Q. Liu, “MaxUp: A simple way to improve generalization of neural network training,” 2020, arXiv:2002.09024.

[55] J. Zhang et al., “Uncertainty inspired RGB-D saliency detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 9, pp. 5761–5779, Sep. 2022.

[56] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in Proc. IEEE Eur. Symp. Secure. Privacy, 2016, pp. 372–387.

[57] Y. Xing, Q. Song, and G. Cheng, “On the generalization properties of adversarial training,” in Proc. Int. Conf. Artif. Intell. Inform. Process., 2021, pp. 505–513.

[58] C. Song, K. He, L. Wang, and J. E. Hopcroft, “Improving the generalization of adversarial training with domain adaptation,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–14.

[59] L. Zhang, Z. Deng, K. Kawaguchi, A. Ghorbani, and J. Zou, “How does data help with robustness and generalization?” 2020, arXiv:2010.04319.

[60] R. Anan, P. Bartlett, P. Mianjy, and N. Srerbro, “Dropout: Explicit forms and capacity control,” in Proc. Int. Conf. Mach. Learn. Mach. Intel., 2021, pp. 351–361.

[61] A. Blumer, “Learnability models and Vapnik-Chervonenkis combinatorics,” in Proc. 8th Int. Conf. Probability, Learning Inform. Theory, Springer, 1992, pp. 390–402.

[62] D.-X. Zhou, “The covering number in learning theory,” J. Complexity, vol. 18, no. 3, pp. 739–767, 2002.

[63] P. L. Bartlett and S. Mendelson, “Rademacher and Gaussian complexities: Risk bounds and structural results,” J. Mach. Learn. Res., vol. 3, pp. 463–482, 2002.

[64] M. Mohri, A. Rostamizadeh, and A. Talwalkar, Foundations of Machine Learning. Cambridge, MA, USA: MIT Press, 2018.

[65] L. Baltrunas, K. Church, A. Karatzoglou, and N. Oliver, “Frappe: Unsupervised learning from an ordered set,” in Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining, 2022, pp. 639–648.

[66] D.-L. Jiang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, “Variational autoencoders for collaborative filtering,” in Proc. World Wide Web Conf., 2018, pp. 689–698.

[67] W. Cheng, Y. Shen, and L. Huang, “Adaptive factorization network: Learning adaptive-order feature interactions,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 3609–3616.

[68] C. Liu, M. Sulzmann, T. Lin, R. Tomioka, and S. Süssstrunk, “On the loss landscape of adversarial training: Identifying challenges and how to overcome them,” in Proc. 34th Int. Conf. Neural Inf. Process. Syst., 2020, Art. no. 1803.

[69] W. Krichene and S. Rendle, “On sampled metrics for item recommendation,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 1748–1757.

[70] J. Zhang et al., “Attacks which do not kill training make adversarial learning stronger,” in Proc. 37th Int. Conf. Mach. Learn., 2020, pp. 11278–11287.

[71] L. Deng, D. Lian, C. Wu, and E. Chen, “Graph convolution network based recommender systems: Learning guarantee and item mixture powered strategy,” in Proc. Adv. Neural Inf. Process. Syst., 2022, pp. 5906–5912.

[72] X. He, Z. He, X. Du, and T.-S. Chua, “Adversarial personalized ranking for recommendation,” in Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2018, pp. 355–364.

[73] C. Wu, D. Lian, Y. Ge, Z. Zhu, E. Chen, and S. Yuan, “Fight fire with fire: Towards robust recommender systems via adversarial poisoning training,” in Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2021, pp. 1074–1083.

Chenwu Wang received the BS degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, in 2018. He is currently working toward the PhD degree with the School of Data Science, University of Science and Technology of China. His current research interests include recommender systems and deep learning. He has published several papers in top-tier journals and conferences in his research areas, including IEEE Transactions on Pattern Analysis and Machine Intelligence, KDD, and SIGIR.

Defu Lian received the PhD degree in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2014. He is currently a professor with the School of Computer Science and Technology, USTC. He has published prolifically in referred journals and conference proceedings, such as IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Knowledge and Data Engineering and ACM Transactions on Information Systems. Conference on Neural Information Processing Systems, IEEE International Conference on Data Mining (ICDM), ACM SIGKDD Conference on Knowledge Discovery and Data Mining, ACM International Conference on Research on Development in Information Retrieval, International Joint Conferences on Artificial Intelligence, and ACM International World Wide Web Conferences. His current research interest includes spatial data mining, recommender systems, and learning to hash.

Yong Ge received the PhD degree in information technology from Rutgers Business School at Rutgers, The State University of New Jersey, in 2013. He is currently an associate professor with management information systems with the Eller College of Management, University of Arizona. His primary research interests include data mining, Big Data, machine learning, recommender systems, personalization services, social networking, target marketing, talent analytics, and business analytics. He has published more than 100 journal and conference papers at both CS and IS outlets, such as IEEE Transactions on Knowledge and Data Engineering, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, and MIS Quarterly. He received the NSF CAREER Award, in 2019.
Min Zhou received the BS degree in automation from the University of Science and Technology of China, and the PhD degree from Industrial Systems Engineering and Management Department, National University of Singapore, respectively. She is currently a principal research engineer of Huawei Noah’s Ark Lab, Shenzhen, China. Her interests include pattern mining and machine learning, and their applications in sequence and graph data. Her several works related to graph learning and mining were published at top conferences, including KDD, WWW, ICDE, and SIGIR.

Enhong Chen (Senior Member, IEEE) received the PhD degree in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 1996. He is currently a professor and the vice dean of the School of Computer Science, USTC. He has published more than 200 papers in refereed conferences and journals, including the IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Mobile Computing, ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), IEEE International Conference on Data Mining (ICDM), Conference on Neural Information Processing Systems, and ACM International Conference on Information and Knowledge Management. His current research interests include data mining and machine learning, social network analysis, and recommender systems. He was a recipient of the Best Application Paper Award on KDD2008, the Best Research Paper Award on ICDM-2011, and the Best of SIAM International Conference on Data Mining (SDM)-2015. He was on the program committees of numerous conferences, including KDD, ICDM, and SDM.

Dacheng Tao (Fellow, IEEE) is the Inaugural director of the JD Explore Academy and a senior vice president of JD.com. He is also an advisor and chief scientist of the digital science institute in the University of Sydney. He mainly applies statistics and mathematics to artificial intelligence and data science, and his research is detailed in one monograph and more than 200 publications in prestigious journals and proceedings at leading conferences. He received the 2015 Australian Scopus-Eureka Prize, the 2018 IEEE ICDM Research Contributions Award, and the 2021 IEEE Computer Society McCluskey Technical Achievement Award. He is a fellow of the Australian Academy of Science, AAAS, and ACM.