Artificial Intelligence 302 (2022) 103589

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Bayesian feature interaction selection for factorization machines

Yifan Chen [a], Yang Wang [b,c,*], Pengjie Ren [d], Meng Wang [b,c], Maarten de Rijke [d,e]

[a] National University of Defense Technology, Changsha, China
[b] Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education, Hefei University of Technology, China
[c] School of Computer Science and Information Engineering, Hefei University of Technology, China
[d] University of Amsterdam, Amsterdam, the Netherlands
[e] Ahold Delhaize, Zaandam, the Netherlands

ARTICLE INFO

Article history:
Received 11 May 2020
Received in revised form 5 July 2021
Accepted 18 August 2021
Available online 26 August 2021

Keywords:
Factorization machine
Bayesian variable selection
Feature interaction selection

ABSTRACT

Factorization machines are a generic supervised method for a wide range of tasks in the field of artificial intelligence, such as prediction, inference, etc., which can effectively model feature interactions. However, handling combinations of features is expensive due to the exponential growth of feature interactions with the order. In nature, not all feature interactions are equally useful for prediction. Recently, a large number of methods that perform feature interaction selection have attracted great attention because of their effectiveness at filtering out useless feature interactions. Current feature interaction selection methods suffered from the following limitations: (1) they assume that all users share the same feature interactions; and (2) they select pairwise feature interactions only. In this paper, we propose novel Bayesian variable selection methods, targeting feature interaction selection for factorization machines, which effectively reduce the number of interactions. We study personalized feature interaction selection to account for individual preferences, and further extend the model to investigate higher-order feature interaction selection on higher-order factorization machines. We provide empirical evidence for the advantages of the proposed Bayesian feature interaction selection methods using different prediction tasks.

1. Introduction

Factorization machines (FMs) [50,51] are a generic supervised learning approach that combines the advantages of support vector machines (SVMs) [61] with factorization models [31]. FMs account for interactions between features with factorized parameters [5,58]; such interactions are defined as combinations of individual features [13]. For example, in a movie recommendation scenario, the features for the movie "Spider-Man" can be "*comics*", "*marvel*" and "*avengers*". Accordingly, feature interactions can be seen as combinations such as, e.g., "(*comics*, *marvel*)", "(*comics*, *avengers*)."

* Corresponding author at: School of Computer Science and Information Engineering, Hefei University of Technology, China.
E-mail addresses: yfchen@nudt.edu.cn (Y. Chen), yangwang@hfut.edu.cn (Y. Wang), p.ren@uva.nl (P. Ren), eric.mengwang@gmail.com (M. Wang), derijke@uva.nl (M. de Rijke).

https://doi.org/10.1016/j.artint.2021.103589
0004-3702/

FMs are widely applied in the field of artificial intelligence (e.g., predictive analytics, inference, etc.), ranging from recommendation [52], computational advertising [23] and search ranking [39] to toxicogenomics prediction [75]. FMs have been well studied for recommendation tasks, by exploiting historical interactions between users and items [50]. Specifically, FMs can incorporate additional information associated with users or items, including content descriptions [76], context information [54], social networks [6,10], and sequential dependencies [33]. While the wide availability of auxiliary data provides rich sources to reveal the user preferences, it increases the dimensionality of the feature space [9]. The curse of dimensionality is particularly severe for FMs, because FMs consider feature interactions. The complexity of FM models grows exponentially with the order of feature interactions. However, not all feature interactions are helpful [12]. Incorporating non-informative and noisy feature interactions will impacts the accuracy and increases the difficulty of interpreting outcomes [73].

To address the above problem, *feature interaction selection* (FIS) has been proposed to filter out useless feature interactions. Existing feature interaction selection (FIS) methods can be divided into two classes: *wrapper methods* [40] and *embedded methods* [12,73]. *Wrapper methods* evaluate subsets of feature interactions by training a model with each subset and scoring on a held-out set. Although wrapper methods are more flexible as they do not depend on the model, they suffer from poor scalability. *Embedded methods* perform FIS during model training, which are more efficient and effective. Sparse factorization machines (SparseFM) [47,73,79] are examples of embedded methods; they exploit sparsity regularization [63,67] to achieve FIS. However, SparseFM requires the expensive cross-validation.

To avoid the expensive cross-validation [63], we opt for Bayesian variable selection (BVS) methods. To achieve FIS, BVS typically assumes the variables follow sparsity prior distributions [3]. The sparsity prior ensures that when the variable is small, it is more likely to be zero. Therefore, the less important or noisy feature interactions can be filtered out. However, the fundamental issue of sparsity prior is that they assign zero probability to the event that a variable equals zero. Instead, we employ spike-and-slab priors (SSPs) [2,17,43], to distinguish between useful and harmful interactions. SSPs explicitly assign non-zero probabilities to events [64].

To these ends, we propose Bayesian feature interaction selection (BFIS) methods to address the problem of FIS for FMs. In our previous work [8], we have studied pairwise feature interaction selection for FMs. Higher-order feature interactions are also considered for FMs [4], in order to understand the complex and non-linear underlying structure in data. Therefore, in this paper, we make substantial improvements over the preliminary version in [8] by extending pairwise FIS to high-order. Compared with pairwise FIS, the essential requirements for a well designed model for higher-order feature interaction selection (H-FIS) are the following:

- *Heredity*: the model should capture dependencies between feature interactions of increasingly orders. The heredity between first- and second-order feature interactions that has been studied in [8] does not necessarily hold for higher-order feature interactions.
- *Enumeration*: the model needs to enumerate all feature interactions comprehensively and without repetition. Brute-force enumeration is costly and a systematic way will be necessary.
- *Efficiency*[1]: the model should be able to be trained efficiently. It is time-consuming to select high-order feature interactions since it involves combinational search.

Contributions By systematically considering these requirements for FIS on higher-order factorization machines (HOFMs), this paper is a substantial extension of our previously published work [8]:

(1) We investigate feature interaction selection (FIS) for higher-order factorization machines (HOFMs). higher-order feature interaction selection (H-FIS) is an effective way to reduce the number of higher-order feature interactions, thereby improving prediction quality.
(2) We propose a Bayesian variable selection (BVS) method to capture the heredity between feature interactions of increasingly higher orders.
(3) We conduct variational inference with reparameterization and form inference networks to efficiently and effectively optimize the model. The efficiency of H-FIS is proved through training on large-scale datasets.
(4) We carry out extensive experiments on benchmark datasets to validate the effectiveness of our model in term of handling higher-order feature interactions.

Roadmap The rest of paper is structured as follows. We first introduce preliminaries in Section 2. We discuss pairwise feature interaction selection in Section 3. Based on that, we extend the idea of exploiting pairwise feature interactions to select higher-order feature interactions in Section 4. We introduce our experimental settings in Section 5 and report and analyze our experimental results in Section 6. Section 7 discusses related work. We conclude the paper in Section 8.

[1] Note that "efficiency" means the training efficiency of performing H-FIS on FMs rather than producing efficient FMs.

Table 1
Summary of symbols and notation.

	Notation	Description
Sets and numbers	$\mathcal{U}$	Set of users
	$\mathcal{F}$	Set of features
	$\mathcal{X}$	Collection of feature vectors, i.e., $\mathcal{X} = \{\boldsymbol{x}\}$
	$\mathcal{Y}$	Collection of labels, i.e., $\mathcal{Y} = \{y(\boldsymbol{x})\}$
	$\mathcal{D}$	Collection of dataset, i.e., $\mathcal{D} = \mathcal{X} \cup \mathcal{Y}$
	M	Number of users, i.e., $M = \lvert\mathcal{U}\rvert$
	D	Number of features, i.e., $D = \lvert\mathcal{F}\rvert$
	K	Dimension of feature embedding
Bayesian variables	w_i	First-order feature interaction weight for feature i
	w_{ui}	Personalized first-order feature interaction weight of user u for feature i
	w_{uij}	Personalized feature interaction weight of user u for interaction $\langle i, j\rangle$
	$w^{(t)}_{j_1,\ldots,j_t}$	The weight for the t-th order interaction $\langle j_1, \ldots, j_t\rangle$
	s_{ui}	The selection variable of user u for feature i
	$\tilde{w}_i$	The weight of feature i
	s_{uij}	The selection variable of user u for interaction $\langle i, j\rangle$
	$\tilde{w}_{ij}$	The weight of interaction $\langle i, j\rangle$
	$s_{j_1,\ldots,j_t}$	The selection variable for interaction $\langle j_1, \ldots, j_t\rangle$
	$\tilde{w}^{(t)}_{j_1,\ldots,j_t}$	The weight of interaction $\langle j_1, \ldots, j_t\rangle$
Parameters	$\boldsymbol{v}^{(t)}_i \in \mathbb{R}^K$	The t-th order latent representation of feature i for factorizing weights. We write $\boldsymbol{v}_i = \boldsymbol{v}^{(2)}_i$
	$\boldsymbol{z}_i \in \mathbb{R}^K$	The latent representation of feature i for interaction selection
	b_0	The global bias of FM (HOFM)
	b_u	The personalized bias of user u
	π_{ui}	The personalized selection parameter of user u for feature i
	$\pi_{j_1,\ldots,j_t}$	The selection parameter for interaction $\langle j_1, \ldots, j_t\rangle$

2. Preliminaries

We denote $\mathcal{U}$ and $\mathcal{F}$ as the set of users and features, respectively. Note that user identifiers are also one of the features, i.e., $\mathcal{U} \subset \mathcal{F}$. The number of users and features are denoted by $M = |\mathcal{U}|$ and $D = |\mathcal{F}|$, respectively. We use $\mathcal{D} = \mathcal{X} \cup \mathcal{Y}$ to represent the collection of data, where $\mathcal{X}$ and $\mathcal{Y}$ are respectively the collection of feature vectors (input) and the collection of labels (output). Each feature vector $\boldsymbol{x} \in \mathbb{R}^D$ in $\mathcal{X}$ is associated with a label $y(\boldsymbol{x}) \in \mathcal{Y}$. We use x_i to represent the i-th element of $\boldsymbol{x}$. We summarize notations in Table 1.

We briefly introduce factorization machines (FMs) [50,51]. Given a real-valued feature vector $\boldsymbol{x} \in \mathbb{R}^D$, FMs predict the target score by:

$$\hat{y}_{FM}(\boldsymbol{x}) = b_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} w_{ij} x_i x_j, \tag{1}$$

where b_0 is the global bias, w_i is the first-order coefficient and w_{ij} is the second-order coefficient. Instead of learning w_{ij} directly, FMs factorize it as $w_{ij} = \boldsymbol{v}_i^T \boldsymbol{v}_j$, where $\boldsymbol{v}_i \in \mathbb{R}^K$ is the embedding of feature i. K is the dimension of the latent space.

We also introduce HOFMs [4,29], which extend FMs to consider higher-order feature interactions. Given $\boldsymbol{x}$, higher-order factorization machines (HOFMs) predict the score by:

$$\hat{y}_{HOFM}(\boldsymbol{x}) = b_0 + \sum_{i \in \mathcal{F}} w_i x_i + \sum_{t=2}^{H} \sum_{j_1 < \cdots < j_t \in \mathcal{F}} w^{(t)}_{j_1,\ldots,j_t} x_{j_1} \cdots x_{j_t}, \tag{2}$$

where $\hat{y}_{HOFM}(\boldsymbol{x})$ indicates the predicted score by HOFM given feature $\boldsymbol{x}$; H is the maximum order of feature interactions; w_0 and w_i are the global bias and the regression coefficient; and $w^{(t)}_{j_1,\ldots,j_t}$ is the coefficient for the t^{th}-order feature interaction $\langle j_1, \ldots, j_t\rangle$, which is further factorized:

$$w^{(t)}_{j_1,\ldots,j_t} = \text{sum}(\boldsymbol{v}^{(t)}_{j_1} \circ \boldsymbol{v}^{(t)}_{j_2} \circ \cdots \circ \boldsymbol{v}^{(t)}_{j_t}), \tag{3}$$

where $\boldsymbol{v}^{(t)}_{j_1}, \ldots, \boldsymbol{v}^{(t)}_{j_t} \in \mathbb{R}^K$ are the t^{th}-order embeddings of feature $j_1, \ldots, j_t$, respectively. Note that HOFM is equivalent to FM if $H = 2$.

3. Pairwise feature interaction selection

In this section, we introduce our proposed method for pairwise FIS. Existing FIS-based methods, including SparseFMs, overwhelmingly select a common set of interactions for all users in a non-personalized manner, on the assumption that the same feature interactions are equally powerful to predict any user's behavior. This assumption may not be valid, as it overlooks the personal characteristics of users' behavior, especially for recommendation tasks. To this end, we account for the individuality in FIS by investigating *personalized feature interaction selection* (P-FIS) [8]. Unlike FIS, personalized feature interaction selection (P-FIS) aims to achieve adaptive FIS for each individual user. Although we can train a particular sparse factorization machine for each user, this is problematic for at least two reasons: (1) it is both time and storage inefficient since we would need to maintain a model for each user; and (2) it is less effective because estimating a model separately for each user fails to take advantage of collaborative filtering. Therefore, we propose a *Bayesian personalized feature interaction selection* (BP-FIS) mechanism for FMs.

In the following, we describe *Bayesian personalized feature interaction selection* (BP-FIS) for personalized feature interaction selection (P-FIS) on FMs with details. We introduce personalized factorization machine (PFM) with personalized feature interaction weights and we propose Bayesian personalized feature interaction selection (BP-FIS) in Section 3.1. For the efficiency of training, we first conduct variational inference in Section 3.2 and devise an efficient optimization algorithm in Section 3.3.

3.1. The proposed model

3.1.1. Personalized factorization machine

To incorporate P-FIS for FMs, we reformulate Eq. (1) as a personalized FM via feature interaction parameters:

$$\hat{y}_{PFM}(\boldsymbol{x}) = w_u + \sum_{i=1}^{D} w_{ui} x_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} w_{uij} x_i x_j, \tag{4}$$

where w_u, w_{ui}, w_{uij} are the personalized coefficients of user u; w_{ui} and w_{uij} reflect the preferences of user u over the first- and second-order feature interactions. While the FM in Eq. (1) can also account for personalization by taking user ID as features, it fails to personalize every interaction, as required by P-FIS.

3.1.2. Personalized feature interaction selection

Next, we recall the definition of BP-FIS by applying Bayesian variable selection (BVS) to feature interaction selection. BVS attempts to estimate the marginal posterior of a variable as the probability that the variable should be selected. Various BVS methods have been proposed [15]; spike-and-slab priors (SSPs) [70] have been widely studied. A variable w following SSPs is sampled from a linear combination of two distributions:

$$w \sim \pi \mathcal{N}(\mu, \sigma^2) + (1 - \pi)\delta_0,$$

where $\mathcal{N}(\mu, \sigma^2)$ is the slab prior, which is modeled as a Gaussian distribution with mean μ and variance σ^2; δ_0 is the spike prior, which is modeled using a Dirac delta mass function centered at zero. The Dirac delta function is a generalized distribution used to model the density of an idealized point mass as a function equal to zero everywhere except for zero and whose integral over the entire real line is equal to one. SSPs assign non-zero probability for the event $w = 0$ ($P(w = 0) = 1 - \pi$). Therefore, SSP is the ideal distribution for variable selection. However, the presence of the Dirac delta function δ_0 in the SSPs complicates inference. SSPs have been reformulated by Titsias and Lázaro-Gredilla [64] in the following manner:

$$s \sim Bernoulli(\pi), \quad \tilde{w} \sim \mathcal{N}(0, 1), \quad w = \tilde{w} \cdot s. \tag{5}$$

This introduces two additional variables, $\tilde{w}$ and s, where $\tilde{w}$ represents the weight of the variable and s indicates whether to select the variable. The SSP in Eq. (5) is amenable to approximate inference [64].

Based on the discussion above, we formulate *Bayesian personalized feature interaction selection* (BP-FIS). A graphical model of BP-FIS is shown in Fig. 1. Each rating $y(\boldsymbol{x})$ of feature $\boldsymbol{x}$ in the dataset is generated by the procedure that is detailed in Algorithm 1. Here, we use $\mathcal{F}, \mathcal{U}, \mathcal{X}$ and $\mathcal{Y}$ to denote the set of features, the set of users, the set of feature vectors, and the set of labels, respectively. We define $M = |\mathcal{U}|$, as the number of users; $N = |\mathcal{X}| = |\mathcal{Y}|$ as the number of data samples.

In the first part of the algorithm (line 1–10), we generate personalized feature interaction weights w_{ui} and w_{uij} for all users. We reformulate w_{ui} by $s_{ui} \cdot \tilde{w}_i$ and w_{uij} by $s_{uij} \cdot \tilde{w}_{ij}$ as suggested by [64], so that we can take advantage of collaborative filtering by learning a single set of feature interaction weights $\tilde{W} = \{\tilde{w}_i\} \cup \{\tilde{w}_{ij}\}$ for all users and operationalize BP-FIS by learning the personalized interaction selection variable $S = \{s_{ui}\} \cup \{s_{uij}\}$. The generation of s_{uij} is conditioned on s_{ui} and s_{uj}, which captures the hereditary relation between the first- and second-order interactions; see Section 3.1.3 for details on s_{ui} and s_{uij}.

In the second part of Algorithm 1 (line 11–13), we calculate the rating prediction by Eq. (4) and generate the rating $y(\boldsymbol{x})$, i.e., the true label of $\boldsymbol{x}$ following $P(y \mid \hat{y}(\boldsymbol{x}))$.

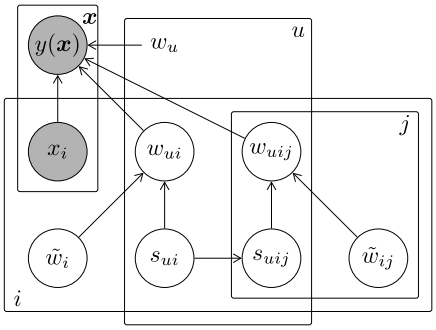

Fig. 1. Graphical model of Bayesian personalized feature interaction selection (BP-FIS). Nodes represent random variables and edges represent dependencies between variables.

Algorithm 1: Generation procedure of BP-FIS.

1 **for** each feature $i \in \mathcal{F}$ **do**
2 draw 1st-order interaction weight $\tilde{w}_i \sim \mathcal{N}(0, 1)$;
3 **for** each user $u \in \mathcal{U}$ **do**
4 draw 1st-order interaction selection $s_{ui} \sim Bernoulli(\pi_1)$;
5 $w_{ui} = s_{ui} \cdot \tilde{w}_i$;
6 **for** each feature pair $i, j \in \mathcal{F}$ **do**
7 draw 2nd-order interaction weight $\tilde{w}_{ij} \sim \mathcal{N}(0, 1)$;
8 **for** each user $u \in \mathcal{U}$ **do**
9 draw 2nd-order interaction selection $s_{uij} \sim P(s_{uij} \mid s_{ui}, s_{uj}, \pi_2)$;
10 $w_{uij} = s_{uij} \cdot \tilde{w}_{ij}$;
11 **for** each feature vector $\boldsymbol{x} \in \mathcal{X}$ **do**
12 calculate the rating prediction $\hat{y}(\boldsymbol{x})$ by Eq. (4);
13 draw $y(\boldsymbol{x}) \sim P(y \mid \hat{y}(\boldsymbol{x}))$;

The distribution of $y(\boldsymbol{x})$ determines the likelihood function for optimization. For example, if we assume $y(\boldsymbol{x})$ to follow a Gaussian distribution $\mathcal{N}(\hat{y}(\boldsymbol{x}), 1)$, we can derive the square loss:

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2} (y(\boldsymbol{x}) - \hat{y}(\boldsymbol{x}))^2. \tag{6}$$

If $y(\boldsymbol{x})$ follows a Bernoulli distribution $Bernoulli(p(\boldsymbol{x}))$, where $p(\boldsymbol{x}) = \text{sigmoid}(\hat{y}(\boldsymbol{x}))$, the logistic log-likelihood can be derived:

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \left(y(\boldsymbol{x}) \log p(\boldsymbol{x}) + (1 - y(\boldsymbol{x})) \log (1 - p(\boldsymbol{x})) \right). \tag{7}$$

The likelihood functions in Eq. (6) and (7) correspond to the squared loss and cross entropy loss, which are widely used in collaborative filtering [36]. Besides, a ranking loss can also be derived, e.g., a pairwise ranking loss [53] or a listwise ranking loss [76]. However, deriving the proper likelihood function for optimization is beyond the scope of this paper. In this work, we employ the Gaussian likelihood of Eq. (6) for optimization.

3.1.3. Hereditary spike-and-slab priors

Although we learn the personalized parameters in Eq. (4) for each user separately, there are two problems. On the one hand, selecting interactions directly based on Eq. (4) fails to take advantage of collaborative filtering. On the other hand, Eq. (4) raises optimization challenges due to the large number of parameters ($\mathcal{O}(MD^2)$).

To address the above problems, we reformulate w_{ui} as $s_{ui} \cdot \tilde{w}_i$ (line 5) and w_{uij} as $s_{uij} \cdot \tilde{w}_{ij}$ (line 10) in the generation procedure in Section 3.1. In order to model s_{ui} and s_{uij}, we propose the *hereditary spike-and-slab prior* (HSSP), which extends SSP by capturing heredity constraints [14] between first- and second-order interactions. The underlying intuition is that there are natural hereditary constraints among the first- and second-order interactions [41], i.e., feature i and feature j are the "parents" of feature interaction $\langle i, j \rangle$. The hereditary constraints help to dramatically decrease the number of candidate interactions.

The hereditary spike-and-slab prior (HSSP) is based on two hereditary constraints: *strong heredity* and *weak heredity*. Based on FMs, we define them as follows:

Definition 1 *(Strong heredity). Given a FM,* strong heredity *is the constraint that if the first-order interactions i and j are selected for the FM, the second-order interaction of their combination, i.e., $\langle i, j\rangle$ will also be selected.*

According to *strong heredity*, we have:

$$\text{if } s_{ui} = 1 \text{ or } s_{uj} = 1 \text{ then } s_{uij} = 1.$$

Definition 2 *(Weak heredity). Given a FM,* weak heredity *is the constraint that if one of the first-order interactions i or j is selected for FM, then the second-order interaction of their combination $\langle i, j\rangle$ will have a non-zero probability to be selected.*

According to *weak heredity*, we have:

$$\text{if } s_{ui} = 1 \text{ or } s_{uj} = 1 \text{ then } P(s_{uij} = 1) > 0.$$

Based on the definitions of *strong heredity* and *weak heredity*, we derive the HSSP by modifying the priors for s_{uij} of SSP:

$$\begin{aligned}
&P(s_{ui} = 1) = P(s_{uj} = 1) = \pi_1 \\
&P(s_{uij} = 1 \mid s_{ui}s_{uj} = 1) = 1 && \textit{(Strong heredity)} \\
&P(s_{uij} = 1 \mid s_{ui} + s_{uj} = 1) = \pi_2 && \textit{(Weak heredity)} \\
&P(s_{uij} = 1 \mid s_{ui} + s_{uj} = 0) = 0,
\end{aligned}$$

where $\pi_1, \pi_2 \in [0, 1]$ are constant values.

Note that we assume s_{ui} follow Bernoulli(π_1). One may argue that since π_1 is a constant, it fails to reveal the personalization of user u. It is worth noting that Bernoulli(π_1) is the prior of s_{ui}. Since we presume no prior knowledge of user's selection preference, we assume equal probability π_i for all users. The posterior of s_{ui}, i.e. $p(s_{ui} \mid \mathcal{X})$, is different for all users, which reflects the personalization of selection.

3.2. Variational inference

To optimize BP-FIS, we maximize the posterior $P(\tilde{W}, S \mid \mathcal{X}, \mathcal{Y})$, such that $S = \{s_{ui}\} \cup \{s_{uij}\}$ and $\tilde{W} = \{\tilde{w}_i\} \cup \{\tilde{w}_{ij}\}$. However, exact inference for $P(\tilde{W}, S \mid \mathcal{X}, \mathcal{Y})$ requires an infeasible combinatorial search over $\mathcal{O}(2^{MD^2})$ possible models. To speed up the process, we conduct variational inference to approximate the posterior $P(\tilde{W}, S \mid \mathcal{X}, \mathcal{Y})$ by a variational distribution $Q(\tilde{W}, S)$. The closeness between the posterior and the variational distribution is measured by Kullback–Leibler (KL)-divergence, i.e., $\mathbb{KL}(Q(\tilde{W}, S) \parallel P(\tilde{W}, S \mid \mathcal{X}, \mathcal{Y}))$. The KL-divergence can be derived as follows:

$$\begin{aligned}
&\mathbb{KL}(Q(\tilde{W}, S) \parallel P(\tilde{W}, S \mid \mathcal{X}, \mathcal{Y})) \\
&\quad = -\mathbb{E}_Q\left[\log P(\tilde{W}, S, \mathcal{X}, \mathcal{Y}) - \log Q(\tilde{W}, S)\right] + \log P(\mathcal{X}, \mathcal{Y}),
\end{aligned} \tag{8}$$

where $\mathcal{L} = \mathbb{E}_Q\left[\log P(\tilde{W}, S, \mathcal{X}, \mathcal{Y}) - \log Q(\tilde{W}, S)\right]$ is the evidence lower bound (ELBO); $\log P(\mathcal{R}, \mathcal{X})$ is the marginal likelihood that does not depend on $Q(\tilde{W}, S)$. Eq. (8) indicates that minimizing the KL-divergence is the same as maximizing the ELBO. To maximize the ELBO, we first discuss the variational distribution $Q(\tilde{W}, S)$.

Variational distribution To ensure the quality of approximation, we assume that the hereditary constraints also hold in the variational distributions. Therefore, $Q(\tilde{W}, S)$ can be factorized as follows:

$$Q(\tilde{W}, S) = \prod_{i=1}^{D}\prod_{j=i+1}^{D} Q(\tilde{w}_i)Q(\tilde{w}_{ij}) \prod_{u\in\mathcal{U}} Q(s_{ui})Q(s_{uij} \mid s_{ui}, s_{uj}). \tag{9}$$

The factorized distributions are formulated as follows:

$$\begin{aligned}
Q(w_i \mid \mu_i, \sigma_i) &= \mathcal{N}(\mu_i, \sigma_i) \\
Q(w_{ij} \mid \mu_{ij}, \sigma_{ij}) &= \mathcal{N}(\mu_{ij}, \sigma_{ij}) \\
Q(s_{ui} \mid \pi_{ui}) &= \textit{Bernoulli}(\pi_{ui}) \\
Q(s_{uij} \mid s_{ui}s_{uj} = 1) &= s_{uij} \\
Q(s_{uij} \mid s_{ui} + s_{uj} = 1, \pi_{uij}) &= \textit{Bernoulli}(\pi_{uij}) \\
Q(s_{uij} \mid s_{ui} + s_{uj} = 0) &= 1 - s_{uij},
\end{aligned} \tag{10}$$

where $\rho = \{\mu_i, \sigma_i\} \cup \{\mu_{ij}, \sigma_{ij}\}$ and $\pi = \{\pi_{ui}\} \cup \{\pi_{uij}\}$ are the variational parameters.

Evidence lower bound Given the variational distribution $Q(\tilde{W}, S \mid \rho, \pi)$, the ELBO can be derived as follows:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_Q\left[\log P(\tilde{W}, S, \mathcal{X}, \mathcal{Y}) - \log Q(\tilde{W}, S \mid \rho, \pi)\right] \\ &= \sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_Q\left[\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}))\right] - \mathbb{KL}\left(Q(\tilde{W}, S \mid \rho, \pi) \parallel P(\tilde{W}, S)\right),\end{aligned}$$

where $\mathbb{E}_Q[\cdot]$ stands for the expectation w.r.t. $Q(\tilde{W}, S \mid \rho, \pi)$. In the ELBO, $\mathbb{KL}(Q(\tilde{W}, S \mid \rho, \pi) \parallel P(\tilde{W}, S))$ can be analytically derived:

$$\begin{aligned}&\mathbb{KL}(Q(\tilde{W}, S \mid \rho, \pi) \parallel P(\tilde{W}, S)) = \\ &\quad\sum_{i=1}^{D} \mathbb{KL}\left(Q(\tilde{w}_i) \parallel Q(\tilde{w}_i)\right) \sum_{u=1}^{D} \mathbb{KL}\left(Q(s_{ui}) \parallel P(s_{ui})\right) + \\ &\quad\sum_{i=1}^{D}\sum_{j=i+1}^{D} \mathbb{KL}\left(Q(\tilde{w}_{ij}) \parallel Q(\tilde{w}_{ij})\right) \sum_{u=1}^{M} \mathbb{KL}\left(Q(s_{uij} \mid s_{ui}, s_{uj}) \parallel P(s_{uij} \mid s_{ui}, s_{uj})\right),\end{aligned}$$

where

$$\begin{aligned}\mathbb{KL}\left(Q(\tilde{w}_i) \parallel P(\tilde{w}_i)\right) &= \frac{1}{2}\left(1 + \log\sigma_i^2 - \mu_i^2 - \sigma_i^2\right) \\ \mathbb{KL}\left(Q(\tilde{w}_{ij}) \parallel P(\tilde{w}_{ij})\right) &= \frac{1}{2}\left(1 + \log\sigma_{ij}^2 - \mu_{ij}^2 - \sigma_{ij}^2\right) \\ \mathbb{KL}\left(Q(s_{ui}) \parallel P(s_{ui})\right) &= \pi_{ui}\log\frac{\pi_{ui}}{\pi_1} + (1-\pi_{ui})\log\frac{1-\pi_{ui}}{1-\pi_1}\end{aligned} \tag{11}$$

and

$$\begin{aligned}&\mathbb{KL}\left(Q(s_{uij} \mid s_{ui}, s_{uj}) \parallel P(s_{uij} \mid s_{ui}, s_{uj})\right) = \\ &\quad(\pi_{ui} + \pi_{uj} - 2\pi_{uij})\left(\pi_{uij}\log\frac{\pi_{uij}}{\pi_2} + (1-\pi_{uij})\log\frac{1-\pi_{uij}}{1-\pi_2}\right).\end{aligned} \tag{12}$$

However, it is problematic to derive $\mathbb{E}_Q\left[\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}))\right]$. To address this, we approximate the expectation with Monte Carlo estimation as follows:

$$\sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_Q\left[\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}))\right] \approx \frac{1}{L}\sum_{l=1}^{L}\sum_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{2}\left(y(\boldsymbol{x}) - \hat{y}(\boldsymbol{x})^{(l)}\right)^2, \tag{13}$$

where $\hat{y}(\boldsymbol{x})^{(l)}$ is calculated by Eq. (4) with the l-th sampling $W^{(l)}$. We write $\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x})^{(l)})$ as a Gaussian log-likelihood as we assume $y(\boldsymbol{x})$ to follow $\mathcal{N}(\hat{y}(\boldsymbol{x}), 1)$ (Eq. (6) in Section 3.1).

Reparameterization When sampling w_{ui} and w_{uij}, we apply the reparameterization trick [28]:

$$\begin{aligned}&\epsilon_1, \epsilon_2 \sim \mathit{Uniform}(0,1), \quad \varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0,1) \\ &s_{ui} = [\![\epsilon_1 \geq \pi_{ui}]\!] \\ &s_{uij} = s_{ui}s_{uj} + (1 - s_{ui}s_{uj})(s_{ui} + s_{uj})[\![\epsilon_2 \geq \pi_{uij}]\!] \\ &\tilde{w}_i = \mu_i + \varepsilon_1\sigma_i \\ &\tilde{w}_{ij} = \mu_{ij} + \varepsilon_2\sigma_{ij} \\ &w_{ui} = \tilde{w}_i \cdot s_{ui} \\ &w_{uij} = \tilde{w}_{ij} \cdot s_{uij}.\end{aligned} \tag{14}$$

In this way, the stochasticity in the sampling process is isolated, while the gradient with respect to $\rho = \{\mu_i, \sigma_i\} \cup \{\mu_{ij}, \sigma_{ij}\}$ is back-propagated through the sampled w_{ui} and w_{uij}. Unfortunately, the above procedure fails to take the gradient of $\pi = \{\pi_{ui}\} \cup \{\pi_{uij}\}$ due to the discrete nature of $S = \{s_{ui}\} \cup \{s_{uij}\}$. We follow [65] to take gradients specifically for discrete variables. To take the gradient for π_{ui}, we marginalize out π_{ui}:

$$\begin{aligned}\mathbb{E}_Q\left[\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}))\right] = \pi_{ui}&\mathbb{E}_{Q\setminus\{s_{ui}\}}\left[\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}), s_{ui} = 1)\right] + \\ &(1-\pi_{ui})\mathbb{E}_{Q\setminus\{s_{ui}\}}\left[\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}), s_{ui} = 0)\right].\end{aligned}$$

The gradient of $\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x})^{(l)} \setminus \{\pi_{ui}\})$ over π_{ui} is:

$$\begin{aligned}\nabla_{\pi_{ui}} &= \mathbb{E}_{Q\setminus\{s_{ui}\}}\left[\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}), s_{ui}=1)\right] - \mathbb{E}_{Q\setminus\{s_{ui}\}}\left[\log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}), s_{ui}=0)\right] \\ &= \frac{1}{L}\sum_{l=1}^{L} \log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x})^{(l)}, s_{ui}=1) - \log P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x})^{(l)}, s_{ui}=0).\end{aligned}$$

The gradient of π_{uij} can be computed in a similar way.

3.3. Optimization

To further speed up the variational inference, we factorize the variational parameter π_{uij} as $\pi_{ui} \cdot \pi_{uj}$. We only need to preserve $\{\pi_{ui}\}$ for each user, decreasing the learning parameters from $\mathcal{O}(MD^2)$ to $\mathcal{O}(MD)$. For $\{\tilde{w}_{ij}\}$, we introduce feature embeddings $V \in \mathbb{R}^{D\times K}$ and replace the variational parameters for $\tilde{w}_{ij}$ as follows:

$$\mu_{ij} = \mu(\boldsymbol{v}_i, \boldsymbol{v}_j) \text{ and } \sigma_{ij} = \sigma(\boldsymbol{v}_i, \boldsymbol{v}_j),$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ are the transformation functions, and $\boldsymbol{v}_i, \boldsymbol{v}_j \in \mathbb{R}^K$ are the embeddings for feature i, j, respectively. Note that we can have different definitions for $\mu(\cdot)$ and $\sigma(\cdot)$. Inspired by [20,72], we define the transformations as follows:

$$\mu(\boldsymbol{v}_i, \boldsymbol{v_j}) = \boldsymbol{\mu}^T f_\phi(\boldsymbol{v}_i \circ \boldsymbol{v_j}) \text{ and } \sigma(\boldsymbol{v}_i, \boldsymbol{v_j}) = \boldsymbol{\sigma}^T f_\phi(\boldsymbol{v}_i \circ \boldsymbol{v_j}), \tag{15}$$

where $\circ$ is the element-wise product operation; and $f_\phi(\cdot)$ is the transformation parameterized by ϕ. Note that $\mu(\cdot)$ and $\sigma(\cdot)$ can be either linear transformations, e.g., $f_\phi(\boldsymbol{v}) = \boldsymbol{v}$, or non-linear transformations, e.g., $f_\phi(\cdot)$ is a neural network. The variational parameter for $\tilde{W}$ is $\rho = \{\phi, \boldsymbol{\mu}, \boldsymbol{\sigma}, V\}$.

Learning As the reparameterization procedures for estimating gradients of ρ and π are different, we propose to optimize the variational parameter π and ρ alternatively. Specifically, at iteration t, by fixing $\rho^{(t-1)}$, we train $\pi^{(t)}$ to update the probabilities that a user will select the specific feature interactions. Then, by fixing $\pi^{(t)}$, rather than training $\rho^{(t)}$ based on $\rho^{(t-1)}$, we train $\rho^{(t)}$ from scratch. This is because when π is updated, we have different user preferences of feature interactions, where ρ should be optimized accordingly. According to empirical studies, adapting ρ fitting for $\pi^{(t-1)}$ to $\pi^{(t)}$ could adversely bias the learning.

Prediction Once the model has been trained, we can generate predictions via point estimation in Eq. (16):

$$\mathbb{E}\left[\hat{y}(\boldsymbol{x})\right] = w_u + \sum_{i=1}^{D} \mathbb{E}\left[w_{ui}\right] x_i + \sum_{i=1}^{D}\sum_{j=i+1}^{D} \mathbb{E}\left[w_{uij}\right] x_i x_j, \tag{16}$$

where $\mathbb{E}\left[w_{ui}\right] = \pi_{ui}\mu_i$ and $\mathbb{E}\left[w_{uij}\right] = [\pi_{ui}\pi_{uj} + (1-\pi_{ui}\pi_{uj})(\pi_{ui}+\pi_{uj})\pi_{ui}\pi_{uj}]\mu_{ij}$.

4. Higher-order feature interaction selection

Beyond recommendation, FMs are also widely applied in other sparse predictive analytics, where higher-order feature interactions are considered. Since the number of feature interactions grows exponentially with the order, to harness HOFMs, efficient algorithms have been proposed to enumerate feature interactions [4,29]. Even though enumerations may be efficient, they can be time-consuming in practice. Therefore, higher-order feature interactions are approximated by neural networks (NNs) [18,35]. Existing NN-based approximations fail to distinguish useful from harmful feature interactions. They can only approximate the prediction with a combination of all feature interactions. Although attention modules are now being employed [59,62,72], a recent study has shown that attention weights may be inconsistent with the importance of features [21,57]. AutoFIS [38] and AutoHash [74] are proposed recently to perform higher-order interaction selection. However, they actually perform field interaction selection rather than feature interaction selection. Please note that there are large number of features compared to a limited number of fields (on the Avazu dataset, #feature=1,544,488 and #field=22). We target FIS for HOFMs – our goal is to filter out irrelevant higher-order feature interactions, explicitly and systematically.

In this section, we propose *Bayesian higher-order feature interaction selection* (BH-FIS) that extends the idea of FIS to higher-order factorization machines (HOFMs). Different from BP-FIS, we do not consider the personalization of selection for higher-order feature interactions. Models that consider higher-order feature interactions are mostly designed for prediction tasks [18,20,49], where personalization is not necessary. However, it is straight-forward to combine BP-FIS with Bayesian higher-order feature interaction selection (BH-FIS) to select both personalized and higher-order feature interactions.

BH-FIS is fundamentally different from existing FIS methods:

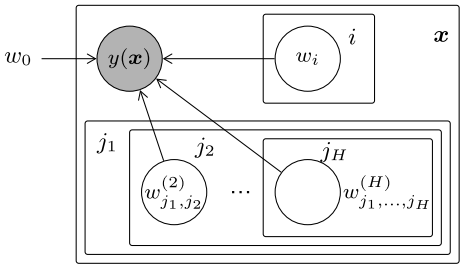

Fig. 2. Graphical representation of H-FIS: the generation of $y(\boldsymbol{x})$ is conditioned on the global bias w_0, first-order coefficients $\{w_i\}$ and higher-order feature interactions $\{w^{(t)}_{j_1,\ldots,j_t}\}$, (for $t = 2, \ldots, H$).

- *Independence*: in BH-FIS we apply BVS to select feature interactions, which is then factorized into separate variables. Therefore, the interaction selection is independent with the learning of interaction weights. The independence ensures the effect of interaction selection. In comparison, in both AutoFIS and AutoHash, the selection is deeply coupled with the learning;
- *Discreteness*: BH-FIS introduced SSP to select feature interactions. SSP performs the true feature selection by assigning non-zero probability to the event that a variable equals zero. In contrast, most other methods uses sparsity priors for variable selection that assigns zero probability to the event. Therefore, SSP involves discrete variables, whereas the selection variables in both AutoFIS and AutoHash are not discrete: AutoFIS achieves selection by learning the importance of interactions with generalized regularized dual averaging (GRDA). The selection is achieved by sparsity regularizations (ℓ_1-norm), which is similar to SparseFM; AutoHash uses Gumbel-softmax trick to approximate the discrete variables. The gradients are unable to back-propagate due to the involvement of discrete variables. Thus, the learning of BH-FIS is harder than AutoFIS and AutoHash.

In the following, we propose BH-FIS in Section 4.1. To ensure the efficiency, we conduct variational inference, where the coefficient is reparameterized with factored parameters (Section 4.2). We provide a systematic way to enumerate higher-order feature interactions comprehensively and non-repetitively and then optimize the model via inference networks (Section 4.3).

4.1. The proposed model

Multi-field inputs To enable learning with multi-field input features, an embedding layer is applied given D feature embeddings [35]. With $\boldsymbol{x}$ as input, the embedding layer will output $|\mathcal{F}|$ embeddings, each of which is the representation of a field: for a univalent field, the corresponding feature embedding is taken as the field representation; for a multivalent field, the sum of feature embeddings is generated.

High-order feature interaction selection To perform FIS on HOFM, we propose *Bayesian higher-order feature interaction selection* (BH-FIS), i.e., a Bayesian variable selection (BVS) method [2,17,43]. The Bayesian generative model of BH-FIS is illustrated in Fig. 2. We explain the model with more details, where each $w^{(t)}_{j_1,\ldots,j_t}$ $(t \geq 2)$ is associated with a spike-and-slab prior (SSP) [70] for FIS:

$$w^{(t)}_{j_1,\ldots,j_t} \sim \pi^{(t)} \mathcal{N}(0, \lambda^{-1}) + (1 - \pi^{(t)}) \delta_0, \tag{17}$$

where $\mathcal{N}(0, \lambda^{-1})$ is the slab prior, modeled as a zero-centered Gaussian distribution; δ_0 is the spike prior, with a zero-centered Dirac delta mass function; and $\pi^{(t)} \in (0, 1)$ is a hyper-parameter to indicate the probability that the variable $w^{(t)}_{j_1,\ldots,j_t}$ will be selected. Compared with sparsity priors [3], SSP assigns non-zero probability $1 - \pi^{(t)}$ to the event that $w^{(t)}_{j_1,\ldots,j_t} = 0$. We consider w_0 as a parameter and assign Gaussian distribution as the prior for w_i to avoid over-fitting:

$$w_i \sim \mathcal{N}(0, \lambda^{-1}).$$

The distribution of $y(\boldsymbol{x})$, conditioned on $\hat{y}(\boldsymbol{x})$, i.e., $P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}))$, determines the loss function for optimization. For example, if we assume a Gaussian distribution, the square loss can be derived; otherwise, if we assume a Bernoulli distribution, cross entropy loss can be formulated. Since deriving the proper loss function for optimization is domain-specific, we refer to the discussion of $P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}))$ in the experiments (Section 5.4).

While H-FIS selects the higher-order feature interactions, it is challenging to perform inference, due to the following: (1) there are a large number of feature interactions, especially when interactions are higher-order; and (2) the SSP further complicates the problem due to the presence of the Dirac delta function δ_0. Therefore, we conduct variational inference to address the issue, as discussed in Section 4.2.

4.2. Variational inference

To solve the Dirac delta function δ_0 in Eq. (17), we follow [64] to reformulate the SSPs as follows:

$$
\begin{aligned}
w^{(t)}_{j_1,\dots,j_t} &= \tilde{w}^{(t)}_{j_1,\dots,j_t} \cdot s_{j_1,\dots,j_t} \\
\tilde{w}^{(t)}_{j_1,\dots,j_t} &\sim \mathcal{N}(0, \lambda^{-1}) \\
s_{j_1,\dots,j_t} &\sim \mathit{Bernoulli}(\pi^{(t)}),
\end{aligned}
$$

where two additional variables are introduced: $\tilde{w}^{(t)}_{j_1,\dots,j_t}$ measures the weight of $\langle j_1, \dots, j_t \rangle$; $s_{j_1,\dots,j_t}$ determines whether to select $\langle j_1, \dots, j_t \rangle$. The superscript (t) is maintained for $\tilde{w}^{(t)}_{j_1,\dots,j_t}$ to indicate that different orders of feature interactions are modeled respectively and removed for $s_{j_1,\dots,j_t}$ to show that the hereditary relations between the selection of different orders of feature interactions are captured (as discussed in Section 4.2.1).

Exact inference for the model is infeasible due to the large number of variables. Therefore, we conduct variational inference. In particular, we approximate the posterior distributions $P(\tilde{w}^{(t)}_{j_1,\dots,j_t} \mid \mathcal{X}, \mathcal{Y})$ and $P(s_{j_1,\dots,j_t} \mid \mathcal{X}, \mathcal{Y})$ by variational distributions $Q(\tilde{w}^{(t)}_{j_1,\dots,j_t})$ and $Q(s_{j_1,\dots,j_t})$, respectively. The essential component of variational inference lies in the definition of variational distributions, which will be discussed below.

4.2.1. Variational distributions

Following the mean-field assumption, we define $Q(\tilde{w}^{(t)}_{j_1,\dots,j_t})$ as a Gaussian distribution with respective parameters:

$$
Q(\tilde{w}^{(t)}_{j_1,\dots,j_t}) = \mathcal{N}(\mu^{(t)}_{j_1,\dots,j_t}, \sigma^{(t)}_{j_1,\dots,j_t}).
$$

In order to reduce the parameter size, we further factorize $\mu^{(t)}_{j_1,\dots,j_t}$ and $\sigma^{(t)}_{j_1,\dots,j_t}$:

$$
\mu^{(t)}_{j_1,\dots,j_t} = f_\mu(\boldsymbol{v}^{(t)}_{j_1} \circ \cdots \circ \boldsymbol{v}^{(t)}_{j_t}) \tag{18}
$$

$$
\sigma^{(t)}_{j_1,\dots,j_t} = f_\sigma(\boldsymbol{v}^{(t)}_{j_1} \circ \cdots \circ \boldsymbol{v}^{(t)}_{j_t}), \tag{19}
$$

where $\circ$ is the element-wise product between vectors; $f_\mu(\cdot)$ and $f_\sigma(\cdot)$ are functions parameterized by μ and σ that transform a vector from $\mathbb{R}^K$ to $\mathbb{R}$.

While we can define $Q(s_{j_1,\dots,j_t})$ similarly by respective variational parameters, the definition fails to capture *hereditary relations*.

Heredity has been shown to be effective [8]. We argue that heredity should also exist among higher-order feature interactions, e.g., the selection of $\langle j_1, \dots, j_t \rangle$ should depend on the selection of $\langle j'_1, \dots, j'_{t-1} \rangle$ where $j'_1, \dots, j'_{t-1} \in \{j_1, \dots, j_t\}$. However, the positive relation between the selection of first- and second-order interactions introduced in Section 3 does not necessarily hold in higher-order scenarios, where negative relations may also exist. For example, it may be unnecessary to preserve the interaction $\langle \text{user} = \mathit{Alice}, \text{movie} = \mathit{Titanic}, \text{genre} = \mathit{Romance}, \text{tag} = \mathit{Love\ Story} \rangle$ if $\langle \text{user} = \mathit{Alice}, \text{movie} = \mathit{Titanic}, \text{genre} = \mathit{Romance} \rangle$ is selected since they carry similar semantics.

Motivated by the dependency between $\langle j_1, \dots, j_t \rangle$ and $\{\langle j'_1, \dots, j'_{t-1} \rangle \mid j'_1, \dots, j'_{t-1} \in \{j_1, \dots, j_t\}\}$ (both positive and negative), we define $Q(s_{j_1,\dots,j_t})$ as a Bernoulli distribution with respective parameters:

$$
Q(s_{j_1,\dots,j_t}) = \mathit{Bernoulli}(\pi_{j_1,\dots,j_t}),
$$

where $\pi_{j_1,\dots,j_t}$ is factorized by shared embeddings:

$$
\pi_{j_1,\dots,j_t} = \text{sigmoid}\left(f_\phi(\boldsymbol{z}_{j_1} \circ \cdots \circ \boldsymbol{z}_{j_t})\right). \tag{20}
$$

Similarly, $f_\phi(\cdot)$[2] is a function parameterized by ϕ that transforms a vector from $\mathbb{R}^K$ to $\mathbb{R}$. It can be seen that this definition captures the heredity in higher-order feature interactions. The dependency between $\langle j_1, \dots, j_t \rangle$ and $\langle j'_1, \dots, j'_{t-1} \rangle$ $(\forall j'_1, \dots, j'_{t-1} \in \{j_1, \dots, j_t\})$ is captured since $Q(s_{j_1,\dots,j_t})$ and $Q(s_{j'_1,\dots,j'_{t-1}})$ are conditioned on the same embeddings $\boldsymbol{z}_{j'_1}, \dots, \boldsymbol{z}_{j'_{t-1}}$.

The above defines the inference model for BH-FIS as summarized in Fig. 3, where we define $V^{(t)} = \{\boldsymbol{v}^{(t)}_1, \dots, \boldsymbol{v}^{(t)}_D\}$ and $Z = \{\boldsymbol{z}_1, \dots, \boldsymbol{z}_D\}$.

[2] We devise inference networks to instantiate f_μ, f_σ and f_ϕ in Section 4.3.1.

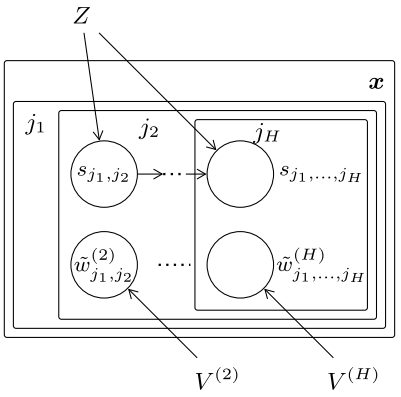

Fig. 3. Graphical representation of inference model: $w^{(t)}_{j_1,\ldots,j_t}$ is reformulated into $\tilde{w}^{(t)}_{j_1,\ldots,j_t}$ and $s_{j_1,\ldots,j_t}$. Heredity is captured by the dependency between $s_{j_1,\ldots,j_t}$ and $s_{j_1,\ldots,j_{t-1}}$. $\tilde{w}^{(t)}_{j_1,\ldots,j_t}$ is conditioned on separate feature embedding $V^{(t)}$ while $s_{j_1,\ldots,j_t}$ is conditioned on shared embedding Z.

4.2.2. Evidence lower bound

As previously discussed, exact inference for variational distribution is not feasible. To this end, we propose to approximate the posterior distributions, by maximizing the ELBO. For ease of presentation, we define $\Theta = \{V^{(2)}, \ldots, V^{(H)}, \mu, \sigma\}$ and $\Phi = \{Z, \phi\}$, which yields the following:

$$\mathcal{L}(Q) = \mathbb{E}_Q \left[\log P(\tilde{W}, S, \mathcal{X}, \mathcal{Y}) - \log Q_\Theta(\tilde{W}) - \log Q_\Phi(S) \right]. \tag{21}$$

In the next section, we will discuss with details on how to maximize the ELBO.

4.3. Optimization

In this section, we discuss how to optimize the ELBO in Eq. (21). Motivated by the idea of auto-encoding variational Bayes (AEVB) [28], we interpret the ELBO via a reconstruction loss and two KL-divergence losses:

$$\mathcal{L}_Q = \mathcal{L}_{Re} + \mathcal{L}_{KL}(S) + \mathcal{L}_{KL}(\tilde{W}), \tag{22}$$

where

$$\mathcal{L}_{Re} = \mathbb{E}_Q [\log P(\mathcal{Y} \mid S, \tilde{W})] \tag{23}$$

$$\mathcal{L}_{KL}(S) = \mathbb{KL}(Q_\Phi(S) \parallel P(S)) \tag{24}$$

$$\mathcal{L}_{KL}(\tilde{W}) = \mathbb{KL}(Q_\Theta(\tilde{W}) \parallel P(\tilde{W})). \tag{25}$$

The KL-divergence losses $\mathcal{L}_{KL}(S)$ and $\mathcal{L}_{KL}(\tilde{W})$ can be derived analytically. Therefore, in the following, we focus on how to derive $\mathcal{L}_{Re}$, which is further formulated below:

$$\mathcal{L}_{Re} = \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{E}_Q \left[\log P(y(\boldsymbol{x}) \mid \tilde{y}(\boldsymbol{x})) \right]. \tag{26}$$

To derive $P(y(\boldsymbol{x}) \mid \hat{y}(\boldsymbol{x}))$, we need to calculate $\hat{y}(\boldsymbol{x})$. According to Eq. (2), we infer $\tilde{W}$ and S, which will be discussed in Section 4.3.1 by devising inference networks.

4.3.1. Inference networks

Based on Eq. (18), (19) and (20), we propose inference networks: f_μ and f_σ with $V^{(2)}, \ldots, V^{(H)}$ as embeddings and f_ϕ with Z as embeddings. Note that f_μ and f_σ do not share any parameters with f_ϕ to ensure that the selection of feature interactions is independent from the learning of factorization machines. This is a fundamental difference with attention-based methods [62,72], where attention weights are not independent from the model. Given a feature interaction, we can evaluate the corresponding variable via inference networks. Fig. 4 shows how $w^{(3)}_{2,3,5}$ is inferred given the 3[rd]-order feature interaction $\langle 2, 3, 5 \rangle$.

Concretely speaking, We first reparameterize the sampling of $w^{(t)}_{j_1,\ldots,j_t}$:

$$\epsilon \sim \mathit{Uniform}(0, 1), \quad \varepsilon \sim \mathcal{N}(0, 1), \tag{27}$$

$$s_{j_1,\ldots,j_t} = [\![\epsilon \geq \pi_{j_1,\ldots,j_t}]\!], \tag{28}$$

Algorithm 2: Reparam.

Input : Feature embeddings for selection: $[\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n]$
Input : Feature embeddings for weight: $[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$
Output: The value of coefficients: $[w_1, \ldots, w_n]$

$[\pi_1, \ldots, \pi_n] \leftarrow f_\phi([\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n])$;
$[\mu_1, \ldots, \mu_n] \leftarrow f_\mu([\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n])$;
$[\sigma_1, \ldots, \sigma_n] \leftarrow f_\sigma([\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n])$;
$[\epsilon_1, \ldots, \epsilon_n] \leftarrow \mathit{Uniform}(0, 1)$;
$[\varepsilon_1, \ldots, \varepsilon_n] \leftarrow \mathcal{N}(0, 1)$;
for $i = 1, \ldots, n$ **do**
  $s_i = [\![\epsilon_i \geq \pi_i]\!]$;
  $\tilde{w}_i \leftarrow \mu_i + \sigma_k \cdot \varepsilon_i$;
  $w_i \leftarrow s_i \cdot \tilde{w}_i$;
return $[w_1, \ldots, w_n]$;

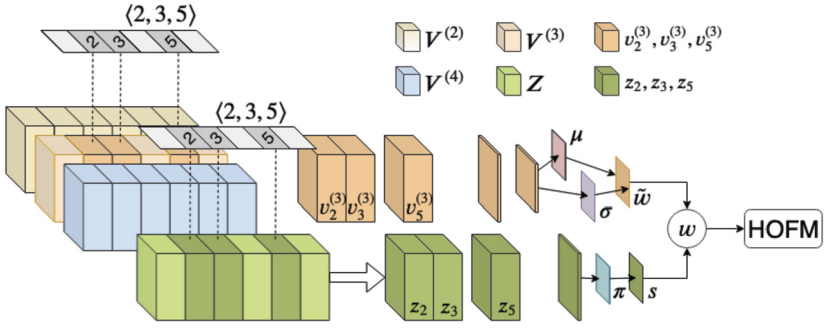

Fig. 4. An illustrative view of how inference networks infer the 3rd-order feature interaction $\langle 2, 3, 5\rangle$ (shortened to w in the figure). The feature embeddings are drawn with cuboids of different colors (refer to the legend). We deepen the color of the selected embeddings. To evaluate $w = \tilde{w} \cdot s$, two independent inference networks are designed, to generate $\tilde{w}$ and s respectively. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\tilde{w}^{(t)}_{j_1,\ldots,j_t} = \mu^{(t)}_{j_1,\ldots,j_t} + \varepsilon \cdot \sigma^{(t)}_{j_1,\ldots,j_t}, \tag{29}$$

$$w^{(t)}_{j_1,\ldots,j_t} = \tilde{w}^{(t)}_{j_1,\ldots,j_t} \cdot s_{j_1,\ldots,j_t}, \tag{30}$$

where $[\![\cdot]\!]$ is the binary operator that outputs 1 if the condition inside is met, and 0 otherwise. We summarize the reparameterization in Algorithm 2.

While we can optimize Θ via back-propagation through the inference network, we cannot back-propagate gradients to optimize Φ, due to the discrete nature of S. One straightforward solution is to rely on a Naïve Monte Carlo gradient estimator [7]. Unfortunately, this estimation suffers from large variance [46]. Inspired by [65], we propose to optimize Φ by marginalizing out the variable of interest.

To generate the prediction $\hat{y}(\boldsymbol{x})$ for the sake of deriving $\mathcal{L}_{Re}$ in Eq. (26), combining with Eq. (2), we also need a systematic way to enumerate all feature interactions, which will be discussed in the following section.

4.3.2. Feature interaction enumeration

To do FIS, a comprehensive and non-repetitive enumeration will be required. Blondel et al. [4] and Trofimov and Novikov [66] proposed efficient enumeration methods. However, they can only be applied when *all* feature interactions are evaluated. For H-FIS, we will filter out non-candidate feature interactions, while evaluating $\hat{y}(\boldsymbol{x})$ with a partially selected set of feature interactions.

One widely applied strategy is the outer-product,[3] which is easily parallelized to enumerate feature interactions [18, 35,58]. Therefore, we propose to explore the outer-product for higher-order feature interaction enumeration. However, one limitation of outer-products lies in the generation of repetitive interactions. To address this problem, we resort to masking techniques.

4.3.3. Masking

To formally discuss masking, we denote C_n^t as the number of t-combinations from n elements:

$$C_n^t = \frac{n(n-1)\cdots(n-t+1)}{t(t-1)\cdots 1}. \tag{31}$$

[3] An example of the outer-product (denoted by $\odot$) is: $\{i_1, i_2\} \odot \{j_1, j_2\} = [\langle i_1, j_1\rangle, \langle i_1, j_2\rangle; \langle i_2, j_1\rangle, \langle i_2, j_2\rangle]$.

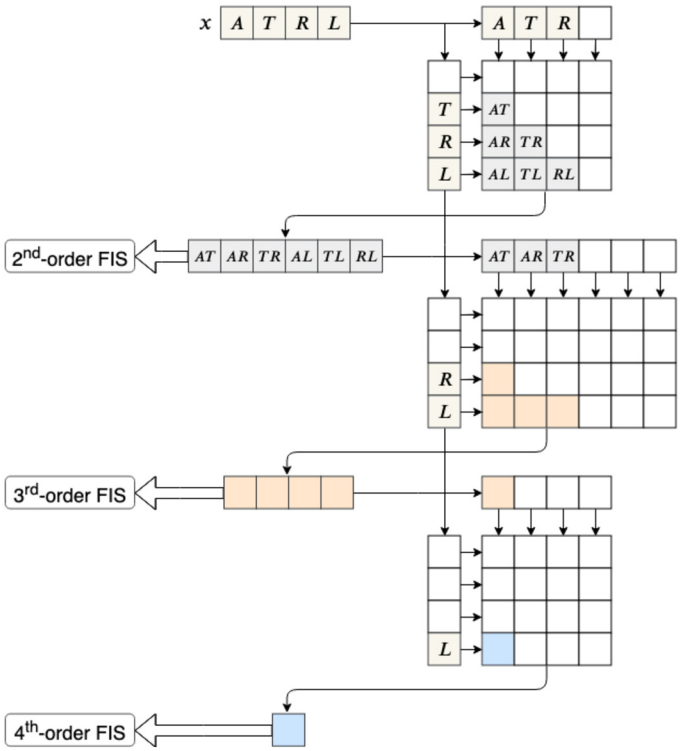

Fig. 5. An illustrative example of interaction enumeration. We abbreviate the input [user = *Alice*, movie = *Titanic*, genre = *Romance*, tag = *Love Story*] as $[A, T, R, L]$.

We define three mask matrices, M_t, M'_t and M^n_t, which are respectively used for masking input features, $(t-1)^{\text{th}}$-order interactions and t^{th}-order interactions:

$$\begin{aligned} M_t &= [\overbrace{0, \ldots, 0}^{t-1}, \overbrace{1, \ldots, 1}^{n-t+1}], \\ M'_t &= [\underbrace{1, \ldots, 1}_{C^{t-1}_{n-1}}, \underbrace{0, \ldots, 0}_{C^{t-2}_{n-1}}], \end{aligned} \tag{32}$$

and M^n_t is a $(n-t+1) \times C^{t-1}_{n-1}$ matrix, with the i-th row:

$$M^t_n(i) = [\underbrace{\overbrace{1, \ldots, 1}^{C^{i-1}_{t+i-2}}, 0, \ldots, 0}_{C^{t-1}_{n-1}}]. \tag{33}$$

Based on the above, we develop an iterative enumeration procedure with outer-product and masking. For ease of understanding, we illustrate the procedure by an example in Fig. 5. In particular, given input [user = *Alice*, movie = *Titanic*, genre = *Romance*, tag = *Love Story*] ($[A, T, R, L]$ for short), we perform outer-product between $\{A, T, R\}$ and $\{T, R, L\}$. Since not all generated pairs are valid, we apply masking to obtain all the 2^{nd}-order feature interactions, i.e.,

$$\{\langle A, T\rangle, \langle A, R\rangle, \langle T, R\rangle, \langle A, L\rangle, \langle T, L\rangle, \langle R, L\rangle\}, \tag{34}$$

which are used for 2^{nd}-order FIS. We then compute the outer-product between $\{\langle A, T\rangle, \langle A, R\rangle, \langle T, R\rangle\}$ and $\{R, L\}$. With masking, the 3^{rd}-order feature interactions can be enumerated $\{\langle A, T, R\rangle, \langle A, T, L\rangle, \langle A, R, L\rangle, \langle T, R, L\rangle\}$, which are used for 3^{rd}-order FIS. The 4^{th}-order interaction $\langle A, T, R, L\rangle$ can be constructed by the outer-product between $\langle A, T, R\rangle$ and $\{L\}$.

Since the designed enumeration is based on the outer-product, it can be easily parallelized. The enumeration also helps to share computations: when computing $\boldsymbol{z}_{j_1} \circ \cdots \circ \boldsymbol{z}_{j_t}$, the previous computation of $\boldsymbol{z}_{j_1} \circ \cdots \circ \boldsymbol{z}_{j_{t-1}}$ can be reused.

To theoretically guarantee the comprehensiveness and non-repetitiveness, we propose Theorem 1. $\mathcal{F}^t_i$ is formulated for the set of t^{th}-order feature interactions constructing from the first i features, i.e., $\{j_1, \ldots, j_i\}$, where $\left|\mathcal{F}^t_i\right| = C^t_i$. We only need to prove that given $\mathcal{F}^{t-1}_n$, we can correctly generate $\mathcal{F}^t_n$ based on the designed enumeration procedure.

Algorithm 3: Evaluate $\hat{y}$.

Input : Input feature: $\boldsymbol{x}$
Input : Feature embeddings: $Z, V^{(1)}, \ldots, V^{(m)}$
Output: Predicted score: $\hat{y}$

$Z' \leftarrow Z$;
for $t = 1, \ldots, H$ **do** $V'_t \leftarrow V^{(t)}$;
$W^{(1)} \leftarrow$Reparam(Z', V'_k);
for $t = 2, \ldots, H$ **do**
  $Z' \leftarrow \left[(Z \circ M_t) \odot (Z' \circ M'_t)\right] \circ M^t_n$;
  for $i = t, \ldots, H$ **do**
    $V'_i \leftarrow \left[(V^{(i)} \circ M_t) \odot (V'_i \circ M'_t)\right] \circ M^t_n$;
  $W^{(t)} \leftarrow$Reparam(Z', V'_t);
$\hat{y} \leftarrow$ calculated by Eq. (2);
return $\hat{y}$;

Theorem 1. *Given $\mathcal{F}^{t-1}_n$, we can generate $\mathcal{F}^t_n$ by applying outer-product and maskings, i.e.,*

$$\mathcal{F}^t_n = \left[\{j_1, \ldots, j_n\} \circ M_t \odot \mathcal{F}^{t-1}_n \circ M'_t\right] \circ M^t_n,$$

where $\circ$ and $\odot$ denote the element-wise product and outer-product, respectively. The generation ensures the comprehensiveness and non-repetitiveness.

Proof. According to the *multi-linearity* property [5], we have:

$$\mathcal{F}^t_i = \mathcal{F}^t_{i-1} \cup (\{j_i\} \odot \mathcal{F}^{t-1}_{i-1}), \quad \mathcal{F}^t_{i-1} \cap (\{j_i\} \odot \mathcal{F}^{t-1}_{i-1}) = \emptyset,$$

based on which we can derive the following:

$$\begin{aligned}
\mathcal{F}^t_n &= \bigcup_{i=0}^{n-t} \{j_{t+i}\} \odot \mathcal{F}^{t-1}_{t+i-1} \\
&= \bigcup_{i=0}^{n-t} \{j_{t+i}\} \odot \mathcal{F}^{t-1}_{n-1} \circ M^t_n(i+1) \\
&= \left[\{j_t, \ldots, j_n\} \odot \mathcal{F}^{t-1}_{n-1}\right] \circ M^t_n \\
&= \left[\{j_1, \ldots, j_n\} \circ M_t \odot \mathcal{F}^{t-1}_n \circ M'_t\right] \circ M^t_n. \quad \square
\end{aligned}$$

Based on the idea of feature interaction enumeration, we summarize the overall procedure of evaluating $\hat{y}(\boldsymbol{x})$ in Algorithm 3.

4.3.4. Complexity

We analyze the complexity of the proposed BH-FIS.

Time complexity We analyze the time complexity of Algorithm 3. When evaluating $\hat{y}$ for one sample, only $|\mathcal{F}|$ features are considered rather than D so that $|\mathcal{F}| \ll D$. The major complexity comes from the outer-product. To calculate the embeddings of t^{th}-order feature interactions, we perform the outer-product for $t-1$ times, the complexity of which is $\mathcal{O}(|\mathcal{F}|^{t-1} K)$. The overall complexity is the summation of the above complexities with $t = 2, \ldots, H$, i.e., $\mathcal{O}(|\mathcal{F}|^H K)$. The time complexity is polynomial in $|\mathcal{F}|$. In practice, it is efficient to train since the outer-product operation can be well parallelized.

Space complexity As shown in Section 4.2.2, the parameters of BH-FIS are $\Theta = \{V^{(2)}, \ldots, V^{(H)}, \mu, \sigma\}$ and $\Phi = \{Z, \phi\}$. In total, there are H feature embeddings, along with three additional inference networks parameterized by μ, σ and ϕ, respectively. Since the networks take the embeddings as input, the size of the input layer is K. For the worst case, the number of neurons in each hidden layer is the same as the input layer. Therefore, the space complexity is $\mathcal{O}(HDK + 3K^L)$, where H is the maximum order, D is the number of features, K is the embedding size and L is the maximum number of layers among the networks. Although we analyze μ and σ separately, they actually share the hidden layers, so that the complexity of $3K^L$ can be further reduced. The biggest contribution to the space complexity comes from the embedding part since K and L are generally small.

The space complexity of BH-FIS has the same scale as HOFM. Although the complexity grows linearly with H, in many practical scenarios, setting $H = 3$ is enough to ensure the expressiveness of methods modeling higher-order feature interactions. When H is large, to reduce the complexity, similar to HOFM, we also consider learning shared embeddings to model different orders of feature interactions in our experiment. The space complexity with shared embeddings is $\mathcal{O}(2DK + 3K^L)$.

Table 2
Statistics of the datasets.

Dataset	#Users	#Items	#Fields	#Features	#Ratings	$\frac{\#\text{Rating}}{\#\text{User}}$
ML1M	6,040	3,706	7	13,234	1,000,209	165.60
ML20M	138,493	26,744	3	26,744	39,169,568	282.83
MLHt	2,113	10,197	3	49,205	855,598	404.92
LastFM	1,892	17,632	4	29,200	341,104	180.29
Delicious	1,867	69,223	4	77,735	372,609	199.58
Avazu	–	–	22	1,544,488	40,428,967	

4.3.5. Prediction

Once the parameters $\Theta = \left\{\mu, V^{(2)}, \ldots, V^{(H)}\right\}$, $\Phi = \{\phi, Z\}$ are optimized, we can directly take the variational distributions as the posterior of variables. To predict the output $\hat{y}(\boldsymbol{x})$, we conduct the point estimation of variables, where the posterior means are taken as the value of variables:

$$\mathbb{E}_Q\left[\hat{y}(\boldsymbol{x})\right] = w_0 + \sum_{i=1}^{D} w_i + \sum_{t=2}^{H} \sum_{j_1 < \cdots < j_t \in \mathcal{F}} \mathbb{E}_Q\left[w^{(t)}_{j_1,\ldots,j_t}\right],$$

where

$$\begin{aligned}
&\mathbb{E}_Q\left[w^{(t)}_{j_1,\ldots,j_t}\right] \\
&= \mathbb{E}_Q\left[\tilde{w}^{(t)}_{j_1,\ldots,j_t}\right] \cdot \mathbb{E}_Q\left[s_{j_1,\ldots,j_t}\right] \\
&= f_\mu(\boldsymbol{v}^{(t)}_{j_1} \circ \cdots \circ \boldsymbol{v}^{(t)}_{j_t}) \cdot f_\phi(\boldsymbol{z}_{j_1} \circ \cdots \circ \boldsymbol{z}_{j_t}).
\end{aligned}$$

5. Experimental setup

We conduct an extensive empirical evaluation of BP-FIS and BH-FIS, respectively, to show the advantages of FIS in selecting pairwise and higher-order feature interactions. We carried out the experiments on two different tasks for a comprehensive evaluation:

- *Top-N recommendation*: Given a user, to predict the scores of the user to the items in the candidate item list. Items are then ranked based on the descending order of the predicted score and the top-N items are recommended to the user. BP-FIS is evaluated on the top-N recommendation task.
- *Click-through rate prediction*: Given an ad with features, to predict whether the ad will be clicked or not. click-through rate (CTR) prediction is defined as a binary classification problem. CTR prediction is also used to evaluate BH-FIS since modeling higher-order feature interactions has been shown to improve the performance of prediction.

5.1. Research questions

We seek to answer the following research questions:

(RQ1) Does P-FIS help to improve the performance of FMs on the top-N recommendation task? Specifically, how well does BP-FIS perform against state-of-the-art FMs?
(RQ2) How does the embedding size impact the ability of BP-FIS to improve the performance of FMs? How does the alternating optimization procedure affect the performance of P-FIS?
(RQ3) Does applying H-FIS on HOFMs improve the performance of prediction? Compared with counterparts, how efficient is BH-FIS?
(RQ4) What is the impact of the order of feature interactions and embedding size on the effectiveness of BH-FIS? What is the significance of the feature interactions of different orders selected by BH-FIS?

We answer RQ1 and RQ2 by performing the top-N recommendation task and address RQ3 and RQ4 with CTR prediction.

5.2. Datasets

The experiments are conducted on six datasets, two of which are constructed from MovieLens datasets [19]. See Table 2.

- **ML1M** – This is the MovieLens dataset with one million movie ratings.[4] Besides rating information, ML1M also contains rich additional information including user profiles (gender, age, occupation, zip) and movie genres;
- **ML20M** – This is the MovieLens dataset containing 20 million ratings.[5] Compared with ML1M, ML20M lacks the additional information. Instead, we sort items by users according to time and form the fields as user, item and latest historical items.

Three of the datasets are constructed by HetRec 2011.[6] These datasets contain social networking, tagging, and resource consuming (Web page bookmarking and music artist listening) information from sets of around 2,000 users. There three datasets are specially utilized to evaluate top-N recommendation.

- **MLHt** – This dataset is an extension of ML10m.[7] It links the movies of the MovieLens dataset with their corresponding web pages at the Internet Movie Database (IMDb)[8] and Rotten Tomatoes movie review systems.[9] MLHt contains users with both rating and tagging information.
- **LastFM** – This dataset contains social networking, tagging, and music artist listening information from Last.fm,[10] the online music system.[7]Each user has a list of most listened music artists, tag assignments, and friend relations within the social network.
- **Delicious** – This dataset contains social networking, bookmarking, and tagging information from the Delicious[11] social bookmarking system.[7]Each user has bookmarks, tag assignments, and contact relations within the social network.

Finally, we also use one benchmark dataset that is used for evaluating CTR prediction.

- **Avazu** – This dataset contains users' mobile behaviors including whether a displayed mobile ad is clicked by a user or not.[12] It has 22 feature fields spanning from user/device features to ad attributes (we discard the useless field id that has a unique value for each sample). We remove the infrequent features and treat them as a single feature "unknown," where we set a threshold to 5.

All datasets but Avazu are used to evaluate top-N recommendation since Avazu lacks user identifier. Avazu is used to evaluate CTR prediction. By regarding user identifier as one of the features (fields), ML1M and MLHt are also used to evaluate CTR prediction. We follow the common setting of evaluating top-N recommendation with implicit feedback [25,71]: treat ratings of at least 3 as positive ($y = 1$ if rating ≥ 3) and treat all other ratings as missing ($y = 0$).

5.3. Methods used for comparison

Personalized FIS To evaluate the effectiveness of BP-FIS for FMs, we apply P-FIS on both linear and non-linear FMs. This yields two methods for comparison. They differ in the definition of $f_\phi(\cdot)$ in Eq. (15): $f_\phi(\cdot)$ is an identity transformation for PFIS and a stack of fully connected network layers for PNFIS. We evaluate PFIS and PNFIS on the top-N recommendation task since they both consider personalization during FIS.

- **PFIS**, which applies P-FIS on top of **FM** [50,51], and
- **PNFIS**, which applies P-FIS on top of **NFM** [20]. NFM is the state-of-the-art neural extension of factorization machines.

High-order FIS Similarly, to evaluate BH-FIS, we also compare with different variations of HOFMs:

- **HFIS**, which applies H-FIS on top of **HFM**. HFM is the original version of HOFM proposed by Blondel et al. [4]. It learns a separate set of feature embeddings for each order of feature interactions;
- **SFIS**, which applies H-FIS on top of **SFM**. SFM is simplified version of the HOFM with shared feature embeddings [4] by removing the kernels;
- **HNFIS**, which applies H-FIS on top of **HNFM**. HNFM extends HFM to neural methods based on the idea of NFM; and
- **SNFIS**, which applies H-FIS on top of **SNFM**. SNFM extends SFM to neural methods based on the idea of NFM.

[4] https://grouplens.org/datasets/movielens/1m/.
[5] https://grouplens.org/datasets/movielens/20m/.
[6] http://ir.ii.uam.es/hetrec2011.
[7] https://grouplens.org/datasets/hetrec-2011/.
[8] http://www.imdb.com.
[9] http://www.rottentomatoes.com.
[10] http://www.last.fm.
[11] http://www.delicious.com.
[12] https://www.kaggle.com/c/avazu-ctr-prediction.

Since HOFMs capture explicit feature interactions only, we also combine them with deep neural networks (DNNs) to capture implicit feature interactions. This design is widely applied for CTR prediction, e.g., **DeepFM** is the combination of FM with DNN.

- **HDeepFIS**, which applies H-FIS on top of **HDeepFM**. HDeepFM is the combination of HFM and DNN, and
- **SDeepFIS**, which applies H-FIS on top of **SDeepFM**. SDeepFM is the combination of SFM and DNN.

HFIS and SFIS are evaluated by the CTR prediction task. For a fair comparison with baselines of the CTR prediction task, we also incorporate DNN to capture implicit feature interactions, where we evaluate HDeepFIS and SDeepFIS.

Baselines We also compare with the following baselines:

- **AFM**: The attentional factorization machine (AFM) [72] learns the importance of pairwise feature interactions from data via an attention network.
- **SparseFM**: The sparse factorization machine (SparseFM) learns sparse 1^{st}- and 2^{nd}-order interactions [47,73,79]. We implement SparseFM, where all feature embeddings are penalized by group Lasso regularizations.
- **CIN**: The compressed interaction network (CIN) [35] is the state-of-the-art method for CTR prediction, which approximate higher-order feature interactions explicitly. To account for implicit feature interactions, CIN is combined with DNN to form **xDeepFM**.
- **AutoInt**: The automatic feature interaction learning (AutoInt) [59] is the state-of-the-art method for CTR prediction, which captures the importance of higher-order feature interactions via self-attention mechanism. **AutoInt+** combines CIN with DNN.
- **AutoFIS**: The recent work that performs field interaction selection on top of higher-order factorization machines [38]. AutoFIS is applied both on FM and DeepFM, which forms **AutoFM** and **AutodeepFM**, respectively.
- **LightFM**: The LightFM [32] is a hybrid matrix factorization model representing users and items as linear combinations of their content features' latent factors.
- **GPFM**: The GPFM [44] is the non-linear and non-parametric context-aware collaborative filtering (CF) method. It models the utility function using the Gaussian process (GP) framework. **GPPW** is the pairwise preference model of **GPFM**.

The methods mentioned above (in **bold face**) are compared in our experiments. We summarize these methods in Table 3. Implementation details are provided in Appendix A.

5.4. Evaluation protocol

MLHt, ML1M, ML20M, LastFM and Delicious are used to evaluate top-N recommendation. Note that LastFM and Delicious contains implicit ratings only. While MLHt, ML1M and ML20M contain explicit ratings, we convert them into implicit for the sake of effectiveness of training and the consistence of evaluation. The implicit ratings generally tell whether an item is relevant or not. While other information like dwell time can tell the grade of relevance, it is not applicable in our datasets. Therefore, we use binary relevance metrics to evaluate top-N recommendation.

We adopt the leave-one-out evaluation, which has been widely used in the literature [26,45]. For each user, we randomly hold-out one of her interactions as the test set and utilize the remaining data for training. Since it is too time-consuming to rank all items for every user during evaluation, we follow the common strategy of randomly sampling items that are not interacted with by the user, ranking the test item among 100 items [31,36]. hit rate (HR) and average reciprocal hit rank (ARHR) are widely utilized to evaluate the performance of top-N recommendation.

The recommendation quality is measured using HR and ARHR, which are defined as follows:

$$\mathrm{HR} = \frac{\#\mathrm{Hit}}{\#\mathrm{User}}, \quad \mathrm{ARHR} = \frac{1}{\#\mathrm{User}} \sum_{i=1}^{\#\mathrm{Hit}} \frac{1}{pos_i},$$

where #User is the total number of users, and #Hit is the number of users whose item in the test set is recommended (i.e., a hit) in the size-N recommendation list. Moreover, pos_i is the position of the item in the ranked recommendation list, if an item of a user is among the list. ARHR is a weighted version of HR and it measures how strongly an item is recommended; the weight is the reciprocal of the hit position in the recommendation list. Note that both HR and ARHR belong to binary relevance metrics. The graded relevance metrics are not applicable.

To evaluate CTR prediction, we employ two metrics: area under the ROC curve (AUC) and binary cross-entropy loss (logLoss). They evaluate the performance from two different perspectives: AUC measures the probability that a positive instance will be ranked higher than a randomly chosen negative one. It only takes into account the order of predicted instances and is insensitive to the class imbalance problem. In contrast, logLoss measures the distance between the predicted score and the true label for each instance:

$$\mathrm{logLoss} = -\frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x} \in \mathcal{X}} y(\boldsymbol{x}) \log \hat{y}(\boldsymbol{x}) + (1 - y(\boldsymbol{x})) \log \left(1 - \hat{y}(\boldsymbol{x})\right). \tag{35}$$

Table 3
Summary of compared methods.

	Method	*Higher-order*	*FIS*	*Non-linear*	*+DNN*
FM	AFM	–	–	–	–
	DeepFM	–	–	–	√
	FM	–	–	–	–
	LightFM	–	–	–	–
	NFM	–	–	√	–
	SparseFM	–	√	–	–
HOFM	HDeepFM	√	–	–	√
	HFM	√	–	–	–
	HNFM	√	–	√	–
	SDeepFM	√	–	–	√
	SFM	√	–	–	–
	SNFM	√	–	√	–
Baseline	AutoInt	√	–	√	–
	AutoInt+	√	–	√	√
	CIN	√	–	√	–
	AutoFM	√	√	–	–
	AutoDeepFM	√	√	–	√
	xDeepFM	√	–	√	√
BP-FIS	PFIS	–	√	–	–
	PNFIS	–	√	√	–
BH-FIS	HDeepFIS	√	√	–	√
	HFIS	√	√	–	–
	HNFIS	√	√	√	–
	SDeepFIS	√	√	–	√
	SFIS	√	√	–	–
	SNFIS	√	√	√	–

"*Higher-order*" indicates whether higher-order feature interactions are modeled. "*FIS*" indicates whether the method select feature interactions. "*Non-linear*" shows whether non-linear transformations are applied. "*+DNN*" shows if DNN is combined with the method during prediction.

We use logLoss to train the methods for the CTR prediction task.

6. Experimental results

We group our experimental results by task, getting started with top-*N* recommendation.

6.1. Top-N recommendation

RQ1 To answer RQ1 for top-*N* recommendation, we compare PFIS and PNFIS with FM and NFM, respectively. We also compare with three baseline methods: AFM, SparseFM, LightFM. We report the recommendation performance of the compared models in Table 4, in terms of HR@1, HR@10 and ARHR@10. Table 4 shows that PFIS and PNFIS consistently outperform the other methods. This illustrates the effectiveness of BP-FIS for improving the performance of both linear and non-linear FMs for top-*N* recommendation.

We analyze the results by separating the comparison between linear and non-linear models. Among the linear models, SparseFM outperforms FM and AFM on the LastFM and Delicious datasets. This supports the need for conducting FIS for FMs. However, SparseFM fails to beat FM on MLHt. In contrast, PFIS achieves improvements over FM constantly on the same dataset. This shows that selecting or weighing a single set of feature interactions for all users overlook the personalization over features, and conducting P-FIS is required. Except for HR@1 on LastFM and Delicious, the improvement achieved by PFIS is significant, especially HR@10 on MLHt, where PFIS improves over FM by 17.286%. This demonstrates the efficacy of BP-FIS. On the Delicious dataset, LightFM performs well. It even outperforms PFIS for HR@1, though the improvement is not significant. This might suggest that the interactions among user features bring more noise than information on the Delicious dataset.

The comparison between the non-linear models, i.e., NFM and PNFIS, shows similar results. PNFIS steadily achieves better performance than NFM on all datasets and all metrics, which shows that the improvement achieved by BP-FIS is orthogonal to the non-linear modeling of interactions.

Results on ML1M and ML20M are reported in Table 5. Similarly, we compare linear and non-linear methods separately. For linear methods, PFIS performs the best and the improvement is significant. This further confirms that performing FIS is necessary and validates the proposed BP-FIS model. Similar results are reveals when comparing non-linear methods. PNFIS

Table 4
Comparison of top-N recommendation methods.

	Method	λ	drop	K	HR@1	HR@10	ARHR@10
MLHt	AFM	2	0.1	64	0.1138	0.4273	0.1969
	FM	0.1	–	64	0.2371	0.6028	0.3398
	SparseFM	0.01	–	64	0.2294	0.6052	0.3351
	LightFM	0.01	–	64	0.0856	0.3834	0.1619
	PFIS	–	–	128	**0.2401**†	**0.7070**‡	**0.3932**‡
	NFM	–	0.2	256	0.2180	0.6257	0.3389
	PNFIS	–	–	128	**0.2519**‡	**0.6831**‡	**0.3814**‡
LastFM	AFM	8	0.7	256	0.2166	0.6126	0.3332
	FM	0.1	–	64	0.1894	0.6403	0.3215
	SparseFM	0.05	–	256	0.2118	0.6542	0.3449
	LightFM	0.01	–	64	0.2204	0.5598	0.3151
	PFIS	–	–	256	**0.2209**	**0.6798**‡	**0.3581**‡
	NFM	–	0.6	64	0.2150	0.6798	0.3563
	PNFIS	–	–	256	**0.2257**	**0.6910**	**0.3660**
Delicious	AFM	2	0.1	64	0.0274	0.1169	0.0494
	FM	0.1	–	64	0.0202	0.1147	0.0440
	SparseFM	0.1	–	128	0.0229	0.1212	0.0465
	LightFM	0.01	–	64	**0.0295**	0.1141	0.0502
	PFIS	–	–	128	0.0278	**0.1240**‡	**0.0509**†
	NFM	–	0.1	64	0.0229	0.1065	0.0426
	PNFIS	–	–	128	**0.0268**	**0.1289**‡	**0.0504**‡

Boldface scores indicate the best results for linear and non-linear FMs on each metric. We conduct two-sided tests for the null hypothesis that the best and the second best have identical average values. † and ‡ indicate that the best score is significantly better than the second best score with p-value < 0.1 and < 0.05, respectively.

Table 5
Top-N recommendation performance on ML1M and ML20M.

Method	ML1M			ML20M		
	HR@1	HR@10	ARHR@10	HR@1	HR@10	ARHR@10
FM	0.2072	0.7394	0.3620	0.4181	0.9264	0.5887
AFM	0.2380	0.7382	0.3877	0.3673	0.9242	0.5522
SparseFM	0.2171	0.7511	0.3754	0.1047	0.8038	0.2818
LightFM	0.0411	0.2643	0.0937	0.2154	0.8576	0.5034
PFIS	**0.2437**‡	**0.7710**‡	**0.3988**‡	**0.4263**‡	**0.9315**†	**0.5970**†
GPFM	0.1676	0.5903	0.2775	–	–	–
GPPW	0.1702	0.6054	0.2846	–	–	–
NFM	0.2148	0.7607	0.3776	0.1872	0.8654	0.3816
PNFIS	**0.2388**‡	**0.7652**	**0.3955**†	**0.4154**‡	**0.9154**‡	**0.5778**‡

The implementation efficiency of GPFM and GPPW is low, and the training cannot be completed in a feasible time. For ML20M, we fix the embedding size as 16 for all the compared methods. Since ML20M contains large number of users, performing P-FIS can be time-consuming. Instead, we follow the idea of [11] to cluster users into groups and perform FIS for each user group.

is significantly better than NFM, demonstrating the effectiveness of FIS on non-linear FMs. NFM is also superior to GPFM and GPPW. This is justifiable since NFM is recently proposed and has achieved state-of-the-art performance. However, PNFIS cannot outperform PFIS, showing that FIS on linear FMs is more effective.

To have a complete analysis of the strengths and weakness of our proposal, we also consider evaluating the top-N recommendation performance by *graded metrics*. Currently, items in the test set are considered to be relevant or not relevant, which is binary. The graded metrics distinguish the degree of relevance. Given the explicit ratings, we can conduct evaluation with graded metrics. We follow [22] to evaluate the top-N recommendation performance on nDCG. nDCG is defined as

$$\text{nDCG@}N = \frac{\text{DCG@}N}{\text{IDCG@}N}, \tag{36}$$

where

$$\text{DCG@}N = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \tag{37}$$

and IDCG@N is ideal value of DCG@N. Typically in recommendation, rel_i is binary that

Table 6
Performance Comparison on graded metric (nDCG).

Method	nDCG@10
FM	0.4682
AFM	0.4465
SparseFM	0.4699
NeurFM	0.4653
PFIS	0.4706
PNFIS	0.4700

$$rel_i = \begin{cases} 1 & r_{ui} \geq \sigma, \\ 0 & \text{otherwise}, \end{cases} \tag{38}$$

where σ is the binary threshold. (We set $\sigma = 3$ as we discussed in Section 5.2.) With explicit ratings, we define the graded relevance:

$$rel_i = \begin{cases} r_i - \sigma + 1 & r_{ui} \geq \sigma, \\ 0 & \text{otherwise}. \end{cases} \tag{39}$$

Based on this, we compare PFIS and PNFIS with baselines on ML1M by nDCG. Results are reported in Table 6. Similar to the results in Table 5, PFIS and PNFIS outperform other methods and PFIS reach the top. This further confirms the effectiveness of performing BP-FIS on FMs.

RQ2 To answer RQ2 for top-N recommendation, we analyze the performance of all models with different embedding sizes. Specifically, we plot figures to show results w.r.t. HR@1, HR@5, HR@10, ARHR@5 and ARHR@10 of all models on the MLHt dataset, as shown in Fig. 6. Generally, we can see that PFIS and PNFIS achieve a better performance than the other models. This demonstrates the robustness of BP-FIS as it consistently improves the performance of FMs, regardless of the embedding size and evaluation metric.

Specifically, for the setting $K = 64$, almost all models show a competitive performance. This is because the space for the performance improvement is limited when $K = 64$ on the MLHt dataset. P-FIS has an insignificant effect on FMs due to the restricted embedding size. In contrast, for the setting $K = 128$, while FM, SparseFM and NFM achieve comparable results, PFIS and PNFIS significantly outperform them. This means that the effect of BP-FIS can be better expressed by increasing the embedding size for FMs. Interestingly, while PFIS performs better than PNFIS for most metrics, the performance gain achieved by PNFIS is more remarkable on HR@1. Similar performance levels are seen when $K = 256$, except that PNFIS does not outperform PFIS. This might be because PNFIS has far more parameters than PFIS when the embedding size is large, which brings extra difficulty to avoid overfitting.

To analyze alternative training procedures of BP-FIS, we show the performance of PFIS and PNFIS after each iteration in Fig. 7. A general trend could be revealed by Fig. 7 that the recommendation performance of both PFIS and PNFIS grows initially and fluctuate successively. Although PNFIS can mostly achieve better performance than PFIS, it also shows higher variation.

When $K = 64$, PFIS outperforms PNFIS w.r.t. HR@1 and performs competitively with PNFIS w.r.t. HR@5 and HR@10. When $K = 128$, we can witness a relatively stable growth in PFIS, especially for HR@5 and HR@10. While a certain level of convergence can be witnessed for HR@5 and HR@10, PNFIS still fluctuates more than PFIS during the training procedure. These observations might suggest that the training of BP-FIS shows better robustness for linear FMs than non-linear ones. PNFIS is more unstable in terms of HR@5 and HR@10 when $K = 256$, the performance of which drops sharply during the 9^{th} iteration. Thus, more iterations do not help a lot for improving the performance but might harm the performance. Another observation is that the training procedure of PFIS and PNFIS varies with different embedding sizes. Training PFIS and PNFIS with $K = 128$ provides the most stable procedure. This shows that a proper selection of embedding size can further extend the potential of BP-FIS.

6.2. CTR prediction

We also answer RQ3 and RQ4 for the CTR prediction task.

RQ3 To answer RQ3 for the CTR prediction task, we first show the outcomes of our experiment on the Avazu dataset; see Table 7. To capture implicit feature interactions, a separate DNN module is typically incorporated; results for the addition of this ingredient are also included in the table (in the rightmost column of Table 7). To compare with baselines, we take the results reported in [59] since we follow exactly the same experimental settings. Note that that a slightly higher AUC or lower logLoss at 0.001-level is regarded significant for CTR prediction task, which has also been pointed out in existing works [13,18,58,59].

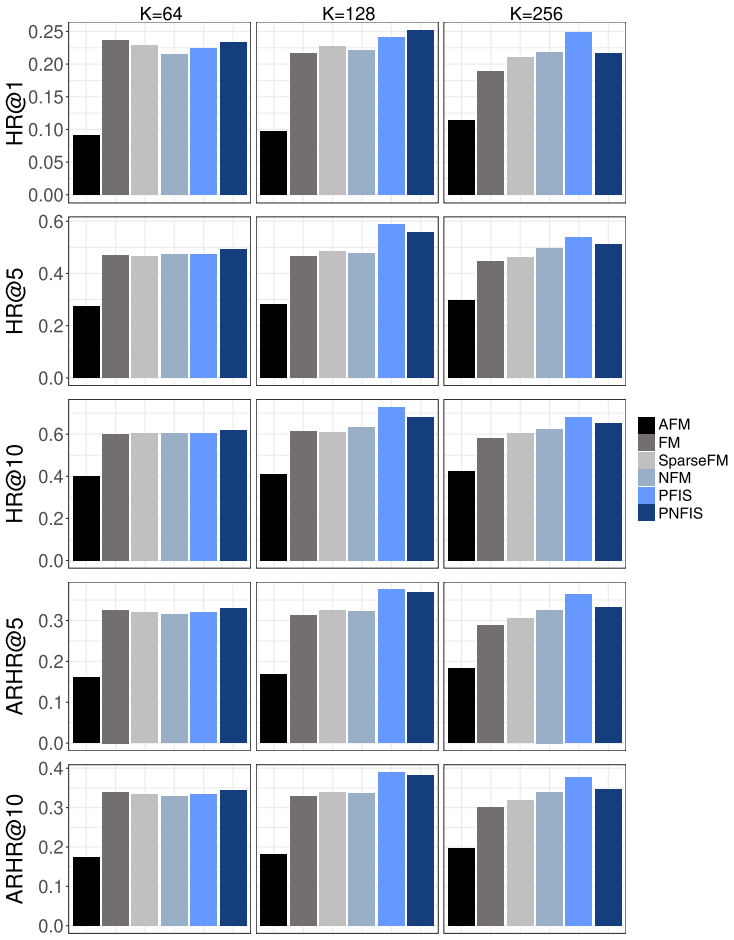

Fig. 6. Performance comparison w.r.t. different embedding sizes. ARHR@1 is actually HR@1.

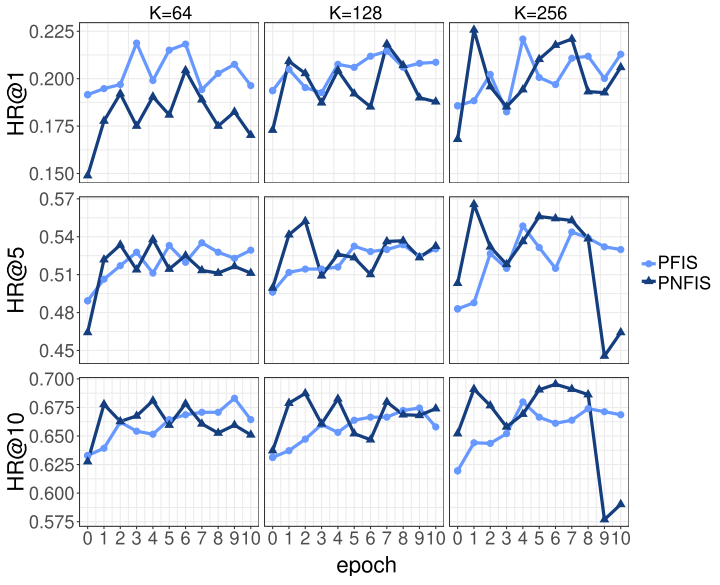

Fig. 7. Training procedure of PFIS and PNFIS on LastFM dataset.

Table 7
Comparison of CTR prediction on the Avazu dataset.

				+DNN		
	Method	AUC	logLoss	Method	AUC	logLoss
Baseline	FM	0.7706	0.3856	DeepFM	0.7751	0.3829
	AFM	0.7718	0.3854	–	–	–
	NFM	0.7708	0.3864	–	–	–
	CIN	0.7758	0.3829	xDeepFM	0.7770	0.3823
	AutoInt	0.7752	0.3824	AutoInt+	<u>0.7774</u>	<u>0.3811</u>
	AutoFM	<u>0.7778</u>	<u>0.3818</u>	AutoDeepFM	0.7754	0.3828
HOFM	SFM	0.7732	0.3850	SDeepFM	0.7769	0.3823
	HFM	0.7746	0.3855	HDeepFM	0.7631	0.3910
BH-FIS	SFIS	0.7828	0.3787	SDeepFIS	0.7829	0.3783
	HFIS	**0.7862**	**0.3768**	HDeepFIS	**0.7858**	**0.3773**

The embedding size and the order of feature interactions are fixed as 16 and 3, respectively. The best results are in **boldface** and the second best are <u>underlined</u>, both with or without DNN. We underline the best baseline results (except BH-FISs) if the best is achieved by BH-FISs.

Table 8
Comparison of training and evaluation time.

Order				+DNN		
	Method	Train	Eval	Method	Train	Eval
2^{nd}	SparseFM	8.02	18	–	–	–
	AutoFM	4.70	8	AutoDeepFM	8.40	18
	SFIS	8.28	30	SDeepFIS	8.53	30
	HFIS	7.80	28	HDeepFIS	9.43	28
3^{rd}	AutoFM	20.68	49	AutoDeepFM	24.31	58
	SFIS	24.80	67	SDeepFIS	26.75	69
	HFIS	25.35	69	HDeepFIS	26.38	67
4^{th}	AutoFM	–	–	AutoDeepFM	–	–
	SFIS	126.93	327	SDeepFIS	134.23	347
	HFIS	126.58	329	HDeepFIS	137.27	348

"Train" means the training time, measured in minutes; "Eval" means the evaluation time, measured in seconds. Please note that this is not a rigorous comparison due to the difference in the implementation details.

As shown in Table 7, the best performance of AUC and logLoss are both achieved by BH-FIS methods. HFIS and HDeepFIS perform better than SFIS and SDeepFIS, respectively. Without DNN, AutoFM is the best performed baseline. However, it has been significantly outperformed by SFIS and HFIS. With DNN, AutoDeepFM is outperformed by AutoInt+. Similarly, it still fails to perform equally better as SDeepFIS and HDeepFIS. In short, on the Avazu dataset, although the state-of-the-art baselines beat HOFMs, they can be outperformed by incorporating BH-FIS, especially on AUC metric.

We also conduct efficiency analysis. We compare both the training time and evaluation time of BH-FIS methods against the counterparts for different orders of interactions (the baseline methods that employ FIS, i.e., SparseFM and AutoFIS). For both training and evaluation, we report the time of one epoch. Specifically, (1) for BH-FIS, it is the time of training feature interaction selection; (2) for AutoFIS, it is the time of searching; and (3) for SparseFM, it is the time of training since the selection is embedded into the learning.

Results are reported in Table 8. Please note that BH-FIS performs a harder task than SparseFM and AutoFIS since BH-FIS involves discrete variables during training. Therefore, the comparison is actually unfair for BH-FIS. As shown in Table 8, when FMs are limited to 2^{nd}-order, BH-FIS methods generally require more time to train and evaluate, except that HFIS is more efficient in training than SparseFM. When modeling 3^{rd}-order interactions, AutoFIS models are constantly efficient than BH-FIS methods (SparseFM models 2^{nd}-order feature interactions only). Please note that AutoFIS selects field interactions rather than feature interactions, where the number of fields is much smaller than the number of features. Nevertheless, the training and evaluation time of both methods are on the same scale. While the growth of time is exponential for BH-FIS, which is confirmed by the time complexity analysis, it provides the possibility to model and select even higher-order feature interaction. In comparison, AutoFIS cannot select more than 3^{rd}-order interactions.

In short, although BH-FIS is less efficient than SparseFM and AutoFIS, the training and evaluation time of BH-FIS is on the same scale as the counterparts. This demonstrates that BH-FIS does not bring extra complexity. Actually, this efficiency of selection is achieved through variational inference.

Next, we report the experimental results on MLHt and ML1M in Table 9 and 10, respectively. We conduct two-sided significance tests to compare BH-FISs with HOFMs and the best performing baselines, respectively. To conduct the test on AUC, we follow [16] to compare the two ROC curves. Among each group, the compared methods are further split into two

Table 9
Comparison of CTR prediction on the ML1M dataset.

	Method			Train	Valid		Test	
		H	K	LogLoss	LogLoss	AUC	LogLoss	AUC
Baseline	FM	2	64	0.3844	0.3868	0.8381	0.3798	0.8418
	AFM	2	64	0.3846	0.3876	0.8376	0.3817	0.8398
	NFM	3	8	0.3040	0.3246	0.8907	0.3210	0.8911
	CIN	2	8	0.2857	0.3329	0.8883	0.3297	0.8878
	AutoInt	2	16	0.3044	0.3251	0.8907	0.3242	0.8894
	DeepFM	2	8	0.2633	0.3229	0.8959	0.3203	0.8967
	xDeepFM	4	8	0.2659	0.3199	0.8967	0.3170	0.8969
	AutoInt+	2	8	0.2863	0.3157	0.8978	<u>0.3139</u>	<u>0.8977</u>
HOFM	SFM	5	64	0.2626	0.3162	0.8988	0.3150	0.8987
	HFM	4	32	0.2752	0.3158	0.8973	0.3139	0.8975
	SDeepFM	4	64	0.2926	0.3175	0.8961	0.3157	0.8959
	HDeepFM	6	64	0.2945	0.3180	0.8961	0.3157	0.8963
BH-FIS	SFIS	4	64	0.2646	0.3149	0.8989	**0.3119**†‡	**0.8991**†‡
	HFIS	6	64	0.2775	0.3162	0.8973	0.3125	0.8979†
	SDeepFIS	4	64	0.2561	0.3163	0.8985	0.3149	0.8977†
	HDeepFIS	3	64	0.2678	0.3160	0.8976	0.3133†	0.8975†

We **boldface** the best result on the test set. We also <u>underline</u> the best results achieved by baselines (except BH-FISs). We conduct two-sided significant tests for the null hypothesis that the compared results have identical average values. The improvement is regarded as significant if the p-value is no larger than 0.05. † indicates a significant improvement of BH-FIS over the HOFM counterpart and ‡ indicates a significant improvement over the best baseline.

Table 10
Comparison of CTR prediction on the MLHt dataset. Same conventions as in Table 9.

	Method			Train	Validation		Test	
		H	K	LogLoss	LogLoss	AUC	LogLoss	AUC
Baseline	FM	2	32	0.4203	0.4252	0.8157	0.4218	0.8195
	AFM	2	32	0.4201	0.4248	0.8157	0.4222	0.8184
	NFM	2	8	0.3684	0.3802	0.8582	0.3750	0.8627
	CIN	6	8	0.3083	0.3707	0.8707	0.3688	0.8726
	AutoInt	3	8	0.3268	0.3649	0.8721	0.3637	0.8741
	DeepFM	2	8	0.3100	0.3675	0.8736	0.3646	<u>0.8763</u>
	xDeepFM	2	8	0.3208	0.3665	0.8721	0.3642	0.8744
	AutoInt+	2	8	0.3354	0.3645	0.8715	<u>0.3613</u>	0.8753
HOFM	SFM	6	64	0.3310	0.3673	0.8702	0.3634	0.8735
	HFM	6	64	0.3297	0.3663	0.8721	0.3643	0.8737
	SDeepFM	4	64	0.3556	0.3714	0.8651	0.3683	0.8680
	HDeepFM	6	8	0.3329	0.3705	0.8684	0.3679	0.8711
BH-FIS	SFIS	4	32	0.3005	0.3605	0.8775	0.3586†	0.8790†
	HFIS	6	64	0.2922	0.3565	0.8794	**0.3546**†‡	**0.8809**†‡
	SDeepFIS	5	32	0.2979	0.3613	0.8774	0.3581†‡	0.8795†‡
	HDeepFIS	5	64	0.3070	0.3577	0.8777	0.3555†‡	0.8796†‡

sub-groups: methods with/without DNN. We only record the best results performed by each method in the table, i.e., the results by the trained model with the least logLoss on the validation set. The results on the validation set are reported to show how well the performance of the selected model on the validation reflects the performance on the test set. The training logLosses are also recorded in the table to show how well the data can be fitted.

On the ML1M dataset (in Table 9), all BH-FISs improve over their HOFM counterparts. SFIS and HDeepFIS significantly outperform SFM and HDeepFM on both metrics. HFIS and SDeepFIS outperform HFM and SDeepFM on both metrics and the improvement w.r.t. AUC is significant. This confirms the effectiveness of H-FIS for HOFM on CTR prediction tasks. SFIS performs the best on both metrics and the improvement over the best baseline AutoInt+ is significant. While HFIS also outperforms AutoInt+, the improvement is not significant. This seems counter-intuitive since BH-FIS with order-aware embeddings performs better than with shared embeddings for the rating prediction task on the same dataset. The effectiveness of SFIS comes both from SFM and H-FIS. Actually, the training error of SFM is low, though the validation and test error is high, meaning that SFM is a potentially good model. By selecting higher-order feature interactions, FIS helps to avoid overfitting, where the potential capability of SFM has been well exploited. SDeepFIS and HDeepFIS fail to outperform AutoInt+, showing that adding DNN to BH-FIS may not be helpful. We claim that since HOFM and BH-FIS can already model the feature interactions well, it is unnecessary to add extra DNN to capture implicit feature interactions.

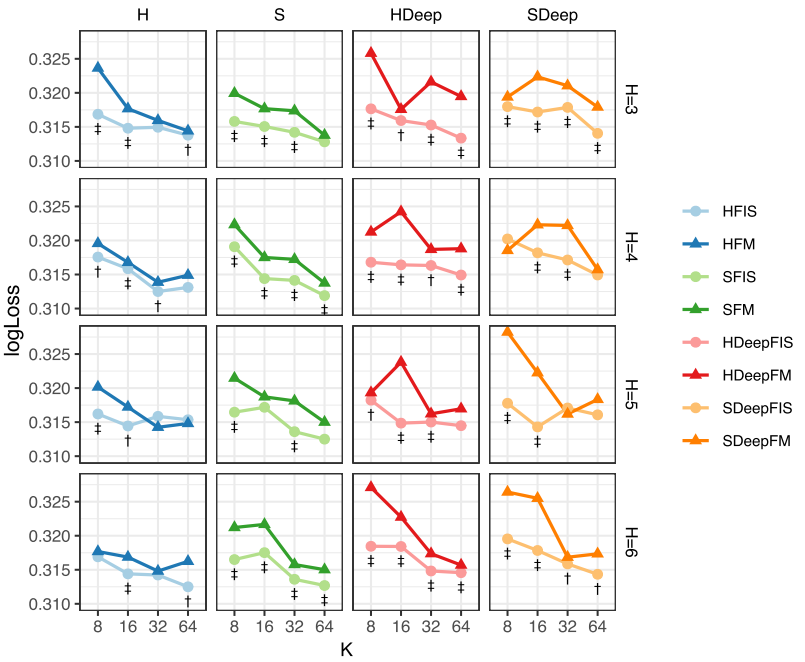

Fig. 8. Pairwise comparison of CTR prediction on the ML1M dataset w.r.t. logLoss. We put a label under the line of BH-FISs if the improvement is significant. † and ‡ indicate the significance with p-value less than 0.05 and 0.01, respectively.

On the MLHt dataset, as shown in Table 10, all four BH-FIS variants significantly outperform their HOFM counterparts, which further confirms the effectiveness of H-FIS. Besides BH-FIS methods, the best logLoss is achieved by AutoInt+ and the best AUC is achieved by DeepFM. This shows the effectiveness of modeling implicit feature interactions via DNNs for the baseline methods. BH-FISs outperform the two baselines on the respective metrics. Except for SFIS, the improvements over the baselines are also significant, among which HFIS performs the best. HFIS also has the smallest training loss and the best performance on the validation set. This demonstrates that HFIS can well fit the MLHt dataset for the CTR prediction task and can generalize during test time. In short, adding DNN is effective only for pairwise FMs and NN-based approximation methods, where higher-order feature interactions are not well modeled.

To sum up, (1) all BH-FISs improve over the HOFM counterparts, and for most cases the improvement is significant, showing the effectiveness of H-FIS for CTR prediction; (2) while BH-FIS with respective embeddings is the best performed method on the MLHt dataset, which is similar to that on the ML1M dataset, SFIS achieves the best performance on the ML1M dataset. The effectiveness of SFIS comes both from SFM and H-FIS; and (3) although +DNN helps to improve the prediction accuracy for the baseline methods, it does not improve HOFM and BH-FIS, which already manage to model feature interactions effectively.

RQ4 To answer RQ4 for the CTR prediction task, we show the impact of parameters, i.e., the maximum order of feature interactions H and the embedding size K, where H is grid-searched from 3, 4, 5, 6 and K from 8, 16, 32, 64. For each value of H and K, BH-FISs are compared with their HOFM counterparts. We show the results of pairwise comparison by plotting line-point figures in Fig. 8, 9, 10 and 11. Significant testing is performed to see if the improvement of BH-FISs over HOFMs is significant or not.

We first study Fig. 8 and 9 to analyze the experimental results on the ML1M dataset. In Fig. 8, the logLosses on the test set are compared for all methods. The first row of Fig. 8 shows the comparison when modeling 3^{rd}-order feature interactions ($H = 3$). HFIS outperforms HFM, and except for $K = 32$, the improvement is significant. Similar results are shown when comparing SFIS and SFM: except for $K = 64$, the improvement achieved by SFIS is significant. With DNN, HDeepFIS and SDeepFIS consistently outperform HDeepFM and SDeepFM, respectively, and significantly so. The lowest logLoss is achieved by SFIS when $K = 64$. Next, we look at the second row of Fig. 8, where $H = 4$. Similar results are shown, except that SDeepFIS is outperformed by SDeepFM when $K = 8$. When $H = 5$, SFIS is not a better performing method than SFM if $K = 32$ or $K = 64$. SDeepFIS is also outperformed by SDeepFM when $K = 32$. Similarly when $H = 6$ (the last row of Fig. 8), BH-FISs significantly improve over HOFMs except for SFIS when $K = 8$ and $K = 32$. The effectiveness of H-FIS is well demonstrated since it improves the performance of HOFMs significantly for most cases. A general trend to be witnessed is that logLoss decreases when K and H increase, which is intuitive since an increase in K and H means an increase in model expressiveness. Fig. 9 shows the performance w.r.t. AUC. Similarly, AUC increases with the growth of K and H. Different from the logLoss in Fig. 8, it is more difficult for HOFM to outperform BH-FIS, where the effectiveness of BH-FIS is better demonstrated.

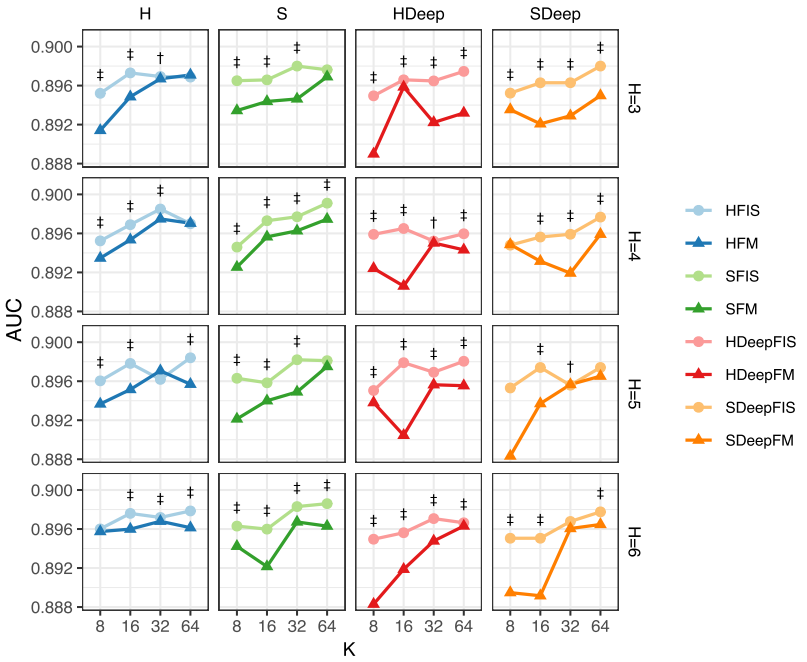

Fig. 9. Pairwise comparison of CTR prediction on the ML1M dataset w.r.t. AUC. We put a label above the line of BH-FISs if the improvement is significant. † and ‡ indicate the significance with p-value less than 0.05 and 0.01, respectively.

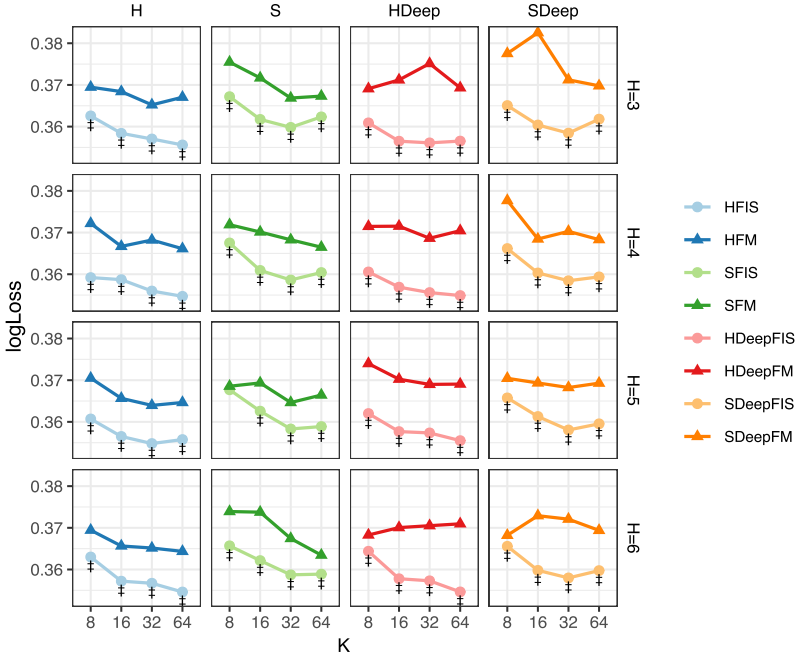

Fig. 10. Pairwise comparison of CTR prediction on the MLHt dataset w.r.t. logLoss.

Some trends can be observed: (1) BH-FISs are more likely to outperform HOFMs as H and K increase. This is because an increase in H and K increases the model complexity and BH-FIS helps to avoid overfitting; (2) the performance of BH-FISs also improves generally with the growth of H, showing that selecting feature interactions is more effective when modeling higher-order feature interactions; and (3) with the growth of H, H-FISs without DNNs show better performance than with DNNs, indicating that modeling higher-order feature interactions complements DNNs or even provides better modeling than DNNs.

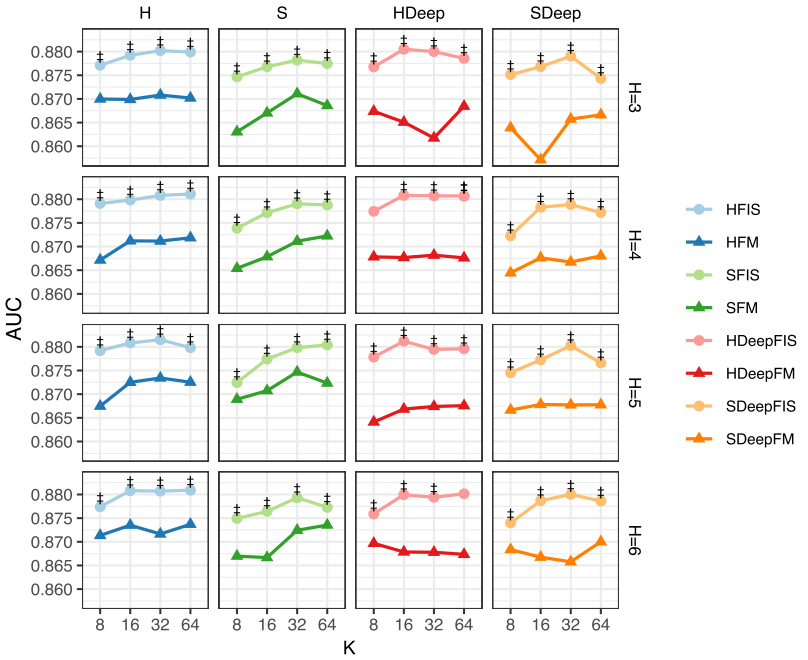

Fig. 11. Pairwise comparison of CTR prediction on the MLHt dataset w.r.t. AUC.

Table 11
Case study on interaction selection.

Order	Field Interactions	Feature Interactions
2^{nd}	⟨Gender, Genres⟩	⟨female, Romance⟩ ⟨female, Drama⟩ ⟨male, Action⟩
	⟨User, Genres⟩	⟨Alice, Comedy⟩ ⟨Alice, Crime⟩ ⟨Bob, Fantasy⟩
3^{rd}	⟨Gender, Occupation, Genres⟩	⟨female, college/grad student, Thriller⟩ ⟨female, self-employed, Animation⟩ ⟨male, programmer, Fantasy⟩
	⟨*User*, *Genres*, *Genres*⟩	⟨Alice, Action, Comedy⟩ ⟨Alice, Action, Adventure⟩ ⟨Bob, Animation, Fantasy⟩

Compared with the results on the ML1M dataset, the superiority of BH-FIS is much more evident on the MLHt dataset (Fig. 10 and 11). BH-FISs significantly outperform HOFMs in most parameter settings. Interestingly, HDeepFIS achieves the best results in most cases when H is small; when H increases, HDeepFIS can be gradually outperformed by HFIS; when $H = 6$, all the best results are generated by HFIS. Different from the baseline methods, which always benefit from adding DNNs, adding DNNs is only useful for H-FIS when H is small. This observation is consistent with our observations on the ML1M dataset. When H is large, H-FIS can already model higher-order feature interactions well and therefore adding extra DNNs does not contribute.

To sum up, (1) BH-FISs consistently outperform HOFMs, showing the effectiveness of H-FIS; (2) when modeling higher-order feature interactions, the effectiveness of BH-FIS becomes more evident; and (3) adding DNNs is helpful only when modeling lower-order feature interactions, which suggests that BH-FISs can already well capture feature interactions when higher-order is modeled.

We further conduct a case study on the ML1M dataset to show the selected feature interactions. ML1M contains 7 fields: “User”, “Movie”, “Gender”, “Age”, “Occupation”, “Zip” and “Genres”. Each field contains multiple unique features. We compare the field interactions selected by AutoFIS and AutoHash with the feature interactions selected by BH-FIS.

Results are shown in Table 11, which is categorized by different orders and only two of each order is shown. First look at the interaction selection on 2^{nd}-order. We report two of the field interactions selected by AutoFIS, i.e., ⟨Gender, Genres⟩ and ⟨Gender, Genres⟩. Since AutoFIS identifies field interactions only, it fails to tell what makes the interaction important. In comparison, BH-FIS can further identify the feature interactions, where each feature belongs to one of the field. As shown

in Table 11, BH-FIS finds that the interaction between "female" and "Romance" is important, which justifies why the interaction between "Gender" and "Genres" is important. We then look at the interaction selection on 3rd-order. Similarly, AutoFIS identifies field interaction like ⟨Gender, Occupation, Genres⟩ but fails to explain. BH-FIS selects the feature interactions to provide detailed explanation. It is worth noting that the field interaction $\langle User, Genres, Genres\rangle$ is unable to be identified by AutoFIS, because each field appears in the interaction once. "Genres" is a multivalent field, which can take multiple features. For example, the movie "Titanic" is both a "Romance" and "Drama". BH-FIS can identify the interaction ⟨Alice, Action, Adventure⟩, which suggests the interaction $\langle User, Genres, Genres\rangle$.

In short, BH-FIS can identify feature interactions under the field interactions, which provide detailed explanation for the prediction. It can also find features within the same field which cannot be discovered by AutoFIS and AutoHash.

7. Related work

In this section, we survey related work on factorization machines, feature interactions and feature selection, respectively.

7.1. Factorization machines

FMs have been widely studied and are commonly used in practical systems for their effectiveness and flexibility. Early studies [50,54] showed the generality of FMs. In contrast to matrix factorization (MF) [60] or tensor factorization (TF) [55], which model interactions between categorical variables only, FMs provide a generic way to model interactions between any real valued features. Rendle [50] showed that FMs can mimic many of the most successful factorization models (including MF, parallel factor analysis, and SVD++ [31]).

Recently, successive variants of FMs have been developed [1,4,5,24,37,39]. They have achieved promising performance in different recommendation scenarios [20,44,47,48,76]. Juan et al. [24] proposed the field-aware factorization machine (FFM) to factorize the interactions of fields (the category of features). Blondel et al. [4] presented the HOFM, which provides an efficient algorithm to train FMs with higher-order interactions. Besides rating prediction, FMs have also been optimized for the top-N recommendation task. Yuan et al. [76] introduced boosted factorization machines (BoostFMs) to incorporate contextual information into FMs for context-aware recommendation (CAR) [42,44,54]. Xiao et al. [72] proposed the attentional factorization machine (AFM), which uses an attention network to learn the importance of each feature interaction. To investigate the linear expressiveness limitation of FMs, He and Chua [20] proposed neural factorization machines (NFMs), which perform non-linear transformations on the latent space of second-order feature interactions.

Despite the effectiveness of modeling various feature interactions, existing FM variants suffer from the high-dimensionality issue which limits their application in high-dimensional scenarios. While FIS has been primarily investigated by SparseFMs [47,73,79], they focus on selecting pair-wise feature interactions only, which limits their generality.

On top of previous work on FMs we supplement a comprehensive study of FIS for FMs: we consider BP-FIS for FMs and further investigate BH-FIS for HOFMs.

7.2. Feature interaction

Besides the interaction between users and items, many prediction tasks also consider feature interactions. linear regression (LR) is a linear approach, which can only model first-order interactions on the linear combination of individual features. factorization machines (FMs) [50] learn second-order feature interactions from inner products of vector. Higher-order feature interactions have also been considered, with the development of efficient evaluation algorithms [4,29].

Higher-order feature interactions are approximated by NNs [18,35]. For example, NFM [20] stacks deep neural networks on top of the output of second-order feature interactions to model higher-order features, which is an implicit manner to pursue higher-order feature interactions. Similarly, the product-based neural network (PNN) [49], the factorization machine supported neural network (FNN) [78], DeepCrossing [58], Wide&Deep [13] and DeepFM [18] all relied on feed-forward neural networks (NNs) to model higher-order feature interactions. Some of them are proposed to model higher-order feature interactions explicitly. Deep&Cross [68] and CIN [35] take outer product of features at the bit- and vector-wise level. Some tree-based methods [69,80,82] combine the power of embedding-based models and tree-based models to capture high-order feature interactions.

Existing NN-based approximations failed to distinguish useful from harmful feature interactions. They can only approximate the prediction with the combination of all feature interactions. Although attention modules are now being employed to distinguish feature interactions [59,62,72], a recent study has shown that attention weights may be inconsistent with the importance of features [21,57].

On top of previous work on feature interaction, our work makes up a parallelizable way to exactly model high-order feature interactions, with H-FIS to ensure efficiency and avoid overfitting.

7.3. Feature selection

An effective approach to alleviate the higher-dimensionality issue of FMs is feature selection. Cheng et al. [12] proposed to select feature interactions though a greedy interaction feature selection algorithm based on gradient boosting. Xu

et al. [73] applied group Lasso [77] to user and item feature embeddings to select interactions between user and item features. Zhao et al. [79] proposed to select meta-graph based features. Similarly, they also applied group Lasso for feature selection. Mao et al. [40] proposed to select context features for FMs. They first choose features based on predictive power. Then, they subsample the set of features selected in the first step.

Feature selection has also been well investigated for feature-based recommender systems (FRSs), which often utilize high-dimensional auxiliary information. Ronen et al. [56] proposed to select content features. Their algorithm selects the most informative features by computing relevance scores based on pluggable feature similarity functions. Koenigstein and Paquet [30] proposed to select content features for Xbox movies by incorporating sparsity priors on feature parameters. Different feature weighing methods have also been proposed to select context features [81]. Li et al. [34] studied personalized feature selection for unsupervised learning; they learn a specific model for each user, which is only applicable with a limited number of users.

On top of previous work on feature selection, we systematically study feature interaction selection (FIS) for FMs, which is a challenging but important task.

8. Conclusion

In this paper, we studied the problem of feature interaction selection to improve the performance of factorization machines (FMs). We propose novel Bayesian generative models, a Bayesian variable selection method, to select personalized and higher-order feature interactions. For the efficiency of optimization, inference models have been developed with factored parameters. Specifically, we propose BP-FIS to select *personalized pairwise feature interactions*. On top of BP-FIS, we further propose BH-FIS, which addressed the problem of selecting *higher-order feature interactions*. To efficiently optimize BH-FIS, we reparameterize the sampling procedure and formulate inference networks. We also devise an iterative procedure with outer-product and masking to enumerate higher-order feature interactions comprehensively and non-repetitively.

Extensive empirical evaluations are carried out on three tasks in the field of artificial intelligence, to show that our proposed methods outperform FMs and state-of-the-art baselines on five datasets. We have found that (1) our proposed BP-FIS and BH-FIS can effectively improve the performance of FMs for recommendation and prediction; (2) the improvement achieved by incorporating FIS for FMs is significant, especially when FMs have large parameters spaces; (3) while incorporating non-linear layers on top of embeddings is effective, the combination with DNNs for BH-FIS is not helpful, especially when higher-order feature interactions are modeled; and (4) the superiority of FIS is more evident when modeling higher-order feature interactions, where the NN-based approximated baselines fail to reproduce the superiority.

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be constructed as influencing the position presented in, or the review of, the manuscript entitled.

Acknowledgements

This research was partially supported by the National Natural Science Foundation of China (NSFC) under grants 61806035, 62172136, U1936217, 61725203, 61732008, and by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Appendix A. Implementation details

We implement PFIS, PNFIS, HFIS, SFIS, HNFIS, SNFIS, HDeepFIS, SDeepFIS based on PyTorch.[13] We also implement SparseFM, HFM, SFM, HNFM, SNFM, HDeepFM, SDeepFM since there are no ready-to-use implementations. We use the TensorFlow[14] implementation in [20] to perform experiments for FM and NFM. We run LightFM using the python library.[15] LightFM is similar to FM. The difference is that LightFM only consider the interaction between user features and item features. The interactions among user features or item features are ignored. Since our motivation is to study FIS on generic FMs, we learn to select feature interactions rather than selecting feature interactions manually according to domain knowledge. We run GPFM and GPPW using the source code implementation by the author [44].[16] While GPFM and GPPW consider

[13] https://pytorch.org/.
[14] https://www.tensorflow.org/.
[15] https://github.com/lyst/lightfm.
[16] http://trungngv.github.io/gpfm/.

non-linear transformation over FMs, they lag behind state-of-the-art methods, e.g. NFM. Besides, training GPFM and GPPW is time-consuming. Training one model of GPFM on ML1M takes about a week. Therefore, we only show results of GPFM and GPPW on the ML1M dataset. For other compared methods, we run experiments using DeepCTR.[17]

For the network structure of NFM, HNFM, SNFM, PNFIS and HNFIS and SNFIS, we follow [20] and set identical dimensions for hidden layers. We use one hidden layer, which has been shown to generate the best performance empirically [20]. We use ReLU as the activation function. For CIN and AutoInt, we also consider the order H, which indicates the number of layers in the neural network. This is because each layer of CIN and AutoInt models feature interactions explicitly and with more layers higher-order feature interactions can be modeled. For models that are combined with DNNs, we fix the DNN as a two-layer perceptron with 200 and 100 neurons per layer, respectively.

We set $\pi_1 = \pi_2 = 0.5$ for PFIS and PNFIS as we presume no prior knowledge about the selection. Similarly, we also set $\pi^{(t)} = 0.5$ for H(S)FIS, HNFIS, SNFIS, HDeepFIS and SDeepFIS. Other hyper-parameters of each model are tuned by grid-search on the validation set. We tune the ℓ_2-norm regularization parameter for FM in $1e^{-6}, 5e^{-6}, \ldots, 1e^{-1}$. Similarly, we tune the $\ell_{2,1}$-norm regularization parameter for SparseFM in $1e^{-6}, 5e^{-6}, \ldots, 1e^{-1}$. For NFM, we fix the dropout rate of the hidden layers as 0.8 and tune the dropout rate for Bi-interaction layer in $0.1, 0.2, \ldots, 1.0$. For AFM, we use dropout for the embedding layer, which is searched from $0.1, 0.2, \ldots, 1.0$. As suggested by the author, we tune the ℓ_2-norm regularization parameter for the attention layer in $0, 0.5, 1, 2, 4, 8, 16$. For both HOFMs and H-FISs models, we consider feature interactions from 2^{nd}-order to 6^{th}-order ($H = 2, 3, \ldots, 6$). Feature embeddings are randomly initialized following $\mathcal{N}(0, 0.01)$. We tune the parameter K (the latent dimension of feature embeddings) differently for different tasks, where for recommendation we tune K from $64, 128, 256$ and for prediction we tune from $8, 16, 32, 64$. The main reason behind is that for prediction tasks (rating prediction and CTR-prediction), higher-order feature interactions are considered, where the time complexity of the compared methods are higher than that in the top-N recommendation task. Therefore, we limit the maximum embedding size in prediction tasks as 64 to ensure the efficiency.

For a fair comparison, we have the following identical experimental settings for all compared methods. All models are optimized using Adam [27]. Adam computes individual adaptive learning rates for different parameters. Therefore, we do not need to tune the learning rate. In practice, setting the initial learning rate as 0.001 provides a good default value. We set the maximum training epochs to 50. We apply early-stop for all methods, where we stop training if no further performance gain is observed for 4 successive epochs.

References

[1] M. Aharon, N. Aizenberg, E. Bortnikov, R. Lempel, R. Adadi, et al., Off-set: one-pass factorization of feature sets for online recommendation in persistent cold start settings, in: Proceedings 7th ACM Conference on Recommender Systems, Hong Kong, China, October 12-16, 2013, RecSys '13, ACM, 2013, pp. 375–378.

[2] T. Alshaybawee, R. Alhamzawi, H. Midi, I.I. Allyas, Bayesian variable selection and coefficient estimation in heteroscedastic linear regression model, J. Appl. Stat. 45 (2018) 2643–2657, https://doi.org/10.1080/02664763.2018.1432576.

[3] C. Archambeau, F.R. Bach, Sparse probabilistic projections, in: Proceedings 22nd Conference on Neural Information Processing Systems, NIPS '08, Vancouver, British Columbia, Canada, 2008, pp. 73–80, http://papers.nips.cc/paper/3380-sparse-probabilistic-projections.

[4] M. Blondel, A. Fujino, N. Ueda, M. Ishihata, Higher-order factorization machines, in: Proceedings 30th Conference on Neural Information Processing Systems, NIPS '16, Barcelona, Spain, 2016, pp. 3351–3359, http://papers.nips.cc/paper/6144-higher-order-factorization-machines.pdf.

[5] M. Blondel, M. Ishihata, A. Fujino, N. Ueda, Polynomial networks and factorization machines: new insights and efficient training algorithms, in: Proceedings 33rd International Conference on Machine Learning, ICML '16, New York City, NY, USA, 2016, pp. 850–858, http://proceedings.mlr.press/v48/blondel16.html.

[6] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao, Y. Yu, Collaborative personalized tweet recommendation, in: Proceedings 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, Portland, OR, USA, 2012, pp. 661–670.

[7] W. Chen, W. Xiong, X. Yan, W.Y. Wang, Variational knowledge graph reasoning, in: Proceedings 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT '18, New Orleans, Louisiana, USA, 2018, pp. 1823–1832.

[8] Y. Chen, P. Ren, Y. Wang, M. de Rijke, Bayesian personalized feature interaction selection for factorization machines, in: Proceedings 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '19, Paris, France, 2019, pp. 665–674.

[9] Y. Chen, X. Zhao, M. de Rijke, Top-n recommendation with high-dimensional side information via locality preserving projection, in: Proceedings 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, 2017, pp. 985–988.

[10] Y. Chen, X. Zhao, X. Lin, Y. Wang, D. Guo, Efficient mining of frequent patterns on uncertain graphs, IEEE Trans. Knowl. Data Eng. 31 (2019) 287–300, https://doi.org/10.1109/TKDE.2018.2830336.

[11] Y. Chen, X. Zhao, J. Liu, B. Ge, W.M. Zhang, Item cold-start recommendation with personalized feature selection, J. Comput. Sci. Technol. 35 (2020) 1217–1230, https://doi.org/10.1007/s11390-020-9864-z.

[12] C. Cheng, F. Xia, T. Zhang, I. King, M.R. Lyu, Gradient boosting factorization machines, in: Proceedings 8th ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA, 2014, pp. 265–272.

[13] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, H. Shah, Wide & deep learning for recommender systems, in: Proceedings 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys '16, Boston, MA, USA, 2016, pp. 7–10.

[14] N.H. Choi, W. Li, J. Zhu, Variable selection with the strong heredity constraint and its oracle property, J. Am. Stat. Assoc. 105 (2010) 354–364, https://doi.org/10.1198/jasa.2010.tm08281.

[15] P. Dellaportas, J.J. Forster, I. Ntzoufras, On bayesian model and variable selection using MCMC, Stat. Comput. 12 (2002) 27–36, https://doi.org/10.1023/A:1013164120801.

[16] E.R. DeLong, D.M. DeLong, D.L. Clarke-Pearson, Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach, Biometrics (1988) 837–845, https://doi.org/10.2307/2531595.

[17] https://github.com/shenweichen/DeepCTR.

[17] C. Févotte, S.J. Godsill, Sparse linear regression in unions of bases via bayesian variable selection, IEEE Signal Process. Lett. 13 (2006) 441–444, https://doi.org/10.1109/LSP.2006.873139.
[18] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, Deepfm: a factorization-machine based neural network for CTR prediction, in: Proceedings 26th International Joint Conference on Artificial Intelligence, IJCAI '17, Melbourne, Australia, 2017, pp. 1725–1731.
[19] F.M. Harper, J.A. Konstan, The movielens datasets: history and context, ACM Trans. Interact. Intell. Syst. 5 (2015) 19:1–19:19, https://doi.org/10.1145/2827872.
[20] X. He, T. Chua, Neural factorization machines for sparse predictive analytics, in: Proceedings 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, Shinjuku, Tokyo, Japan, 2017, pp. 355–364.
[21] S. Jain, B.C. Wallace, Attention is not explanation, in: Proceedings 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT '19, Minneapolis, MN, USA, 2019, pp. 3543–3556.
[22] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Trans. Inf. Syst. 20 (2002) 422–446, https://doi.org/10.1145/582415.582418.
[23] Y. Juan, D. Lefortier, O. Chapelle, Field-aware factorization machines in a real-world online advertising system, in: Proceedings 26th International Conference on World Wide Web Companion, WWW '17, Perth, Australia, 2017, pp. 680–688.
[24] Y. Juan, Y. Zhuang, W. Chin, C. Lin, Field-aware factorization machines for CTR prediction, in: Proceedings 10th ACM Conference on Recommender Systems, RecSys '16, ACM, Boston, MA, USA, 2016, pp. 43–50.
[25] S. Kabbur, X. Ning, G. Karypis, FISM: factored item similarity models for top-n recommender systems, in: Proceedings 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD '13, Chicago, IL, USA, 2013, pp. 659–667.
[26] G. Karypis, Evaluation of item-based top-n recommendation algorithms, in: Proceedings 10th ACM International Conference on Information and Knowledge Management, CIKM '01, Atlanta, Georgia, USA, 2001, pp. 247–254.
[27] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings 3rd International Conference on Learning Representations, ICLR '15, San Diego, CA, USA, 2015, http://arxiv.org/abs/1412.6980.
[28] D.P. Kingma, M. Welling, Auto-encoding variational bayes, in: Proceedings 2nd International Conference on Learning Representations, ICLR '14, Banff, AB, Canada, 2014, http://arxiv.org/abs/1312.6114.
[29] J. Knoll, Higher-order factorization machines: implementation, application, and comparison of a state-of-the-art recommender approach, Ph.D. thesis, University of Erlangen-Nuremberg, Germany, 2017, http://d-nb.info/1149295155.
[30] N. Koenigstein, U. Paquet, Xbox movies recommendations: variational bayes matrix factorization with embedded feature selection, in: Proceedings 7th ACM Conference on Recommender Systems, RecSys '13, ACM, 2013, pp. 129–136.
[31] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD '08, Las Vegas, Nevada, USA, 2008, pp. 426–434.
[32] M. Kula, Metadata embeddings for user and item cold-start recommendations, in: T. Bogers, M. Koolen (Eds.), Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems Co-Located with 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16–20, 2015, in: CEUR Workshop Proceedings, vol. 1448, CEUR-WS.org, 2015, pp. 14–21, http://ceur-ws.org/Vol-1448/paper4.pdf.
[33] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, J. Ma, Neural attentive session-based recommendation, in: Proceedings 26th ACM Conference on Information and Knowledge Management, CIKM '17, Singapore, 2017, pp. 1419–1428.
[34] J. Li, L. Wu, H. Dani, H. Liu, Unsupervised personalized feature selection, in: Proceedings 32nd AAAI Conference on Artificial Intelligence, AAAI '18, 2018, pp. 3514–3521, https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16288.
[35] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, G. Sun, xdeepfm: combining explicit and implicit feature interactions for recommender systems, in: Proceedings 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, SIGKDD '18, London, UK, 2018, pp. 1754–1763.
[36] D. Liang, R.G. Krishnan, M.D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings 2018 World Wide Web Conference on World Wide Web, WWW '18, Lyon, France, 2018, pp. 689–698.
[37] X. Lin, W. Zhang, M. Zhang, W. Zhu, J. Pei, P. Zhao, J. Huang, Online compact convexified factorization machine, in: Proceedings 27th World Wide Web Conference on World Wide Web, WWW '18, Lyon, France, 2018, pp. 1633–1642.
[38] B. Liu, C. Zhu, G. Li, W. Zhang, J. Lai, R. Tang, X. He, Z. Li, Y. Yu, Autofis: automatic feature interaction selection in factorization models for click-through rate prediction, in: R. Gupta, Y. Liu, J. Tang, B.A. Prakash (Eds.), KDD '20: the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, ACM, 2020, pp. 2636–2645.
[39] C. Lu, L. He, W. Shao, B. Cao, P.S. Yu, Multilinear factorization machines for multi-task multi-view learning, in: Proceedings 10th ACM International Conference on Web Search and Data Mining, WSDM '17, Cambridge, United Kingdom, 2017, pp. 701–709.
[40] X. Mao, S. Mitra, V. Swaminathan, Feature selection for fm-based context-aware recommendation systems, in: Proceedings 19th IEEE International Symposium on Multimedia, ISM '17, Taichung, Taiwan, 2017, pp. 252–255.
[41] P. McCullagh, J.A. Nelder, Generalized Linear Models, Springer, 1989, https://link.springer.com/chapter/10.1007/978-1-4757-3121-7_7.
[42] L. Mei, P. Ren, Z. Chen, L. Nie, J. Ma, J. Nie, An attentive interaction network for context-aware recommendations, in: Proceedings 27th ACM International Conference on Information and Knowledge Management, CIKM '18, ACM, 2018, pp. 157–166.
[43] T.J. Mitchell, J.J. Beauchamp, Bayesian variable selection in linear regression, J. Am. Stat. Assoc. 83 (1988) 1023–1032, https://doi.org/10.2307/2290129.
[44] T.V. Nguyen, A. Karatzoglou, L. Baltrunas, Gaussian process factorization machines for context-aware recommendations, in: Proceedings 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, Gold Coast, QLD, Australia, ACM, 2014, pp. 63–72.
[45] X. Ning, G. Karypis, SLIM: sparse linear methods for top-n recommender systems, in: Proceedings 11th IEEE International Conference on Data Mining, ICDM '11, Vancouver, BC, Canada, 2011, pp. 497–506.
[46] J.W. Paisley, D.M. Blei, M.I. Jordan, Variational bayesian inference with stochastic search, in: Proceedings 29th International Conference on Machine Learning, ICML '12, Edinburgh, Scotland, UK, 2012, http://icml.cc/2012/papers/687.pdf.
[47] Z. Pan, E. Chen, Q. Liu, T. Xu, H. Ma, H. Lin, Sparse factorization machines for click-through rate prediction, in: Proceedings 16th IEEE International Conference on Data Mining, ICDM '16, Barcelona, Spain, 2016, pp. 400–409.
[48] R. Pasricha, J. McAuley, Translation-based factorization machines for sequential recommendation, in: Proceedings 12th ACM Conference on Recommender Systems, RecSys '18, ACM, Vancouver, BC, Canada, 2018, pp. 63–71.
[49] Y. Qu, B. Fang, W. Zhang, R. Tang, M. Niu, H. Guo, Y. Yu, X. He, Product-based neural networks for user response prediction over multi-field categorical data, ACM Trans. Inf. Syst. 37 (2019) 5:1–5:35, https://doi.org/10.1145/3233770.
[50] S. Rendle, Factorization machines, in: Proceedings 10th IEEE International Conference on Data Mining, ICDM '10, Sydney, Australia, 2010, pp. 995–1000.
[51] S. Rendle, Factorization machines with libFM, ACM Trans. Intell. Syst. Technol. 3 (2012) 57:1–57:22, https://doi.org/10.1145/2168752.2168771.
[52] S. Rendle, Learning recommender systems with adaptive regularization, in: Proceedings 5th International Conference on Web Search and Web Data Mining, WSDM '12, Seattle, WA, USA, 2012, pp. 133–142.
[53] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: bayesian personalized ranking from implicit feedback, in: Proceedings Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, Montreal, QC, Canada, 2009, pp. 452–461.
[54] S. Rendle, Z. Gantner, C. Freudenthaler, L. Schmidt-Thieme, Fast context-aware recommendations with factorization machines, in: Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11, Beijing, China, 2011, pp. 635–644.

[55] S. Rendle, L. Schmidt-Thieme, Pairwise interaction tensor factorization for personalized tag recommendation, in: Proceedings 3rd International Conference on Web Search and Web Data Mining, WSDM '10, ACM, New York, NY, USA, 2010, pp. 81–90.
[56] R. Ronen, N. Koenigstein, E. Ziklik, N. Nice, Selecting content-based features for collaborative filtering recommenders, in: Proceedings 7th ACM Conference on Recommender Systems, RecSys '13, ACM, Hong Kong, China, 2013, pp. 407–410.
[57] S. Serrano, N.A. Smith, Is attention interpretable?, in: Proceedings 57th Conference of the Association for Computational Linguistics, ACL '19, Florence, Italy, 2019, pp. 2931–2951.
[58] Y. Shan, T.R. Hoens, J. Jiao, H. Wang, D. Yu, J.C. Mao, Deep crossing: web-scale modeling without manually crafted combinatorial features, in: Proceedings 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD '16, San Francisco, CA, USA, 2016, pp. 255–262.
[59] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, J. Tang, Autoint: automatic feature interaction learning via self-attentive neural networks, in: Proceedings 28th ACM International Conference on Information and Knowledge Management, CIKM '19, Beijing, China, 2019, pp. 1161–1170.
[60] N. Srebro, J.D.M. Rennie, T.S. Jaakkola, Maximum-margin matrix factorization, in: Proceedings 18th Conference on Neural Information Processing Systems, NeurIPS '04, 2004, pp. 1329–1336, http://papers.nips.cc/paper/2655-maximum-margin-matrix-factorization.
[61] I. Steinwart, A. Christmann, Support Vector Machines, Springer Science & Business Media, 2008, https://link.springer.com/book/10.1007/978-0-387-77242-4.
[62] Z. Tao, X. Wang, X. He, X. Huang, T.-S. Chua, Hoafm: a high-order attentive factorization machine for ctr prediction, Inf. Process. Manag. (2019) 102076, https://doi.org/10.1016/j.ipm.2019.102076.
[63] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. B 58 (1996) 267–288, http://www.jstor.org/stable/2346178.
[64] M.K. Titsias, M. Lázaro-Gredilla, Spike and slab variational inference for multi-task and multiple kernel learning, in: Proceedings 25th Conference on Neural Information Processing Systems, NIPS '11, Granada, Spain, 2011, pp. 2339–2347, http://papers.nips.cc/paper/4305-spike-and-slab-variational-inference-for-multi-task-and-multiple-kernel-learning.
[65] S. Tokui, I. Sato, Reparameterization trick for discrete variables, CoRR, arXiv:1611.01239 [abs], 2016.
[66] M. Trofimov, A. Novikov, tffm: tensorflow implementation of an arbitrary order factorization machine, https://github.com/geffy/tffm, 2016.
[67] M. Wang, X. Liu, X. Wu, Visual classification by ℓ_1-hypergraph modeling, IEEE Trans. Knowl. Data Eng. 27 (2015) 2564–2574, https://doi.org/10.1109/TKDE.2015.2415497.
[68] R. Wang, B. Fu, G. Fu, M. Wang, Deep & cross network for ad click predictions, in: Proceedings ADKDD'17, vol. 12, Halifax, NS, Canada, 2017, pp. 1–7.
[69] X. Wang, X. He, F. Feng, L. Nie, T. Chua, TEM: tree-enhanced embedding model for explainable recommendation, in: Proceedings 2018 World Wide Web Conference, WWW '18, Lyon, France, 2018, pp. 1543–1552.
[70] M. West, Bayesian factor regression models in the "large p, small n" paradigm, in: Bayesian Statistics, Oxford University Press, 2003, pp. 723–732, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.3036.
[71] Y. Wu, C. DuBois, A.X. Zheng, M. Ester, Collaborative denoising auto-encoders for top-n recommender systems, in: Proceedings Ninth ACM International Conference on Web Search and Data Mining, WSDM '16, San Francisco, CA, USA, 2016, pp. 153–162.
[72] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T. Chua, Attentional factorization machines: learning the weight of feature interactions via attention networks, in: Proceedings 26th International Joint Conference on Artificial Intelligence, IJCAI '17, Melbourne, Australia, 2017, pp. 3119–3125.
[73] J. Xu, K. Lin, P. Tan, J. Zhou, Synergies that matter: efficient interaction selection via sparse factorization machine, in: Proceedings 2016 SIAM International Conference on Data Mining, SDM '16, Miami, Florida, USA, 2016, pp. 108–116.
[74] N. Xue, B. Liu, H. Guo, R. Tang, F. Zhou, S.P. Zafeiriou, Y. Zhang, J. Wang, Z. Li, Autohash: learning higher-order feature interactions for deep ctr prediction, IEEE Trans. Knowl. Data Eng. (2020) 1, https://doi.org/10.1109/TKDE.2020.3016482.
[75] M. Yamada, W. Lian, A. Goyal, J. Chen, K. Wimalawarne, S.A. Khan, S. Kaski, H. Mamitsuka, Y. Chang, Convex factorization machine for toxicogenomics prediction, in: Proceedings 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD '17, Halifax, NS, Canada, 2017, pp. 1215–1224.
[76] F. Yuan, G. Guo, J.M. Jose, L. Chen, H. Yu, W. Zhang, Boostfm: boosted factorization machines for top-n feature-based recommendation, in: Proceedings 22nd International Conference on Intelligent User Interfaces, IUI '17, Limassol, Cyprus, 2017, pp. 45–54.
[77] M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables, J. R. Stat. Soc., Ser. B, Stat. Methodol. 68 (2006) 49–67, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.79.2062.
[78] W. Zhang, T. Du, J. Wang, Deep learning over multi-field categorical data – a case study on user response prediction, in: Proceedings 38th European Conference on IR Research, ECIR '16, Padua, Italy, 2016, pp. 45–57.
[79] H. Zhao, Q. Yao, J. Li, Y. Song, D.L. Lee, Meta-graph based recommendation fusion over heterogeneous information networks, in: Proceedings 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD '17, Halifax, NS, Canada, 2017, pp. 635–644.
[80] Q. Zhao, Y. Shi, L. Hong, GB-CENT: gradient boosted categorical embedding and numerical trees, in: Proceedings 26th International Conference on World Wide Web, WWW '17, Perth, Australia, 2017, pp. 1311–1319.
[81] Y. Zheng, R.D. Burke, B. Mobasher, Recommendation with differential context weighting, in: Proceedings 21th International Conference on User Modeling, Adaptation, and Personalization, UMAP '13, Rome, Italy, 2013, pp. 152–164.
[82] J. Zhu, Y. Shan, J.C. Mao, D. Yu, H. Rahmanian, Y. Zhang, Deep embedding forest: forest-based serving with deep embedding features, in: Proceedings 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD '17, Halifax, NS, Canada, 2017, pp. 1703–1711.