Link Prediction Using Higher-Order Feature Combinations across Objects

Kyohei ATARASHI†(a), Nonmember, Satoshi OYAMA†(b), and Masahito KURIHARA†(c), Members

SUMMARY Link prediction, the computational problem of determining whether there is a link between two objects, is important in machine learning and data mining. Feature-based link prediction, in which the feature vectors of the two objects are given, is of particular interest because it can also be used for various identification-related problems. Although the factorization machine and the higher-order factorization machine (HOFM) are widely used for feature-based link prediction, they use feature combinations not only across the two objects but also from the same object. Feature combinations from the same object are irrelevant to major link prediction problems such as predicting identity because using them increases computational cost and degrades accuracy. In this paper, we present novel models that use higher-order feature combinations only across the two objects. Since there were no algorithms for efficiently computing higher-order feature combinations only across two objects, we derive one by leveraging reported and newly obtained results of calculating the ANOVA kernel. We present an efficient coordinate descent algorithm for proposed models. We also improve the effectiveness of the existing one for the HOFM. Furthermore, we extend proposed models to a deep neural network. Experimental results demonstrated the effectiveness of our proposed models.

key words: link prediction, higher-order feature combinations, bilinear model, factorization machines, matrix factorization

1. Introduction

Link prediction is the computational problem of determining whether two given objects are linked. In a common setting, an adjacency matrix with missing values for an objective network is given, and the task is predicting the missing values. In this paper, we consider a feature-based link prediction problem in which the feature vectors of two objects are given. Feature-based link prediction is used in a more general setting because feature-based link prediction can be applied not only to the common adjacency-matrix-based (in other words, index-based) link prediction problem by regarding the indices of objects as features but also to many other identification-related problems: face verification by using two facial images, disambiguation of two author names in two different papers by using words in titles and names of co-authors, link prediction for a social network by using member features, and so on. While index-based link prediction methods cannot predict links between an unknown (completely new) object and unknown or known objects, feature-based link prediction methods can do it as long as the feature vectors are given.

Models using second-order feature combinations are effective for feature-based link prediction [1]–[7]. The higher-order factorization machine (HOFM) [8], which is an extension of the factorization machine (FM) [4], [7] that enables higher-order feature combinations, outperforms models using second-order feature combinations. It has thus been attracting the attention of many machine learning researchers. However, in feature-based link prediction, the HOFM uses higher-order feature combinations not only across the two objects but also from the same object. Feature combinations from the same object are irrelevant to major link prediction problems such as predicting identity (face verification, author name disambiguation, and so on) because it is not reasonable to determine whether two objects are the same from the features of only one object. As an efficient method for computing feature combinations only across two objects is not available, a model is needed that uses feature combinations only across two objects. We have developed models that use higher-order feature combinations only across two objects.

The contributions of this paper are as follows:

- We derive an algorithm for efficiently computing the sum of higher-order feature combinations only across two objects.
- We present a model that uses feature combinations only across two objects and can be efficiently evaluated using the proposed algorithm.
- We present an efficient coordinate descent (CD) algorithm. Keys of our algorithm’s efficiency make the CD algorithm for the HOFM faster than the original CD algorithm proposed by [8].
- We also propose deep-neural-network-extensions of our proposed models.
- We describe and discuss the relationships among proposed models and existing models on the basis of representabilities of these models.

This paper is organized as follows. In Sect. 2 and 3, we explain existing methods [1]–[14]. We present our contributions described above in Sect. 4. Experimental results are shown in Sect. 5 and finally we conclude in Sect. 6.
1.1 Problem Formulation and Notation

Feature-based link prediction is the computational problem of determining whether two objects, $a \in \mathbb{R}^{d_1}$ and $b \in \mathbb{R}^{d_2}$, are linked or not. Therefore, our goal is to obtain classifier $f : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \rightarrow \mathbb{R}$ such that $f(a, b) \geq 0$ if two objects are linked and $f(a, b) < 0$ otherwise. Supervised learning approaches have been used to obtain accurate classifier $f$. With these approaches, the user first collects a training set, i.e., a set of labeled pairs of objects $\mathcal{D} = \{(a_1, b_1, t_1), \ldots, (a_N, b_N, t_N)\}$, where $t_i \in \{-1, 1\}$ is the label of the $i$-th pair of objects, $t_i = 1$ means the two objects are linked, and $t_i = -1$ means they are not linked. The machine learning user next inputs $\mathcal{D}$ into the supervised learning algorithm to obtain classifier $f$.

**Notation.** We use $[M, N]$ to denote the set $\{M, M + 1, \ldots, N - 1, N\}$ and use $[N]$ when $M = 1$. Given $x \in \mathbb{R}^d$, we use $x_{i,j}$ to denote the $d - 1$ dimensional vector with $x_j$ removed. We use $(\cdot, \cdot)$ to denote the standard inner (dot) product. Given $a \in \mathbb{R}^{d_1}$ and $b \in \mathbb{R}^{d_2}$, we use $a \otimes b \in \mathbb{R}^{d_1 \times d_2}$, where $(a \otimes b)_{i,j} = a_i b_j$, to denote the tensor (outer) product of $a$ and $b$. We use $(a; b)$ to denote the concatenation of $a$ and $b$: $(a; b) = (a_1, \ldots, a_{d_1}, b_1, \ldots, b_{d_2})^\top \in \mathbb{R}^{d_1 + d_2}$.

2. Existing Methods Using Feature Combinations Across Objects

2.1 Feature Combinations Across Objects

As described above, classifier $f$ is obtained by using a supervised learning algorithm after collecting a training set. However, the feature vectors of pairs are needed in order to use conventional supervised learning algorithms and models. Because only feature vectors for each object are given in our feature-based link prediction setting, design feature vectors of the pairs are required for common algorithms and models. However, designing appropriate feature vectors is a difficult problem in general. Fortunately, second-order feature combinations across objects work effectively [1]–[3], [5], [6].

The vector representing feature combinations across $a$ and $b$ can be written as the tensor product of $a$ and $b$:

$$\text{vec}(b \otimes a) = (a_1 b_1, \ldots, a_1 b_{d_2}, \ldots, a_{d_1} b_1, \ldots, a_{d_1} b_{d_2})^\top. \quad (1)$$

It is possible to obtain $f$ enabling second-order feature combinations across object by using this feature vector as the feature vector of a pair, but the computational cost of computing this vector is $O(d_1 d_2)$, which is too high.

2.2 Kernel Method

[1] and [2] proposed the following kernel function for second-order feature combinations across objects:

$$\mathcal{P}^2((a, b), (a', b')) := \sum_{i,j} a_i a'_i b_j b'_j. \quad (2)$$

We call this kernel a pairwise kernel. Clearly it enables second-order feature combinations across objects and can be computed in $O(d_1 d_2)$ time by $(a, a') \cdot (b, b')$. Hence, when the number of training data $N$ is not so large, using a model (e.g., support vector machines (SVMs)) along with the pairwise kernel $\mathcal{P}^2$ enables the use of second-order feature combinations across objects without computing the feature vector of Eq. (1) efficiently.

Although using a second-order polynomial kernel for concatenating the feature vectors of two objects ($(a; b), (a'; b')^2$) enables the use of second-order feature combinations, then combinations from the same object, e.g., $a_1 a_2 a_1' a_2'$, are also included. They are irrelevant for such applications as predicting identity because it is not reasonable to determine whether two objects are the same from the features of only one object. Indeed, SVMs with a pairwise kernel outperformed ones with polynomial kernels in author name disambiguation [1].

2.3 Matrix Factorization Method

The linear regression model with $\text{vec}(a \otimes b)$ can be written in matrix form:

$$f_{\text{BM}}(a, b; W) := a^\top W b = \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} w_{i,j} a_i b_j, \quad (3)$$

where $W \in \mathbb{R}^{d_1 \times d_2}$ is the learnable parameter. We call this a bilinear model (BM). The size of the BM and the computational cost of evaluating the BM are $O(d_1 d_2)$, and these may be prohibitive. One proposed solution is factorization of $W$ [3], [5], [6]:

$$f_{\text{FBM}}(a, b; U, V) := a^\top UV^\top b = f_{\text{BM}}(a; b; UV^\top), \quad (4)$$

where $U \in \mathbb{R}^{d_1 \times k}$ and $V \in \mathbb{R}^{d_2 \times k}$ are the learnable parameters and $k \in \mathbb{N}$ is the rank hyper-parameter. We call this the factorized bilinear model (FBM). The size of the FBM and the computational cost of evaluating the FBM are $O(k(d_1 + d_2))$, which are acceptable.

3. Factorization Machines and Related Models

3.1 Factorization Machines and Higher-Order Factorization Machines

Rendle proposed the factorization machine (FM) [4], [7], which enables the use of second-order combinations of different features for common regression, classification, and ranking problems. Blondel et al. extended the FM to a higher-order FM (HOFM) [8]. The HOFM has higher representability than the FM because of the higher-order feature combinations. The model formula for an $m$-order HOFM is given by
\[
    f_{\text{HOFM}}^m(x; w, P^{(2)}, \ldots, P^{(m)})
    = \langle w, x \rangle + \sum_{t=2}^{m} \sum_{r=1}^{k_t} \mathcal{A}(p^{(t)}_r, x),
\]

(5)

where \( P^{(t)} \in \mathbb{R}^{d \times k_t} \) is the weight matrix for \( t \)-order feature combinations, \( p^{(t)}_r \) is the \( r \)-th row vector of \( P^{(t)} \), the \( k_t \in \mathbb{N} \) is a rank hyper-parameter, \( t \in [2, m] \) is the order of feature combinations considered, and \( \mathcal{A} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R} \) is the \( t \)-order ANOVA kernel [15], [16]:

\[
    \mathcal{A}(p, x) = \sum_{i_1 < \cdots < i_d} p_{i_1} x_{i_1} \cdots p_{i_d} x_{i_d}. \tag{6}
\]

Hereafter, we assume \( k_2 = \cdots = k_m = k \) for simplicity. When \( m = 2 \), the HOFM are equivalent to the FM. From Eq. (5) and (6), the HOFM uses only combinations of different features. The FM and HOFM learn well in a sparse setting because the weights for feature combinations are not independent from each other [4], [8]. From the following well-known recursion of the ANOVA kernel [8], [16], [17], \( \mathcal{A}(p, x) \) and the FM and HOFM are clearly multi-linear w.r.t \( p_1, \ldots, p_d, x_1, \ldots, x_d \):

\[
    \mathcal{A}^m(p, x) = \mathcal{A}(p_{-i, x_{-i}}) + p_j x_j \mathcal{A}^{m-1}(p_{-j, x_{-j}}). \tag{7}
\]

Blondel et al. proposed dynamic programming algorithms for evaluating and learning the HOFM efficiently [8]. For evaluating \( f_{\text{HOFM}}^m(x) \), their algorithm runs in \( O(n_r(x)km) \) time and \( O(dm) \) memory, where \( n_r(x) \) is the number of non-zero entries in \( x \). For updating all coordinates of \( P^{(m)} \) once on \( x \), their coordinate descent algorithm requires \( O(n_r(X)m^2k) \) time and \( O(Nm) \) memory.

The HOFM and FM have been used in recommender systems in which the feature vectors of each user and item are given [4], [17] and in feature-based link prediction [8] by using \( (a, b) \) as \( x \). In this case, the FM not only uses feature combinations across objects but also uses different-feature combinations from the same object: \( \{a_i a_j \mid i \neq j\} \cup \{b_i b_j \mid i \neq j\} \), so they are irrelevant to some problems like predicting identity.

3.2 Deep Neural Networks Based FMs

In the last decade, deep neural networks (DNNs) have achieved state-of-the-art performances in many tasks. With these success, several researches have proposed DNN-based FMs [9]–[13]. Several researchers have proposed deep neural network (DNN) [18] based FMs [9]–[13]. These researches introduced new layers that use feature combinations like FMs and proposed DNN-based models using them. [9] proposed neural factorization machines (NFM)s that use second-order feature combinations in the bi-interaction layer, which is a hidden layer using second-order feature combinations. Each unit in the bi-interaction layer is a second-order ANOVA kernel and NFM}s are DNNs using the bi-interaction layer as the first hidden layer. The main idea of other researches [10]–[12] are almost the same as that of [9] and these DNN-based FMs achieved better performance than that of the original FMs.

3.3 Polynomial Networks

[14] proposed polynomial networks (PNs), which are depth-2 neural networks with a polynomial as the activation function:

\[
    f_{\text{PN}}^2(x; w, P, \lambda) = \langle w, x \rangle + \sum_{s=1}^{k} \lambda_s \langle p_s, x \rangle^2, \tag{8}
\]

where \( w \in \mathbb{R}^d, P \in \mathbb{R}^{d \times k} \), and \( \lambda \in \mathbb{R}^k \) are the learnable parameters, and \( k \) is the number of hidden units. \( P \) is the weight matrix between the input and hidden layers, and \( \lambda \) is the weight vector between the hidden layer and the output layer. They showed that PNs can approximate neural networks with a sigmoidal activation function and that they provide an efficient convergence-guaranteed learning algorithm based on solving an eigenvalue problem. They also extended second-order PNs to third-order ones, which compris a subset of depth-3 PNs.

PNs also enable feature combinations because of the second term in Eq. (8). However, PNs with \( (a, b) \) as \( x \) in feature-based link prediction use the features of all combinations not only across the two objects but also from the same object: \( \{a_i a_j \} \cup \{b_i b_j\} \). These combinations are also irrelevant to such problems as predicting identity.

4. Higher-Order Feature Combinations across Two Objects

4.1 Basic Idea of Our Research

As mentioned above, the use of second-order feature combinations across objects is an effective approach in feature-based link prediction [1]–[3], [5], [6]. Although the HOFM [8] using higher-order feature combinations achieved better performance than the second-order FM [4], [7], the HOFM and FM use feature combinations from the same object, and these combinations are irrelevant to such problems as predicting identity. Therefore, models using higher-order feature combinations only across the two objects should outperform these models described above. Figure 1 summarizes relationships among the proposed methods: pairwise networks (PairNets), higher-order pairwise networks (HOPairNets), and higher-order pairwise deep neural networks (HOPairDNNs) that will be presented later and some existing methods.

4.2 Higher-Order Pairwise Kernel

We define the higher-order feature combinations across \( a \in \mathbb{R}^d \) and \( b \in \mathbb{R}^d \) as

\[
    \bigcup_{t=1}^{m-1} \{a_{i_1} \cdots a_{i_t} b_{j_1} \cdots b_{j_{m-t}} \mid i_1 < i_2 < \cdots < i_t \land j_1 < j_2 \land \forall t_1 < t_2\},
\]
that is, \(m\)-order feature combinations including at least one feature of both objects, and not including feature combinations from the same object. We also define a kernel using higher-order feature combinations across objects:

\[
P^m((u, v), (a, b)) = \sum_{i=1}^{m-1} \sum_{t=1}^{\mathcal{A}(u, a), \mathcal{A}^{m-t}(v, b)} \prod_{t=1}^{m-1} a_t, a_t, b_{j}, u_{j}, v_{j}, \lambda, \nu.
\]

(9)

where \(u \in \mathbb{R}^d_1\) and \(v \in \mathbb{R}^d_2\). When \(m = 2\), this equation is equivalent to the pairwise kernel in Eq. (2), so we call this kernel an \(m\)-order pairwise kernel.

Naive computation of an \(m\)-order pairwise kernel takes \(O\left(\sum_{i=1}^{m-1} d_i^2 d_i^{m-i}\right)\) time, which may be prohibitive. However, a clue to efficiently computing a higher-order pairwise kernel can be obtained from the following transformation:

\[
P^m((u, v), (a, b)) = \sum_{i=1}^{m-1} \mathcal{A}(u, a), \mathcal{A}^{m-t}(v, b).
\]

(10)

This equation shows that an \(m\)-order pairwise kernel can be computed in \(O(m)\) time when the series of the ANOVA kernels, \(\mathcal{A}(u, a)\) and \(\mathcal{A}(v, b)\), for \(t \in [m-1]\), are given. The ANOVA kernels \(\mathcal{A}(u, a)\) and \(\mathcal{A}(v, b)\) for \(t \in [m-1]\) are computed in \(O(md_1)\) and \(O(md_2)\) time and memory by previous algorithm [8]. Therefore, an \(m\)-order pairwise kernel can be computed in \(O(m(d_1 + d_2))\) time and \(O(m)\) memory. We show the procedure of this efficient algorithm for computing ANOVA kernels in Algorithm 1. The final value of \(a_t\) in Algorithm 1 is \(\mathcal{A}(p, x)\) for \(t \in [m]\). Therefore, not only \(\mathcal{A}(p, x)\) for \(t \in [m-1]\) is obtained in \(O(md_1)\) time and \(O(m)\) memory. This insight for the memory efficiency of recursion (7) can be useful for the optimization and it will be described in Sect. 4.4.

There are two important properties of the higher-order pairwise kernel. The first one is multi-linearity. It is derived from the multi-linearity of the ANOVA kernels and Eq. (10). The second one is homogeneity. Let \(\lambda \in \mathbb{R}\) and \(m, n \in \mathbb{N}_{\geq 2}\). Then,

\[
\lambda^m P^m((u, v), (a, b)) = P^m((\lambda u, \lambda v), (a, b)).
\]

(11)

When \(m = 2\), the following equation is also satisfied:

\[
\lambda^2 P^2((u, v), (a, b)) = P^2((\lambda u, v), (a, b)).
\]

(12)

It is derived from the homogeneity of the ANOVA kernel [17]: \(\lambda^m \mathcal{A}(p, x) = \mathcal{A}(\lambda p, x)\). From Eq. (10) and homogeneity of the ANOVA kernel, Eq. (11) is obtained:

\[
\lambda^m P^m((u, v), (a, b)) = \sum_{i=1}^{m-1} \lambda \mathcal{A}(u, a), \mathcal{A}^{m-i}(v, b)
\]

\[
= \sum_{i=1}^{m-1} \mathcal{A}(\lambda u, a), \mathcal{A}^{m-i}(\lambda v, b).
\]

(13)

Equation (12) is obtained in similar way: \(\lambda^2 P^2((u, v), (a, b)) = \lambda \mathcal{A}(u, a), \mathcal{A}(v, b) = \mathcal{A}(\lambda u, v), (a, b)\). It is used to discuss the representability of our proposed models.

4.3 Higher-Order Pairwise Network and Pairwise Network

We first propose a higher-order pairwise network (HOPairNet) that uses higher-order feature combinations only across the two objects. It is based on the definition of the HOFM and PN [14], [17]. The model formula for this model is given by

\[
f^{m}_{\text{HOPairNet}}(a, b; U^{(m)}, V^{(m)}, \Lambda^{(m)}) := \sum_{i=2}^{m} \sum_{s=1}^{k} \lambda^{s}_{i} P^s((u^{(i)}, v^{(i)}), (a, b)).
\]

(14)

where \(U^{(1)}, \ldots, U^{(m)} \in \mathbb{R}^{d_1 \times k}\), \(V^{(1)}, \ldots, V^{(m)} \in \mathbb{R}^{d_2 \times k}\), and \(\lambda^{(1)}, \ldots, \lambda^{(m)} \in \mathbb{R}^d\) are the learnable parameters, and \(U^{(m)}, V^{(m)}, \text{and} \Lambda^{(m)}\) are sets of them. An \(m\)-order HOPairNet clearly enables the use of from second to \(m\)-order feature combinations only across the two objects. We call a second-order HOPairNet a PairNet:

\[
f_{\text{Pair}}(a, b, \Lambda, U, V) := \sum_{i=1}^{k} \lambda^{i} P^i((u_i, v_i), (a, b)).
\]

(15)
The most important property of the HOPairNet and PairNet is multi-linearity w.r.t. $\lambda_1^{(i)}$, $u_1^{(i)}$, and $v_1^{(i)}$. It is easily derived from the multi-linearity of the higher-order pairwise kernel. It makes the objective function of the HOPairNet model multi-convex, enabling it to be efficiently optimized by using the CD algorithm, as with the HOFM.

With regards to the representability of the PairNet, [17] showed that when $m$ is odd, $\lambda = 1$ can be fixed without loss of generality in the FM and PN. Similar results are obtained with a model using a higher-order pairwise kernel.

**Lemma 1.** Representability of Pairwise Networks.

Let $f_{\text{pair}}(a, b; \lambda, U, V) := \sum_{i=1}^{F} \lambda_i f_{\text{pair}}((u_i, v_i), (a, b))$, $l : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \geq 0$ be a convex loss function, and $L_{\text{pair}}(\lambda, U, V) := \frac{1}{N} \sum_{n=1}^{N} l(t_n, f_{\text{pair}}(a_n, b_n))$. Then, if $m$ is even then $\lambda = 1$ can be fixed without loss of generality in the PairNet.

\[
\min_{\lambda, U, V} L_{\text{pair}}(\lambda, U, V) \leq \min_{U, V} L_{\text{pair}}(1, U, V),
\]

and otherwise (that is, if $m$ is odd or 2)

\[
\min_{\lambda, U, V} L_{\text{pair}}(\lambda, U, V) = \min_{U, V} L_{\text{pair}}(1, U, V).
\]

**Proof.** With the use the homogeneity of the higher-order pairwise kernel, Lemma 1 is derived in the same way as the result for the PN and FM ([17], Lemma 4). \quad \square

Because $f_{\text{pair}}$ is equivalent to the PairNet, $\lambda = 1$ can be fixed without loss of generality in the PairNet. Lemma 1 says that introducing $\lambda^{(i)}$ for even $t \geq 4$ improves the representability of the HOPairNet.

We now show that the PairNet is equivalent to the FBM from Lemma 1 and from the result of transformation to the matrix form of Eq. (15).

**Lemma 2.** Equivalence of the PairNet and the FBM.

For every PairNet $f_{\text{pair}}$, there exist an FBM $f_{\text{FBM}}$ such that $f_{\text{pair}}(a, b) = f_{\text{FBM}}(a, b)$ for all $a \in \mathbb{R}^{d_1}$, $b \in \mathbb{R}^{d_2}$.

**Proof.** We first show the matrix form of the model equation of the PairNet, which is given by

\[
f_{\text{pair}}(a, b; \lambda, U, V) = a^T U \text{diag}(\lambda^T) V^T b.
\]

From Lemma 1, $\lambda = 1$ can be fixed in the PairNet without loss of generality. When $\lambda = 1$, the PairNet is equivalent to the FBM: $f_{\text{pair}}(a, b; 1, U, V) = a^T U V^T b = f_{\text{FBM}}(a, b; U, V)$. \quad \square

Therefore, the HOPairNet can be also regarded as an higher-order generalization of the FBM. Furthermore, the following result for regularization in the BM, the PairNet, and the FBM is derived from Lemma 2, Eq. (4), and previous results of regularization in the PN and FM ([17], Theorem 2).

**Theorem 3.** Equivalence of regularized problems. Let $\| \cdot \|$ be the nuclear norm. Then,

\[
\min_{W} \frac{1}{N} \sum_{n=1}^{N} l(t_n, f_{\text{FBM}}) + \beta \|W\|_*,
\]

\[
= \min_{\lambda, U, V} \frac{1}{N} \sum_{n=1}^{N} l(t_n, f_{\text{pair}}) + \beta \sum_{i=1}^{k} |\lambda_i| \Omega(u_i, v_i),
\]

\[
= \min_{U, V} \frac{1}{N} \sum_{n=1}^{N} l(t_n, f_{\text{FBM}}) + \beta \left(\|U\|^2 + \|V\|^2\right),
\]

where $W \in \mathbb{R}^{d_1 \times d_2}$, $U \in \mathbb{R}^{d_1 \times k}$, $V \in \mathbb{R}^{d_2 \times k}$, $\lambda \in \mathbb{R}^k$, rank($W^*$) \leq k, $\Omega(u, v) := \|u\|^2 + \|v\|^2$, and $W^* = \arg\min_{W} \frac{1}{N} \sum_{n=1}^{N} l(t_n, f_{\text{FBM}}(a, b; W)) + \beta \|W\|_*$ (and we omit the input vectors for each model).

**Proof.** These results can be obtained in the same way as in ([17], Theorem 2). First, the value of loss term in (19) is equal to that in (21) when $W = UV^T$. Then, (19) = (21) is derived from the relationship between the nuclear norm and the Frobenius norm of the factorized matrix $\|W\|_* = \min_{U, V} \|U\|^2 + \|V\|^2 / 2$. (21) = (20) is derived from the following transformation of Eq. (18). Let $\lambda = (\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_t})$ and $\lambda_{\text{sign}} = (\text{sign}(\lambda_1), \ldots, \text{sign}(\lambda_t))$. Then,

\[
f_{\text{pair}}(a, b; \lambda, U, V) = a^T U \text{diag}(\lambda) V^T b
\]

\[
= a^T U \text{diag} (\lambda_{\text{sign}}) \text{diag}(\lambda_{\text{sign}}) V^T b.
\]

Let $\hat{U} = U \text{diag} (\lambda_{\text{sign}})$ and $\hat{V} = \text{diag}(\lambda_{\text{sign}}) V$. Then, substituting $\hat{U}$ and $\hat{V}$ as $U$ and $V$ in (21) results (20). \quad \square

Nuclear norm regularization has been used for obtaining low-rank solutions [19], [20], so a low-rank solution is expected for problem (20) for the PairNet. Although we cannot derive a theoretical result for the HOPairNet (i.e., the higher-order case), we can use the straightforward extension of Eq. (20) as the objective function,

\[
\frac{1}{N} \sum_{n=1}^{N} l(t_n, y_n) + \beta \sum_{i=1}^{k} \sum_{s=1}^{|t_i|} |\lambda_s^{(i)}| \Omega(u_s^{(i)}, v_s^{(i)}),
\]

where $y_i = f_{\text{HOPairNet}}(a_i, b_i)$, because problem (23) can be regarded as the least absolute shrinkage and selection operator (LASSO) [21] for $\lambda^{(i)}$ for $t \in [2, m]$ when $U^{(i)}$ and $V^{(i)}$ are fixed. Therefore, sparse solutions for $\lambda^{(i)}$ can be expected, and obtaining sparse solutions can be regarded as the selection of bases; that is, a low-rank solution can be expected. To be more precise, because $\lambda^{(i)} = 1$ can be fixed when $t$ is odd or 2 from Lemma 1, we only fit $\lambda^{(i)}$, which has even $t$ greater than 2. We use the CD algorithm with proximal operation [22] for optimizing such $\lambda^{(i)}$. It is easily done by caching $\Omega (u_s^{(i)}, v_s^{(i)}); (a_i, b_i)$ for all $s \in [k]$ and $i \in [n]$.

4.4 CD Algorithm for HOPairNets

As described above, optimization problems (20) and (23) are
multi-convex optimization problems. Hence, the two models proposed above can be efficiently optimized by using the CD algorithm. Here we describe its use for the HOPairNet that includes the PairNet (when $m = 2$). We assume that loss function $l$ is convex and $\mu$-smooth function. Similar to the CD algorithm for the HOFM [8], the update rule for $u_{j,s}^{(m)}$ is $u_{j,s}^{(m)} \leftarrow u_{j,s}^{(m)} - \eta_{j,s} \partial L/\partial u_{j,s}^{(m)}$, where $L$ is the objective function in Eq. (23) and $\eta_{j,s} = \mu \sum_{i=1}^{N} (\partial y_{i}/\partial u_{m}^{(i)})^{2} / [N + \beta_{i}^{(i)}]$. The update rule for $v_{j,s}^{(i)}$ is easily derived in a similar manner. For updating, one obviously must compute the partial gradient

$$\frac{\partial y_{i}}{\partial u_{j,s}^{(m)}} = \lambda_{i}^{m} \sum_{t=1}^{m-1} \frac{\partial \mathcal{A}(u_{i}^{(m)}, a_{i})}{u_{j,s}^{(m)}(t)} \mathcal{A}^{m-t}(v_{i}^{(m)}, b_{i}),$$

and it requires $\partial \mathcal{A}(u_{i}^{(m)}, a_{i})/\partial u_{j,s}^{(m)}$ for $t \in [m-1]$. One can compute it by existing algorithm proposed by Blondel et al. [8]. Then, the computational cost for updating all coordinates of $U^{(m)}$ and $V^{(m)}$ once is $O(m^{2}k(n_{2}(A) + n_{2}(B)))$ time and $O(Nm)$ memory, where $A$ and $B$ are matrices in which the $i$-th row vector is $a_{i}$ and $b_{i}$, respectively.

Here, we present a more efficient CD algorithm for the HOpairNet that takes only $O(mk(n_{2}(A) + n_{2}(B)))$ time for updating all the coordinates of $U^{(m)}$ and $V^{(m)}$ once. It is based on the insight for the memory efficiency of recursion (7) (i.e., Algorithm 1) and the following new recursion for calculating the partial gradient of the ANOVA kernel:

$$\frac{\partial \mathcal{A}^{m}(p,x)}{\partial p_{j}} = x_{j} \mathcal{A}^{m-1}(p_{j}, x_{j}) = x_{j} \mathcal{A}^{m-1}(p,x) - p_{j} \frac{\partial \mathcal{A}^{m-1}(p,x)}{\partial p_{j}}. \quad (24)$$

The advantage of recursion (24) is its reduced time complexity. When $\mathcal{A}(p,x)$ for $t \in [m]$ are given, $\partial \mathcal{A}(p,x)/\partial p_{j}$ for $t \in [m]$ can be computed in $O(m)$ time. The algorithm used for computing them using recursion (24) is shown in Algorithm 2. $\mathcal{A}(p,x)$ for $t \in [m]$ can be computed in $O(md)$ time and $O(m)$ memory with Algorithm 1. With Algorithms 1 and 2, $\mathcal{A}(p,x)$ and $\partial \mathcal{A}(p,x)/\partial p_{j}$ for $t \in [m]$ can be computed in $O(md)$ time and $O(m)$ memory while an algorithm proposed by Blondel et al. [8] takes $O(md + m^{2})$ time.

For an efficient implementation of the CD algorithm, we need a method for efficiently synchronizing prediction. Fortunately, the prediction is also synchronized in $O(m)$ time. Let $p_{j}^{\text{new}} = p_{j} - \Delta$ be the value after updating and $p_{j}^{\text{new}} = (p_{1}, \ldots, p_{j-1}, p_{j}^{\text{new}}, p_{j+1}, \ldots, p_{d})^{T}$. Then,

$$\mathcal{A}^{m} (p^{\text{new}}, x) = \mathcal{A}^{m} (p,x) - \Delta \frac{\partial \mathcal{A}^{m-1}(p,x)}{\partial p_{j}}. \quad (25)$$

From these results about the complexities of the computing partial gradient and synchronizing ANOVA kernels and predictions, using proposed algorithms can reduce the computational cost of the CD algorithm for the HOPairNet from $O(m^{2}k(n_{2}(A) + n_{2}(B)))$ to $O(mk(n_{2}(A) + n_{2}(B)))$ time. This improvement is easily applied to the CD algorithm for the HOFM.

4.5 Symmetrization

For some applications such as predicting identity, symmetry must be ensured; that is, $f(a,b) = f(b,a)$ must be satisfied. In such applications, the domains of $a$ and $b$ are the same, so $d_{1} = d_{2} = d$. We discuss here the method for ensuring symmetry in our proposed models. We first consider the symmetry of the PairNet. The most straightforward way of ensuring symmetry is parameter sharing: $U = V = P$. However, this parameter sharing does not preserve the multi-linearity of proposed models and thus we cannot optimize them efficiently.

Here, we use the relationship between the PairNet and the BM (Lemma 2 and Eq. (4)). The $f_{\text{BM}}(a,b;W)$ clearly satisfies the symmetry requirement when $W$ is a symmetric matrix. Because the PairNet can be regarded as a BM with $W = U \text{diag}(A)V^{T}$, the model $f_{\text{BM}}(a,b;U \text{diag}(A)V^{T} + V \text{diag}(A)U^{T})/2$ can be regarded as the symmetric PairNet. This technique is called the symmetrization trick [17] and preserves the multi-linearity of the model. A method for ensuring HOpairNet symmetry can be easily derived from the following transformation for the symmetric PairNet:

$$a^{T} \frac{1}{2} (U \text{diag}(A)V^{T} + V \text{diag}(A)U^{T}) b = \frac{1}{2} (f_{\text{Pair}}(a,b;\lambda,U,V) + f_{\text{Pair}}(b,a;\lambda,U,V)). \quad (26)$$

From this, a symmetrization method for the HOpairNet is derived: $\frac{1}{2} (f_{\text{HOPair}}^{m}(a,b) + f_{\text{HOPair}}^{m}(b,a))$.

4.6 Higher-Order Pairwise Deep Neural Networks

Finally, we present DNN-based models that enable higher-order feature conjunctions across objects. We first give the interpretation of the PairNet as a neural network (NN). Equation (15) can be rewritten:

$$f_{\text{Pair}}(a,b;\lambda,U,V) = \sum_{s=1}^{k} \lambda_{s}(\text{vec}(b \otimes a), \text{vec}(v_{s} \otimes u_{s})). \quad (27)$$

Therefore, the PairNet can be regarded as a depth-2 NN with $\text{vec}(b \otimes a)$ as input, an identity function as the activation function, $\text{vec}(v_{s} \otimes u_{s})$ as the weight between the input and $s$-th units in the hidden layer, $A$ as the weight between the hidden layer and the output unit, and $k$ is the number of
hidden units. We propose a pairwise deep neural network (PairDNN):

\[
\text{fPairDNN}(\mathbf{a}, \mathbf{b}; U, V, \Theta) := f_{\text{DNN}}(\sigma(z); \Theta),
\]

where \(\sigma(z) = (\sigma(z_1), \ldots, \sigma(z_k))\), \(z_s = \mathcal{P}^2((u_s, v_s), (a, b))\) for \(s \in [k]\), is the input of \(f_{\text{DNN}}()\), \(\sigma : \mathbb{R} \rightarrow \mathbb{R}\) is an element-wise activation function, \(f_{\text{DNN}} : \mathbb{R}^k \rightarrow \mathbb{R}\) is a DNN, and \(\Theta\) is the set of parameters for the DNN (we do not specify the architecture (form) of \(f_{\text{DNN}}\)). This modeling is an analogy of some existing DNN-extensions of FMs \([9]–[13]\). While the DNN can extract useful feature representation, introducing the layer using neural network (HOPairDNN) by defining other researchers \([24], [25]\). While the DNN can extract useful feature representation, introducing the layer using feature representations by some graph embedding vectors is equivalent to the generalized matrix factorization layer, which is used in the neural collaborative filtering \([26]\) that is a DNN-extension of the latent factor model.

In index-based link prediction setting, one can obtain some feature representations by some graph embedding methods \([29]\). Several researches showed that using both indices and features of objects could improve the performances \([4], [5], [30]\). Moreover, some researches showed the effectiveness of the bilinear pooling, which uses second-order feature combinations between extracted (embedded) feature vectors \([24], [27]\). Thus, it seems promising to use feature-based link prediction methods that use feature combinations (e.g., proposed methods) with both indices and embedded features of objects in index-based link prediction. We leave the investigation of it for future work.

There have been many other index-based link prediction methods and recently the method using graph NN \([31]\) has been proposed \([32]\). Since this is also a method for index-based link prediction, this method cannot predict links for unknown nodes although they can leverage additional features of nodes. On the other hand, our methods (and other methods for feature-based link prediction) can learn models without indices of objects and predict links for unknown objects.

4.7 Relationship between Proposed Methods and Existing Methods for Index-Based Link Prediction

As described in Sect. 1, while index-based link prediction methods cannot predict links between an unknown object and unknown or known objects, feature-based link prediction methods can do it as long as the feature vectors are given. Moreover, methods for feature-based link prediction can be applied to index-based link prediction by regarding the indices of objects as features; given indices of nodes \(a_{\text{ind}}\) and \(b_{\text{ind}}\), one-hot encoding vectors of indices can be used as feature vectors of nodes. Then, \(d_1\) and \(d_2\) correspond to the number of nodes. The model equation of the latent factor (feature) model \([28]\)\(^6\), which is a well-known method for index-based link prediction, \(f_{\text{latent}}(a_{\text{ind}}, b_{\text{ind}}) : [d_1] \times [d_2] \rightarrow \langle \mathbf{a}_{\text{ind}}, \mathbf{b}_{\text{ind}} \rangle + \mathbf{w}^{(a)} + \mathbf{w}^{(b)}\) where \(\mathbf{a}_i \in \mathbb{R}^{d_1} (i \in [d_1]), \mathbf{b}_j \in \mathbb{R}^{d_2} (j \in [d_2]), \mathbf{w}^{(a)} \in \mathbb{R}^{d_1}\) and \(\mathbf{w}^{(b)} \in \mathbb{R}^{d_2}\) are learnable parameters. Parameters \(\mathbf{a}_i\) and \(\mathbf{b}_j\) are called latent factors, and \(\mathbf{w}^{(a)}\) and \(\mathbf{w}^{(b)}\) are called biases. It is known that the FM for index-based link prediction (i.e., using one-hot encoding vectors) is equivalent to this latent factor model and the FM using both indices and additional features of objects outperformed the latent factor model in the recommendation task \([4]\). The PairNet for index-based link prediction is equivalent to the latent factor model without biases: \(f_{\text{PairNet}}(\mathbf{a}, \mathbf{b}; 1, U, V) = \langle \mathbf{a}_{\text{ind}}, \mathbf{b}_{\text{ind}} \rangle\). Furthermore, from this result, it is clear that the second-order pairwise network layer for one-hot encoding vectors is equivalent to the generalized matrix factorization layer, which is used in the neural collaborative filtering \([26]\) that is a DNN-extension of the latent factor model.

5. Evaluation

We evaluated the performance of our proposed models using three feature-based link prediction tasks: author disambiguation, co-author link prediction and recommendation.

5.1 Datasets

- **DBLP.** For the author disambiguation task, we extracted 3,384 papers in which there were 729 unique author names from the DBLP dataset \([23]\). Each paper was considered an object. If two papers had an author in common, we gave that pair of papers a positive label. We used all the words in the titles, the coauthor names, and the publication venues for creating bag-of-words feature vectors.

- **NIPS.** For the co-author link prediction task, we obtained a dataset of co-author graphs from the first 12 editions of the Neural Information Processing Systems Conference \([33]\). There were 2,037 authors in this dataset. If two authors had collaborated, we gave that pair of authors a positive label. Each object (author) was represented by a bag-of-words feature vector, which used words in their publications.

- **ML100K.** For the recommendation task, we ob-
Table 1

| Dataset | #train | #valid | #test | d1    | d2 |
|---------|--------|--------|-------|-------|----|
| DBLP    | 22,416 | 1,000  | 21,264| 9,264 | 49 |
| NIPS    | 3,134  | 134    | 3,000 | 13,649| 1,000|
| ML100K  | 21,200 | 1,000  | 20,200| 49    | 134|

Table 2

| Model | DBLP | NIPS | ML100K |
|-------|------|------|--------|
| HOPairNet (m = 2) | 0.7639 | 0.8567 | 0.5951 |
| HOPairNet (m = 3) | 0.7827 | 0.8570 | 0.6239 |
| HOPairNet (m = 4) | 0.7761 | 0.8443 | 0.6070 |
| HOFM (m = 2) | 0.7037 | 0.7840 | 0.6362 |
| HOFM (m = 3) | 0.7470 | 0.7858 | 0.6362 |
| HOFM (m = 4) | 0.7412 | 0.7840 | 0.6225 |
| PairSVM | 0.7290 | 0.9171 | 0.5972 |

We first compared our proposed HOPairNet with these existing models:

- **HOPairNet.** Our proposed HOPairNet defined in Eq. (14). We minimized the objective function (23) by using the CD algorithm. We introduced $\lambda$ when $m \geq 4$ because of the Lemma 1. For the DBLP and NIPS datasets, we used the symmetrization method described in Sect. 4.5.

- **HOFM.** The higher-order factorization machine [4, 8] defined in Eq. (5). It was optimized by using the CD algorithm.

- **PairSVM.** We used the SVM with a second-order pairwise kernel proposed by [1], [2]:
  
  $y = \sum_{i=1}^{N} \alpha_i \psi^2((a_i, b_i), (a, b))$.

For the DBLP and NIPS datasets, we set $k = 30$ for the HOPairNet and HOFM following [17]. For the ML100K dataset, we set $k = 10$ for the HOPairNet and HOFM. For the HOPairNet and HOFM we set $\beta$ (a regularization hyper-parameter) to $10^{-7}, 10^{-6}, \text{or } 10^{-5}$ on the basis of the accuracy for the validation set. We set the regularization hyper-parameter in the objective function of the PairSVM to $10^{-7}, 10^{-6}, \ldots, 10^{-2}$. We set $l(\cdot, \cdot)$ as the logistic loss for the HOPairNet, HOFM. We compared the accuracies of these models for the three test sets. Note that this is an experiment of feature-based link prediction task and hence we did not use indices of objects (namely, the adjacency-matrix/graph). Hence, we cannot compare these methods and other methods for index-based link prediction that require indices of objects, e.g., latent factor models [28] and the neural collaborative filtering [26], the graph neural networks [32].

The number of positive and negative pairs in all datasets were equalized by undersampling the negative pairs similarly in [1], [23]. For all datasets, we created a validation set by randomly sampling the test set after creating the training and test sets. The details for the three datasets are summarized in Table 1. Intuitively, proposed methods seem suitable for the ML100K training and test sets. The details for the three datasets are summarized in Table 1. Intuitively, proposed methods seem suitable for the ML100K datasets because of the Lemma 1. For the DBLP and NIPS datasets we used the symmetrization method described in Sect. 4.5.

- **PairSVM.** We used the SVM with a second-order pairwise kernel proposed by [1], [2]:
  
  $y = \sum_{i=1}^{N} \alpha_i \psi^2((a_i, b_i), (a, b))$.

For the DBLP and NIPS datasets, we set $k = 30$ for the HOPairNet and HOFM following [17]. For the ML100K dataset, we set $k = 10$ for the HOPairNet and HOFM. For the HOPairNet and HOFM we set $\beta$ (a regularization hyper-parameter) to $10^{-7}, 10^{-6}, \text{or } 10^{-5}$ on the basis of the accuracy for the validation set. We set the regularization hyper-parameter in the objective function of the PairSVM to $10^{-7}, 10^{-6}, \ldots, 10^{-2}$. We set $l(\cdot, \cdot)$ as the logistic loss for the HOPairNet, HOFM. We compared the accuracies of these models for the three test sets. Note that this is an experiment of feature-based link prediction task and hence we did not use indices of objects (namely, the adjacency-matrix/graph). Hence, we cannot compare these methods and other methods for index-based link prediction that require indices of objects, e.g., latent factor models [28] and the neural collaborative filtering [26], the graph neural networks [32].

5.2 Comparison with HOPairNets and Existing Models

We first compared our proposed HOPairNet with these existing models:

- **HOPairNet.** Our proposed HOPairNet defined in Eq. (14). We minimized the objective function (23) by using the CD algorithm. We introduced $\lambda$ when $m \geq 4$ because of the Lemma 1. For the DBLP and NIPS datasets, we used the symmetrization method described in Sect. 4.5.

- **HOFM.** The higher-order factorization machine [4, 8] defined in Eq. (5). It was optimized by using the CD algorithm.

- **PairSVM.** We used the SVM with a second-order pairwise kernel proposed by [1], [2]:
  
  $y = \sum_{i=1}^{N} \alpha_i \psi^2((a_i, b_i), (a, b))$.

For the DBLP and NIPS datasets, we set $k = 30$ for the HOPairNet and HOFM following [17]. For the ML100K dataset, we set $k = 10$ for the HOPairNet and HOFM. For the HOPairNet and HOFM we set $\beta$ (a regularization hyper-parameter) to $10^{-7}, 10^{-6}, \text{or } 10^{-5}$ on the basis of the accuracy for the validation set. We set the regularization hyper-parameter in the objective function of the PairSVM to $10^{-7}, 10^{-6}, \ldots, 10^{-2}$. We set $l(\cdot, \cdot)$ as the logistic loss for the HOPairNet, HOFM. We compared the accuracies of these models for the three test sets. Note that this is an experiment of feature-based link prediction task and hence we did not use indices of objects (namely, the adjacency-matrix/graph). Hence, we cannot compare these methods and other methods for index-based link prediction that require indices of objects, e.g., latent factor models [28] and the neural collaborative filtering [26], the graph neural networks [32].

Results are shown in Table 2. For the DBLP dataset, the third-order HOPairNet achieved the best accuracy. The accuracy of the HOPairNet was more robust than that of the HOFM with respect to the order of feature combinations. Higher-order feature combinations across objects can be effective as shown by the better performance of the third- and fourth-order HOPairNet compared to the second-order HOPairNet (which is a PairNet equivalent to the FBM) for all datasets. The better performance of the HOPairNet compared to the HOFM for the DBLP and NIPS datasets shows that using feature combinations only across objects is important for such problems as predicting identity.

While the PairSVM achieved the best performance for the NIPS dataset, its performance for the DBLP dataset was not good. The proportion of non-zero values in the DBLP and NIPS datasets was 0.1% and 7.5%. Hence, if the data are not very sparse and the number of training instances is not so large, the PairSVM is a good choice.

For the ML100K dataset, the HOFM outperformed the HOPairNet and PairSVM because the HOFM use feature combinations from the same object and the ML100K dataset is designed for recommender systems as described above. Because the ML100K dataset is designed for recommender systems, use of feature combinations from the same object can be effective (e.g., there may be movies that tend to receive high ratings).

5.3 Comparison of HOPairDNN with Existing DNN-Based Models

Next we compared our proposed HOPairDNN with a DNN-based FM/HOFM for the DBLP dataset.

- **HOPairDNN.** Our proposed HOPairDNN is described in Sect. 4.6. We minimized the logistic loss without regularization terms by using the Adam stochastic optimization method [36]. We set the learning rate for $\Theta$ to the default value (0.001), and for $U^{(t)}, V^{(t)}, t \in [2, m]$ to 0.001, 0.002, $\ldots$, or 0.009. We used the identity function as an activation function in the higher-order pairwise combination layer $\sigma(\cdot)$. We used the relu function as an activation function in $f_{\text{DNN}}$.

- **HONFM.** The HONFM is the higher-order extension of the NFM [9]. We also tuned the HONFM as for the HOPairDNN.

- **Concat.** A DNN with $(a; b)$ as input. We used Adam with the learning rate 0.001, 0.002, $\ldots$, or 0.009. We used relu activation function in all hidden layers.
The number of hidden layers was four in all models. We used the Dropout [37] for regularization in all models. We adjusted the number of hidden units to make the number of parameters almost the same as that for the simple DNN that has four hidden layers with 1,000 units and whose input vectors are the addition of $a$ and $b$: $a + b$. We ran the experiment five times using different initial values and compared the average values.

As shown in Table 3, the HOPairDNNS, especially the third-order one, achieved the best performance. Note that the differences in performance between the second-, third-, and fourth-order HOPairDNNS were smaller than those for the HOPairNets. Since DNN-based models are strongly non-linear, the effect of higher-order feature combinations may be smaller. Although the results of the HONFM, which are the DNN-extension of the HOFM, were better than those of HOFMs and PairNets, they were inferior to those of HOPairDNNS. Both HONFM and HOPairNets outperformed the Concat and thus explicitly using feature combinations are effective.

### 5.4 Comparison on Imbalanced Setting

Next, we compared our proposed methods and existing methods on the imbalanced DBLP dataset. Although we undersampled the negative pairs for training, validation, and test data in Sect. 5.2 and 5.3, we undersampled the negative pairs for only training data in this experiment. The number of pairs in validation and test data were 284,099 and 1,136,398, respectively. We used the same training data as in Sect. 5.2 and Sect. 5.3. We used the area under the receiver operating characteristic curve (ROC-AUC), precision, and recall as the evaluation metrics since the number of negative pairs was one hundred times more than that of positive pairs in imbalanced DBLP dataset. We tuned the hyper-parameters of the proposed and existing methods as in Sect. 5.2 and 5.3 with ROC-AUC as the evaluation metric.

As shown in Table 4, the HOPairDNNS, especially fourth-order one, achieved the best ROC-AUC, and our proposed HOPairNets outperformed HOFMs. Although the precisions of all models were low, that of the PairSVM was higher than those of other models and thus the ROC-AUC of the PairSVM was higher than those of the HOPairNets, HOFM, and HONFM’s. In our hyper-parameter tuning scenario, the highest recall of the PairSVM was 0.5247, it was lower than those of proposed models in Table 4. We note that the HOPairDNNS achieved the best precision and F-measure when we tuned the hyper-parameters of all models on the basis of the F-measure for the validation set; the highest F-measure and precision were 0.4024 and 0.9077, respectively. Therefore, our experimental results suggested that a machine learning user should use the HOPairDNN or PairSVM if the precision is more important, otherwise, normally use the HOPairDNN.

Although we also experimented other existing or baseline models [14, 38–40] and compared the efficiency of the proposed CD algorithm with that of previous one, we omit the results of them due to the space limitation.

### 6. Conclusion

We presented models using higher-order feature combinations only across the two objects being compared. Our proposed model, HOPairNet, can be regarded as a higher-order generalization of the factorized bilinear model or pairwise extension of the higher-order factorization machine. We also presented an algorithm for efficiently computing higher-order feature combinations only across two objects. Moreover, we proposed an efficient CD algorithm for proposed models. Furthermore, we proposed the HOPairDNN, which is a DNN-extension of the HOPairNet. We also presented the relationships among proposed methods, existing methods for feature-based link prediction and for index-based link prediction. Experimental results demonstrated the effectiveness of our proposed models.

### Acknowledgements

This work was partially supported by JSPS KAKENHI Grant Numbers JP15H05711 and JP18H03337 and by the Global Station for Big Data and CyberSecurity, a project of the Global Institution for Collaborative Research and Education at Hokkaido University. Kyohei Atarashi would like to give special thanks to Kazune Furudo for getting me interested in this problem and for helpful advice and discussion.

### Table 3 Comparison of accuracies of DNN-based models for DBLP dataset with those of existing models. For comparison, some results from Table 2 are shown.

| Model       | Accuracy | Model       | Accuracy |
|-------------|----------|-------------|----------|
| HOPairDNN ($m = 2$) | 0.8527   | HOPairNet ($m = 2$) | 0.7639   |
| HOPairDNN ($m = 3$) | 0.8547   | HOPairNet ($m = 3$) | 0.7827   |
| HOPairDNN ($m = 4$) | 0.8483   | HOPairNet ($m = 4$) | 0.7761   |
| HONFM ($m = 2$)   | 0.8144   | HOFM ($m = 2$)     | 0.7057   |
| HONFM ($m = 3$)   | 0.8142   | HOFM ($m = 3$)     | 0.7470   |
| HONFM ($m = 4$)   | 0.8124   | HOFM ($m = 4$)     | 0.7412   |
| Concat        | 0.7600   | PairSVM       | 0.7290   |

### Table 4 Comparison of ROC-AUCs, precisions, and recalls for imbalanced DBLP dataset.

| Model       | ROC-AUC | Precision | Recall  |
|-------------|---------|-----------|---------|
| HOPairDNN ($m = 2$) | 0.9064  | 0.0822    | 0.7629  |
| HOPairDNN ($m = 3$) | 0.9057  | 0.1392    | 0.7154  |
| HOPairDNN ($m = 4$) | 0.9124  | 0.1873    | 0.6993  |
| HOPairNet ($m = 2$) | 0.8350  | 0.0800    | 0.5659  |
| HOPairNet ($m = 3$) | 0.8393  | 0.0822    | 0.5925  |
| HOPairNet ($m = 4$) | 0.8384  | 0.0792    | 0.6075  |
| HONFM ($m = 2$)    | 0.8700  | 0.0297    | 0.7841  |
| HONFM ($m = 3$)    | 0.8676  | 0.0359    | 0.7487  |
| HONFM ($m = 4$)    | 0.8670  | 0.0364    | 0.7456  |
| Concat            | 0.8331  | 0.0923    | 0.5379  |
| PairSVM           | 0.8811  | 0.2047    | 0.5223  |
References

[1] S. Oyama and C.D. Manning, “Using feature conjunctions across examples for learning pairwise classifiers,” ECML, vol.3201, pp.322–333, 2004.

[2] A. Ben-Hur and W.S. Noble, “Kernel methods for predicting protein–protein interactions,” Bioinformatics, vol.21, no.suppl_1, pp.138–146, 2005.

[3] W. Wu, Z. Lu, and H. Li, “Learning bilinear model for matching queries and documents,” The Journal of Machine Learning Research, vol.14, no.1, pp.2519–2548, 2013.

[4] S. Rendle, “Factorization machines,” ICDM, pp.995–1000, 2010.

[5] A.K. Menon and C. Elkan, “Link prediction via matrix factorization,” ECML-PKDD, vol.6912, pp.437–452, 2011.

[6] N. Natarajan and I.S. Dhillon, “Inductive matrix completion for predicting gene–disease associations,” Bioinformatics, vol.30, no.12, pp.160–168, 2014.

[7] S. Rendle, “Factorization machines with libfm,” ACM Transactions on Intelligent Systems and Technology, vol.3, no.3, p.57, 2012.

[8] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, “Higher-order factorization machines,” NeurIPS, pp.3351–3359, 2016.

[9] X. He and T.-S. Chua, “Neural factorization machines for sparse predictive analytics,” SIGIR, pp.355–364, 2017.

[10] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, “Product-based neural networks for user response prediction,” ICDM, pp.1149–1154, 2016.

[11] W. Zhang, T. Du, and J. Wang, “Deep learning over multi-field categorical data,” ECIR, vol.9626, pp.45–57, 2016.

[12] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, “Deepfm: a factorization-machine based neural network for ctr prediction,” IJCAI, pp.1725–1731, 2017.

[13] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, “xdeepfm: Combining explicit and implicit feature interactions for recommender systems,” KDD, pp.1754–1763, 2018.

[14] R. Livni, S. Shalev-Shwartz, and O. Shamir, “On the computational efficiency of training neural networks,” NeurIPS, pp.855–863, 2014.

[15] V. Vapnik, Statistical learning theory, Wiley New York, 1998.

[16] J. Shawe-Taylor and N. Cristianini, “Kernel Methods for Pattern Analysis,” Cambridge University Press, Cambridge, UK, 2004.

[17] M. Blondel, M. Ishihata, A. Fujino, and N. Ueda, “Polynomial networks and factorization machines: new insights and efficient training algorithms,” ICML, pp.850–858, 2016.

[18] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning,” Book in preparation for MIT Press, pp.443–485, 2016.

[19] M. Fazel, Matrix rank minimization with applications, Ph.D. thesis, Stanford University, 2002.

[20] M. Jaggi, M. Suvorov, et al., “A simple algorithm for nuclear norm regularized problems,” ICML, pp.471–478, 2010.

[21] R. Tibshirani, “Regression shrinkage and selection via the lasso,” Journal of the Royal Statistical Society: Series B (Methodological), vol.58, no.1, pp.267–288, 1996.

[22] N. Parikh and S. Boyd, “Proximal algorithms,” Foundations and Trends® in Optimization, vol.1, no.3, pp.127–239, 2014.

[23] K. Atarashi, S. Oyama, M. Kurihara, and K. Furudo, “A deep neural network for pairwise classification: Enabling feature conjunctions and ensuring symmetry,” PKDD, vol.10234, pp.83–95, 2017.

[24] Y. Li, N. Wang, J. Liu, and X. Hou, “Factorized bilinear models for image recognition,” ICCV, pp.2098–2106, 2017.

[25] K.W. On, J.H. Kim, J. Kim, and J.W. Hu, “Hadamard product for low-rank bilinear pooling,” ICLR, 2017.

[26] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” WWW, pp.173–182, 2017.

[27] T.-Y. Lin, A. RoyChowdhury, and S. Maji, “Bilinear cnn models for fine-grained visual recognition,” ICCV, pp.1449–1457, 2015.

[28] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” Computer, vol.42, no.8, pp.30–37, 2009.

[29] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” KDD, pp.855–864, 2016.

[30] H. Zhao, L. Du, and W. Buntine, “Leveraging node attributes for incomplete relational data,” ICM, pp.4072–4081, 2017.

[31] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P.S. Yu, “A comprehensive survey on graph neural networks,” arXiv preprint arXiv:1901.00596, 2019.

[32] M. Zhang and Y. Chen, “Link prediction based on graph neural networks,” NeurIPS, pp.5165–5175, 2018.

[33] S. Roweis, “https://cs.nyu.edu/~roweis/data.html,” 2002.

[34] GroupLens, “http://grouplens.org/datasets/movielens/,” 1998.

[35] A. Novikov, M. Trofimov, and I. Oseledets, “Exponential machines. arxiv preprint,” ICLR Workshop., 2017.

[36] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” ICLR, 2014.

[37] S. Wager, S. Wang, and P.S. Liang, “Dropout training as adaptive regularization,” NeurIPS, pp.351–359, 2013.

[38] J. Bromley, J.W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, “Signature verification using a “siamese” time delay neural network,” NeurIPS, pp.737–744, 1993.

[39] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” CVPR, pp.539–546, 2005.

[40] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” CVPR, pp.1701–1708, 2014.

Kyohei Atarashi received a B.S. in engineering and an M.S. in information science and technology from Hokkaido University in 2016 and 2018, respectively. He is currently a doctoral course student at the Graduate School of Information Science and Technology, Hokkaido University.

Satoshi Oyama is an associate professor at Hokkaido University in Japan. He received his B.Eng., M.Eng., and Ph.D. degrees from Kyoto University in 1994, 1996, and 2002, respectively. Before he joined Hokkaido University, he was an assistant professor at Kyoto University from 2002 to 2009. He was also a visiting assistant professor at Stanford University from 2003 to 2004 and a visiting research professor at New York University in 2015. His research interests include machine learning, data mining, information retrieval, and crowdsourcing.

Masahito Kurihara received a B.S., a M.S., and a Ph.D. in engineering from Hokkaido University, in 1978, 1980, and 1986, respectively. He is currently a professor of computer science and information technology at Hokkaido University. His research interests include automated reasoning in computer science and artificial intelligence. He is a member of IPSJ, JSSST, and JSAI.