Learning from Multi-View Multi-Way Data via Structural Factorization Machines

Chun-Ta Lu
University of Illinois at Chicago
clu29@uic.edu

Lifang He*
Cornell University
lifanghescut@gmail.com

Hao Ding
Purdue University
haoding.tourist@gmail.com

Bokai Cao
University of Illinois at Chicago
caobokai@uic.edu

Philip S. Yu
University of Illinois at Chicago
Tsinghua University
psyu@cs.uic.edu

ABSTRACT

Real-world relations among entities can often be observed and determined by different perspectives/views. For example, the decision made by a user on whether to adopt an item relies on multiple aspects such as the contextual information of the decision, the item's attributes, the user's profile and the reviews given by other users. Different views may exhibit multi-way interactions among entities and provide complementary information. In this paper, we introduce a multi-tensor-based approach that can preserve the underlying structure of multi-view data in a generic predictive model. Specifically, we propose structural factorization machines (SFMs) that learn the common latent spaces shared by multi-view tensors and automatically adjust the importance of each view in the predictive model. Furthermore, the complexity of SFMs is linear in the number of parameters, which make SFMs suitable to large-scale problems. Extensive experiments on real-world datasets demonstrate that the proposed SFMs outperform several state-of-the-art methods in terms of prediction accuracy and computational cost.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Supervised learning**; **Factorization methods**;

KEYWORDS

Tensor Factorization; Multi-Way Interaction; Multi-View Learning

ACM Reference Format:
Chun-Ta Lu, Lifang He, Hao Ding, Bokai Cao, and Philip S. Yu. 2018. Learning from Multi-View Multi-Way Data via Structural Factorization Machines. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/https://doi.org/10.1145/3178876.3186071

*Corresponding author.

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8.
https://doi.org/https://doi.org/10.1145/3178876.3186071

1 INTRODUCTION

With the ability to access massive amounts of heterogeneous data from multiple sources, multi-view data have become prevalent in many real-world applications. For instance, in recommender systems, online review sites (like Amazon and Yelp) have access to contextual information of shopping histories of users, the reviews written by the users, the categorizations of the items, as well as the friends of the users. Each view may exhibit pairwise interactions (e.g., the friendships between users) or even higher-order interactions (e.g., a customer write a review for a product) among entities (such as customers, products, and reviews), and can be represented in a multi-way data structure, i.e., tensor. Since different views usually provide complementary information [4, 6, 28], how to effectively incorporate information from multiple structural views is critical to good prediction performance for various machine learning tasks.

Typically, a predictive model is defined as a function of predictor variables (e.g., the customer id, the product id, and the categories of the product) to some target (e.g., the rating). The most common approach in predictive modeling for multi-view multi-way data is to describe samples with feature vectors that are flattened and concatenated from structural views, and apply a classical vector-based method, such as linear regression (LR) and support vector machines (SVMs), to learn the target function from observed samples. Recent works have shown that linear models fail for tasks with very sparse data [34]. A variety of methods have been proposed to address the data sparsity issue by factorizing the monomials (or feature interactions) with kernels, such as the ANOVA kernels used in FMs [2, 34] and polynominal kernels used in polynominal networks [3, 27]. However, the disadvantages of this approach are that (1) the important structural information of each view will be discarded which may lead to the degraded prediction performance and (2) the feature vectors can grow very large which can make learning and prediction very slow or even infeasible, especially if each view involves relations of high cardinality. For example, including the relation "friends of a user" in the feature vector (represented by their IDs) can result in a very long feature vector. Further, it will repeatedly appear in many samples that involve the given user.

Matrix/tensor factorization models have been a topic of interest in the areas of multi-way data analysis, e.g., community detection [16], collaborative filtering [23, 36], knowledge graph completion [43], and neuroimage analysis [15]. Assuming multi-view data have the same underlying low-rank structure (at least in one

mode), coupled data analysis such as collective matrix factorization (CMF) [38] and coupled matrix and tensor factorization (CMTF) [1] that jointly factorize multiple matrices (or tensors) has been applied to applications such as clustering and missing data recovery. However, they are only applicable to categorical variables. Moreover, since existing coupled factorization models are unsupervised, the importance of each structural view in modeling the target value cannot be automatically learned. Furthermore, when applying these models to data with rich meta information (e.g., friendships) but extremely sparse target values (e.g., ratings), it is very likely the learning process will be dominated by the meta information without manual tuning some hyperparameters, e.g., the weights of the fitting error of each matrix/tensor in the objective function [38], the weights of different types of latent factors in the predictive models [24], or the regularization hyperparamters of latent factor alignment [29].

In this paper, we propose a general and flexible framework for learning the predictive structure from the complex relationships within the multi-view multi-way data. Each view of an instance in this framework is represented by a tensor that describes the multi-way interactions of subsets of entities, and different views have some entities in common. Constructing the tensors for each instance may not be realistic for real-world applications in terms of space and computational complexity, and the model parameters can have exponential growth and tend to be overfitting. In order to preserve the structural information of multi-view data without physically constructing the tensors, we introduce structural factorization machines (SFMs) that can learn the consistent representations in the latent feature spaces shared in the multi-view tensors while automatically adjust the contribution of each view in the predictive model. Furthermore, we provide an efficient method to avoid redundant computing on repeating patterns stemming from the relational structure of the data, such that SFMs can make the same predictions but with largely speed up computation.

The contributions of this paper are summarized as follows:

- We introduce a novel multi-tensor framework for mining data from heterogeneous domains, which can explore the high order correlations underlying multi-view multi-way data in a generic predictive model.
- We develop structural factorization machines (SFMs) tailored for learning the common latent spaces shared in multi-view tensors and automatically adjusting the importance of each view in the predictive model. The complexity of SFMs is linear in the number of features, which makes SFMs suitable to large-scale problems.
- Extensive experiments on eight real-world datasets are performed along with comparisons to existing state-of-the-art factorization models to demonstrate its advantages.

The rest of this paper is organized as follows. In Section 2, we briefly review related work on factorization models and multi-view learning. We introduce the preliminary concepts and problem definition in Section 3. We then propose the framework for learning multi-view multi-way data, and develop the structural factorization machines (SFMs), and provide an efficient computing method in Section 4. The experimental results and parameter analysis are reported in Section 5. Section 6 concludes this paper.

2 RELATED WORK

Feature Interactions. Rendle pioneered the concept of feature interactions in Factorization Machines (FM) [34]. Juan et al. presented Field-aware Factorization Machines (FFM) [20] to allow each feature to interact differently with another feature depending on its field. Novikov et al. proposed Exponential Machines (ExM) [32] where the weight tensor is represented in a factorized format called Tensor Train. Zhang et al. used FM to initialize the embedding layer in a deep model [44]. Qu et al. added a product layer on the top of the embedding layer to increase the model capacity [33]. Other extensions of FM to deep architectures include Neural Factorization Machines (NFM) [17] and Attentional Factorization Machines (AFM) [40]. In order to effectively model feature interactions, a variety of models has been developed in the industry as well. Microsoft studied feature interactions in deep models, including Deep Semantic Similarity Model (DSSM) [19], Deep Crossing [37] and Deep Embedding Forest [47]. They use features as raw as possible without manually crafted combinatorial features, and let deep neural networks take care of the rest. Alibaba proposed a Deep Interest Network (DIN) [46] to learn user embeddings as a function of ad embeddings. Google used deep neural networks to learn from heterogeneous signals for YouTube recommendations [9]. In addition, Wide & Deep Models [7] were developed for app recommender systems in Google Play where the wide component includes cross features that are good at memorization and the deep component includes embedding layers for generalization. Guo et al. proposed to use FM as the wide component in Wide & Deep with shared embeddings in the deep component [11]. Wang et al. developed the Deep & Cross Network (DCN) to learn explicit cross features of bounded degree [39].

Multi-View Learning. Multi-view learning (MVL) is concerned with predicting unknown values by taking multiple views into account. The traditional MVL refers to using relational features to construct a set of disjoint views, and these uncorrelated views are then used to model a target function to approximate the target concept to be learned [12]. There are currently a plethora of studies available for MVL. Interested readers are referred to [41] for a comprehensive survey of these techniques and applications. The most related works to ours are [5, 6, 25] that introduced and explored the tensor product operator to integrate different views together in a tensor. Lu et al. further studied the multi-view feature interactions in the context of multi-task learning [28]. However, this approach will introduce unexpected noise from the irrelevant feature interactions that can even be exaggerated after combinations, thereby degrading performance as demonstrated in the experiments. Different from conventional MVL approaches, the proposed algorithm can learn the common latent spaces shared in multi-view tensors and automatically adjusting the importance of each view in the predictive model.

3 PRELIMINARIES

In this section, we begin with a brief introduction to some related concepts and notation in tensor algebra, and then proceed to formulate the problem we are concerned with multi-view learning.

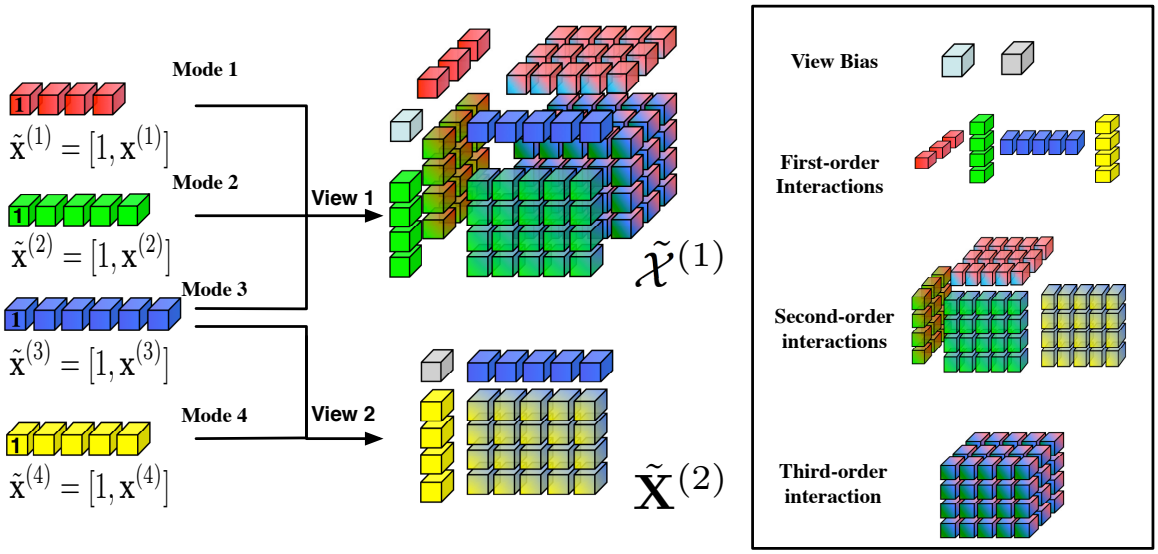


Figure 1: Example of multiple structural views, where $\tilde{\mathcal{X}}^{(1)} = \tilde{\mathbf{x}}^{(1)} \circ \tilde{\mathbf{x}}^{(2)} \circ \tilde{\mathbf{x}}^{(3)}$ and $\tilde{\mathbf{X}}^{(2)} = \tilde{\mathbf{x}}^{(3)} \circ \tilde{\mathbf{x}}^{(4)}$.

3.1 Tensor Basics and Notation

Tensor is a mathematical representation of a multi-way array. The order of a tensor is the number of modes (or ways). A zero-order tensor is a scalar, a first-order tensor is a vector, a second-order tensor is a matrix and a tensor of order three or higher is called a higher-order tensor. An element of a vector $\mathbf{x}$, a matrix $\mathbf{X}$, or a tensor $\mathcal{X}$ is denoted by $x_i, x_{i,j}, x_{i,j,k}$, etc., depending on the number of modes. All vectors are column vectors unless otherwise specified. For an arbitrary matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$, its i-th row and j-th column vector are denoted by $\mathbf{x}^i$ and $\mathbf{x}_j$, respectively. Given two matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{I \times J}$, $\mathbf{X} * \mathbf{Y}$ denotes the element-wise (Hadamard) product between $\mathbf{X}$ and $\mathbf{Y}$, defined as the matrix in $\mathbb{R}^{I \times J}$. An overview of the basic symbols used in this paper can be found in Table 1.

Definition 3.1 (Inner product). The inner product of two same-sized tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ is defined as the sum of the products of their entries:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_M=1}^{I_M} x_{i_1, i_2, \ldots, i_M} y_{i_1, i_2, \ldots, i_M}. \quad (1)$$

Definition 3.2 (Outer product). The outer product of two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{I'_1 \times I'_2 \times \cdots \times I'_M}$ is a $(N+M)$th-order tensor denoted by $\mathcal{X} \circ \mathcal{Y}$, and the elements are defined by

$$(\mathcal{X} \circ \mathcal{Y})_{i_1, i_2, \ldots, i_N, i'_1, i'_2, \ldots, i'_M} = x_{i_1, i_2, \cdots, i_N} y_{i'_1, i'_2, \cdots, i'_M} \quad (2)$$

for all values of the indices.

Notice that for rank-one tensors $\mathcal{X} = \mathbf{x}^{(1)} \circ \mathbf{x}^{(2)} \circ \cdots \circ \mathbf{x}^{(M)}$ and $\mathcal{Y} = \mathbf{y}^{(1)} \circ \mathbf{y}^{(2)} \circ \cdots \circ \mathbf{y}^{(M)}$, it holds that

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \left\langle \mathbf{x}^{(1)}, \mathbf{y}^{(1)} \right\rangle \left\langle \mathbf{x}^{(2)}, \mathbf{y}^{(2)} \right\rangle \cdots \left\langle \mathbf{x}^{(M)}, \mathbf{y}^{(M)} \right\rangle. \quad (3)$$

Definition 3.3 (CP factorization [22]). Given a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ and an integer R, the CP factorization is defined by factor matrices $\mathbf{X}^{(m)} \in \mathbb{R}^{I_m \times R}$ for $m \in [1:M]$, respectively, such that

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{x}_r^{(1)} \circ \mathbf{x}_r^{(2)} \circ \cdots \circ \mathbf{x}_r^{(M)} = [\![\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(M)}]\!], \quad (4)$$

where $\mathbf{x}_r^{(m)} \in \mathbb{R}^{I_m}$ is the r-th column of the factor matrix $\mathbf{X}^{(m)}$, and $[\![\cdot]\!]$ is used for shorthand notation of the sum of rank-one tensors.

Table 1: List of basic symbols.

Symbol	Definition and description
x	each lowercase letter represents a scalar
$\mathbf{x}$	each boldface lowercase letter represents a vector
$\mathbf{X}$	each boldface uppercase letter represents a matrix
$\mathcal{X}$	each calligraphic letter represents a tensor
$\mathfrak{X}$	each gothic letter represent a general set or space
$[1:N]$	a set of integers in the range of 1 to N inclusively.
$\langle \cdot, \cdot \rangle$	denotes inner product
$\circ$	denotes tensor product (outer product)
$*$	denotes Hadamard (element-wise) product

3.2 Problem Formulation

Our problem is different from conventional multi-view learning approaches where multiple views of data are assumed independent and disjoint, and each view is described by a vector. We formulate the multi-view learning problem using coupled analysis of multi-view features in the form of multiple tensors.

Suppose that the problem includes V views where each view consists of a collection of subsets of entities (such as person, company, location, product) and different views have some entities in common. We denote a view as a tuple $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(M)})$, $M \geq 2$, where $\mathbf{x}^{(m)} \in \mathbb{R}^{I_m}$ is a feature vector associated with the entity m. Inspired by [6], we construct tensor representation for each view

over its entities by

$$\tilde{\mathcal{X}} = \tilde{\mathbf{x}}^{(1)} \circ \tilde{\mathbf{x}}^{(2)} \circ \cdots \circ \tilde{\mathbf{x}}^{(M)} \in \mathbb{R}^{(1+I_1)\times\cdots\times(1+I_M)},$$

where $\tilde{\mathbf{x}}^{(m)} = [1; \mathbf{x}^{(m)}] \in \mathbb{R}^{1+I_m}$ and $\circ$ is the outer product operator. In this manner, the full-order interactions [1] between entities are embedded within the tensor structure, which not only provides a unified and compact representation for each view, but also facilitate efficient design methods. Fig. 1 shows an example of two structural views, where the first view consists of the full-order interactions among the first three modes (e.g., review text, item ID, and user ID), and the second view consists of the full-order interactions among the last two modes (e.g., user ID and friend IDs).

After generating the tensor representation for each view, we define the multi-view learning problem as follows. Given a training set $\mathfrak{D} = \left\{\left(\left\{\tilde{\mathcal{X}}_n^{(1)}, \tilde{\mathcal{X}}_n^{(2)}, \cdots, \tilde{\mathcal{X}}_n^{(V)}\right\},\ y_n\right) \mid n \in [1:N]\right\}$, where $\tilde{\mathcal{X}}_n^{(v)} \in \mathbb{R}^{(1+I_1)\times\cdots\times(1+I_{M_v})}$ is the tensor representation in the v-th view for the n-th instance, y_n is the response of the n-th instance, M_v is the number of the constitutive modes in the v-th view, and N is the number of labeled instances. We assume different views have common entities, thus the resulting tensors will share common modes, e.g., the third mode in Fig 1. As we are concerned with predicting unknown values of multiple coupled tensors, our goal is to leverage the relational information from all the views to help predict the unlabeled instances, as well as to use the complementary information among different views to improve the performance. Specifically, we are interested in finding a predictive function $f : \mathfrak{X}^{(1)} \times \mathfrak{X}^{(2)} \cdots \times \mathfrak{X}^{(V)} \to \mathfrak{Y}$ that minimizes the expected loss, where $\mathfrak{X}^{(v)}, v \in [1:V]$ is the input space in the v-th view and $\mathfrak{Y}$ is the output space.

4 METHODOLOGY

In this section, we first discuss how to design the predictive models for learning from multiple coupled tensors. We then derive structural factorization machines (SFMs) that can learn the common latent spaces shared in multi-view coupled tensors and automatically adjust the importance of each view in the predictive model.

4.1 Predictive Models

Without loss of generality, we take two views as an example to introduce our basic design of the predictive models. Specifically, we consider coupled analysis of a third-order tensor and a matrix with one mode in common, as shown in Fig. 1. Given an input instance $\left(\left\{\tilde{\mathcal{X}}^{(1)}, \tilde{\mathbf{X}}^{(2)}\right\},\ y\right)$, where $\tilde{\mathcal{X}}^{(1)} = \tilde{\mathbf{x}}^{(1)} \circ \tilde{\mathbf{x}}^{(2)} \circ \tilde{\mathbf{x}}^{(3)} \in \mathbb{R}^{(1+I)\times(1+J)\times(1+K)}$ and $\tilde{\mathbf{X}}^{(2)} = \tilde{\mathbf{x}}^{(3)} \circ \tilde{\mathbf{x}}^{(4)} \in \mathbb{R}^{(1+K)\times(1+L)}$. An intuitive solution is to build the following multiple linear model:

$$f\left(\left\{\tilde{\mathcal{X}}^{(1)}, \tilde{\mathbf{X}}^{(2)}\right\}\right) = \left\langle \tilde{\mathcal{W}}^{(1)}, \tilde{\mathcal{X}}^{(1)} \right\rangle + \left\langle \tilde{\mathbf{W}}^{(2)}, \tilde{\mathbf{X}}^{(2)} \right\rangle \tag{5}$$

where $\tilde{\mathcal{W}}^{(1)} \in \mathbb{R}^{(1+I)\times(1+J)\times(1+K)}$ and $\tilde{\mathbf{W}}^{(2)} \in \mathbb{R}^{(1+K)\times(1+L)}$ are the weights for each view to be learned.

However, in this case it does not take into account the relations and differences between two views. In order to incorporate the relations between two views and also discriminate the importance of each view, we introduce an indicator vector $\mathbf{e}_v \in \mathbb{R}^V$ for each view v as

$$\mathbf{e}_v = [\underbrace{0, \cdots, 0}_{\text{v-1}}, 1, 0, \cdots, 0]^{\mathrm{T}},$$

and transform the predictive model in Eq. (5) into

$$f\left(\left\{\tilde{\mathcal{X}}^{(1)}, \tilde{\mathbf{X}}^{(2)}\right\}\right) = \left\langle \hat{\mathcal{W}}^{(1)}, \tilde{\mathcal{X}}^{(1)} \circ \mathbf{e}_1 \right\rangle + \left\langle \hat{\mathcal{W}}^{(2)}, \tilde{\mathbf{X}}^{(2)} \circ \mathbf{e}_2 \right\rangle, \tag{6}$$

where $\hat{\mathcal{W}}^{(1)} \in \mathbb{R}^{(1+I)\times(1+J)\times(1+K)\times 2}$ and $\hat{\mathcal{W}}^{(2)} \in \mathbb{R}^{(1+K)\times(1+L)\times 2}$.

Directly learning the weight tensors $\hat{\mathcal{W}}$s leads to two drawbacks. First, the weight parameters are learned independently for different modes and different views. When the feature interactions rarely (or even never) appear during training, it is unlikely to learn the associated parameters appropriately. Second, the number of parameters in Eq. (6) is exponential to the number of features, which can make the model prone to overfitting and ineffective on sparse data. Here, we assume that each weight tensor has a low-rank approximation, and $\hat{\mathcal{W}}^{(1)}$ and $\hat{\mathcal{W}}^{(2)}$ can be decomposed by CP factorization as

$$\begin{aligned}\hat{\mathcal{W}}^{(1)} &= [\![\hat{\boldsymbol{\Theta}}^{(1,1)}, \hat{\boldsymbol{\Theta}}^{(1,2)}, \hat{\boldsymbol{\Theta}}^{(1,3)}, \boldsymbol{\Phi}]\!] \\ &= [\![[\mathbf{b}^{(1,1)}; \boldsymbol{\Theta}^{(1)}], [\mathbf{b}^{(1,2)}; \boldsymbol{\Theta}^{(2)}], [\mathbf{b}^{(1,3)}; \boldsymbol{\Theta}^{(3)}], \boldsymbol{\Phi}]\!],\end{aligned}$$

and

$$\hat{\mathcal{W}}^{(2)} = [\![\hat{\boldsymbol{\Theta}}^{(2,3)}, \hat{\boldsymbol{\Theta}}^{(2,4)}, \boldsymbol{\Phi}]\!] = [\![[\mathbf{b}^{(2,3)}; \boldsymbol{\Theta}^{(3)}], [\mathbf{b}^{(2,4)}; \boldsymbol{\Theta}^{(4)}], \boldsymbol{\Phi}]\!],$$

where $\boldsymbol{\Theta}^{(m)} \in \mathbb{R}^{I_m \times R}$ is the factor matrix for the features in the m-th mode. It is worth noting that $\boldsymbol{\Theta}^{(3)}$ is shared in the two views. $\boldsymbol{\Phi} \in \mathbb{R}^{2\times R}$ is the factor matrix for the view indicator, and $\mathbf{b}^{(v,m)} \in \mathbb{R}^{1\times R}$, which is always associated with the constant one in $\tilde{\mathbf{x}}^{(m)} = [1; \mathbf{x}^{(m)}]$, represents the bias factors of the m-th mode in the v-th view. Through $\mathbf{b}^{(v,m)}$, the lower-order interactions (the interactions excluding the features from the m-th mode) in the v-th view are explored in the predictive function.

Then we can transform Eq. (6) into

$$\begin{aligned}&\left\langle \hat{\mathcal{W}}^{(1)}, \tilde{\mathcal{X}}^{(1)} \circ \mathbf{e}_1 \right\rangle + \left\langle \hat{\mathcal{W}}^{(2)}, \tilde{\mathbf{X}}^{(2)} \circ \mathbf{e}_2 \right\rangle \\ =&\sum_{r=1}^{R} \left\langle \hat{\boldsymbol{\theta}}_r^{(1,1)} \circ \hat{\boldsymbol{\theta}}_r^{(1,2)} \circ \hat{\boldsymbol{\theta}}_r^{(1,3)} \circ \boldsymbol{\phi}_r\ ,\ \tilde{\mathbf{x}}^{(1)} \circ \tilde{\mathbf{x}}^{(2)} \circ \tilde{\mathbf{x}}^{(3)} \circ \mathbf{e}_1 \right\rangle \\ &+ \sum_{r=1}^{R} \left\langle \hat{\boldsymbol{\theta}}_r^{(2,3)} \circ \hat{\boldsymbol{\theta}}_r^{(2,4)} \circ \boldsymbol{\phi}_r\ ,\ \tilde{\mathbf{x}}^{(3)} \circ \tilde{\mathbf{x}}^{(4)} \circ \mathbf{e}_2 \right\rangle \\ =&\boldsymbol{\phi}^1 \left(\prod_{m=1}^{3} * \left(\tilde{\mathbf{x}}^{(m)\mathrm{T}} \hat{\boldsymbol{\Theta}}^{(1,m)} \right) \right)^{\mathrm{T}} + \boldsymbol{\phi}^2 \left(\prod_{m=3}^{4} * \left(\tilde{\mathbf{x}}^{(m)\mathrm{T}} \hat{\boldsymbol{\Theta}}^{(2,m)} \right) \right)^{\mathrm{T}} \\ =&\boldsymbol{\phi}^1 \left(\prod_{m=1}^{3} * \left(\mathbf{x}^{(m)\mathrm{T}} \boldsymbol{\Theta}^{(m)} + \mathbf{b}^{(1,m)} \right) \right)^{\mathrm{T}} + \boldsymbol{\phi}^2 \left(\prod_{m=3}^{4} * \left(\mathbf{x}^{(m)\mathrm{T}} \boldsymbol{\Theta}^{(m)} + \mathbf{b}^{(2,m)} \right) \right)^{\mathrm{T}}\end{aligned} \tag{7}$$

where $*$ is the Hadamard (elementwise) product and $\boldsymbol{\phi}^v \in \mathbb{R}^{1\times R}$ is the v-th row of the factor matrix $\boldsymbol{\Phi}$.

For convenience, we let $\mathbf{h}^{(m)} = \boldsymbol{\Theta}^{(m)\mathrm{T}} \mathbf{x}^{(m)}$, $S_M(v)$ denote the set of modes in the v-th views, $\boldsymbol{\pi}^{(v)} = \prod_{m \in S_M(v)} * \left(\mathbf{h}^{(m)} + \mathbf{b}^{(v,m)^{\mathrm{T}}} \right)$, and $\boldsymbol{\pi}^{(v,-m)} = \prod_{m' \in S_M(v), m' \neq m} * \left(\mathbf{h}^{(m')} + \mathbf{b}^{(v,m')^{\mathrm{T}}} \right)$. The predictive

[1] Full-order interactions range from the first-order interactions (i.e., contributions of single entity features) to the highest-order interactions (i.e., contributions of the outer product of features from all entities).

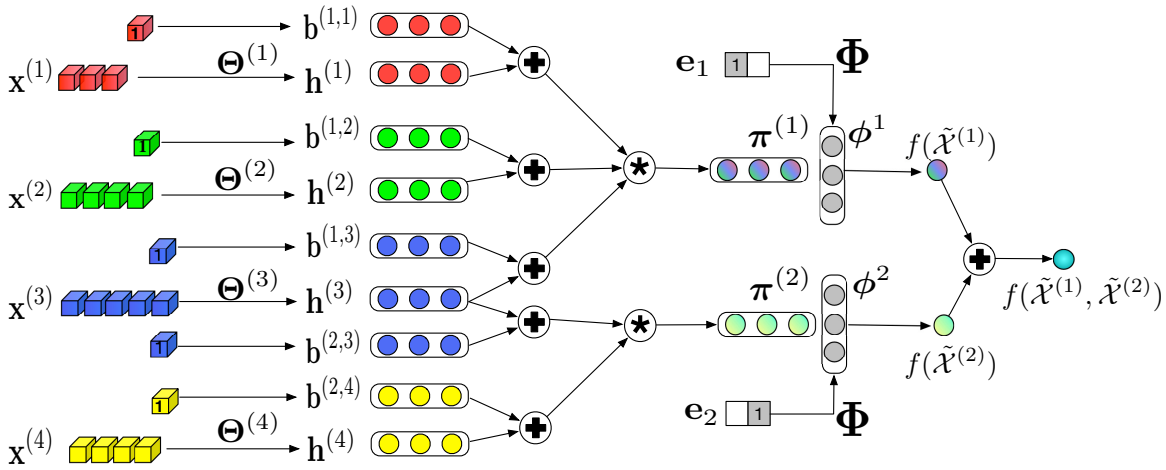


Figure 2: Example of the computational graph in a structural factorization machine, given the input $\tilde{\mathcal{X}}^{(1)}$ and $\tilde{\mathcal{X}}^{(2)}$. By jointly factorizing weight tensors, the $\mathbf{h}^{(m)}$ can be regarded as the latent representation of the feature $\mathbf{x}^{(m)}$ in m-th mode, and $\boldsymbol{\pi}^{(v)}$ can be regarded as the joint representation of all the modes in the v-th view, which can be easily computed through the Hadamard product. The contribution of $\boldsymbol{\pi}^{(v)}$ to the final prediction score is automatically adjusted by the weight vector $\boldsymbol{\phi}^v$.

model for the general cases is given as follows

$$\begin{aligned} f(\{\tilde{\mathcal{X}}^{(v)}\}) &= \sum_{v=1}^{V} \left\langle \hat{\mathcal{W}}^{(v)}, \tilde{\mathcal{X}}^{(v)} \circ \mathbf{e}_v \right\rangle \\ &= \sum_{v=1}^{V} \boldsymbol{\phi}^v \prod_{m \in S_M(v)} * \left(\mathbf{x}^{(m)^{\mathrm{T}}} \mathbf{\Theta}^{(m)} + \mathbf{b}^{(v,m)} \right)^{\mathrm{T}} \\ &= \sum_{v=1}^{V} \boldsymbol{\phi}^v \prod_{m \in S_M(v)} * \left(\mathbf{h}^{(m)} + \mathbf{b}^{(v,m)^{\mathrm{T}}} \right) \end{aligned} \tag{8}$$

A graphical illustration of the proposed model is shown in Fig. 2. We name this model as structural factorization machines (SFMs). Clearly, the parameters are jointly factorized, which benefits parameter estimation under sparsity since dependencies exist when the interactions share the same features. Therefore, the model parameters can be effectively learned without direct observations of such interactions especially in highly sparse data. More importantly, after factorizing the weight tensor $\hat{\mathcal{W}}$s, there is no need to construct the input tensor physically. Furthermore, the model complexity is linear in the number of original features. In particular, the model complexity is $O(R(V + I + \sum_v M_v))$, where M_v is the number of modes in the v-th view.

4.2 Learning Structural Factorization Machines

Following the traditional supervised learning framework, we propose to learn the model parameters by minimizing the following regularized empirical risk:

$$\mathcal{R} = \frac{1}{N} \sum_{n=1}^{N} \ell \left(f(\{\mathcal{X}_n^{(v)}\}), y_n \right) + \lambda \Omega(\mathbf{\Phi}, \{\mathbf{\Theta}^{(m)}\}, \{\mathbf{b}^{(v,m)}\}) \tag{9}$$

where ℓ is a prescribed loss function, Ω is the regularizer encoding the prior knowledge of $\{\mathbf{\Theta}^{(m)}\}$ and $\mathbf{\Phi}$, and $\lambda \geq 0$ is the regularization parameter that controls the trade-off between the empirical loss and the prior knowledge.

The partial derivative of $\mathcal{R}$ w.r.t. $\mathbf{\Theta}^{(m)}$ is given by

$$\frac{\partial \mathcal{R}}{\partial \mathbf{\Theta}^{(m)}} = \frac{\partial \mathcal{L}}{\partial f} \frac{\partial f}{\partial \mathbf{\Theta}^{(m)}} + \lambda \frac{\partial \Omega_\lambda(\mathbf{\Theta}^{(m)})}{\partial \mathbf{\Theta}^{(m)}} \tag{10}$$

where $\frac{\partial \mathcal{L}}{\partial f} = \frac{1}{N} \left[\frac{\partial \ell_1}{\partial f}, \cdots, \frac{\partial \ell_N}{\partial f} \right]^{\mathrm{T}} \in \mathbb{R}^N$.

For convenience, we let $S_V(m)$ denote the set of views that contains the m-th mode, $\mathbf{X}^{(m)} = [\mathbf{x}_1^{(m)}, \cdots, \mathbf{x}_N^{(m)}]$, $\mathbf{\Pi}^{(v)} = [\boldsymbol{\pi}_1^{(v)}, \cdots, \boldsymbol{\pi}_N^{(v)}]^{\mathrm{T}}$ and $\mathbf{\Pi}^{(v,-m)} = [\boldsymbol{\pi}_1^{(v,-m)}, \cdots, \boldsymbol{\pi}_N^{(v,-m)}]^{\mathrm{T}}$. We then have that

$$\frac{\partial \mathcal{L}}{\partial f} \frac{\partial f}{\partial \mathbf{\Theta}^{(m)}} = \mathbf{X}^{(m)} \left(\sum_{v \in S_V(m)} \left(\left(\frac{\partial \mathcal{L}}{\partial f} \boldsymbol{\phi}^v \right) * \mathbf{\Pi}^{(v,-m)} \right) \right) \tag{11}$$

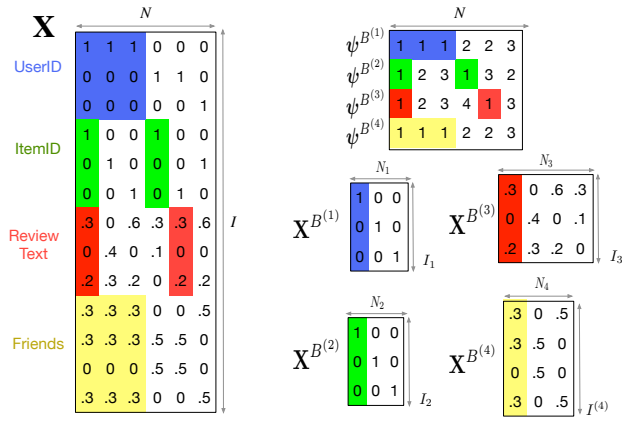
Similarly, the partial derivative of $\mathcal{R}$ w.r.t. $\mathbf{b}^{(v,m)}$ is given by

$$\begin{aligned} \frac{\partial \mathcal{R}}{\partial \mathbf{b}^{(v,m)}} &= \frac{\partial \mathcal{L}}{\partial f} \frac{\partial f}{\partial \mathbf{b}^{(v,m)}} + \lambda \frac{\partial \Omega_\lambda(\mathbf{b}^{(v,m)})}{\partial \mathbf{b}^{(v,m)}} \\ &= \mathbf{1}^{\mathrm{T}} \left(\left(\frac{\partial \mathcal{L}}{\partial f} \boldsymbol{\phi}^v \right) * \mathbf{\Pi}^{(v,-m)} \right) + \lambda \frac{\partial \Omega_\lambda(\mathbf{b}^{(v,m)})}{\partial \mathbf{b}^{(v,m)}} \end{aligned} \tag{12}$$

The partial derivative of $\mathcal{R}$ w.r.t. $\mathbf{\Phi}$ is given by

$$\frac{\partial \mathcal{R}}{\partial \mathbf{\Phi}} = \left[\left(\frac{\partial \mathcal{L}}{\partial f} \right)^{\mathrm{T}} \mathbf{\Pi}^{(1)} ; \cdots ; \left(\frac{\partial \mathcal{L}}{\partial f} \right)^{\mathrm{T}} \mathbf{\Pi}^{(V)} \right] + \lambda \frac{\partial \Omega_\lambda(\mathbf{\Phi})}{\partial \mathbf{\Phi}} \tag{13}$$

Finally, the gradient of $\mathcal{R}$ can be formed by vectorizing the partial derivatives with respect to each factor matrix and concatenating



(a) Plain Format of Feature Matrix (b) Relational Structure Representation

Figure 3: (a) Feature vectors of the same entity repeatedly appear in the plain formatted feature matrix X. (b) Repeating patterns in X can be formalized by the relational structure B of each mode. For example, the forth column of the feature matrix X can be represented as $\mathbf{x}_4 = [\mathbf{x}^{(1)}_{\psi(4)}; \mathbf{x}^{(2)}_{\psi(4)}; \mathbf{x}^{(3)}_{\psi(4)}; \mathbf{x}^{(4)}_{\psi(4)}] = [\mathbf{x}^{B^{(1)}}_2; \mathbf{x}^{B^{(2)}}_1; \mathbf{x}^{B^{(3)}}_4; \mathbf{x}^{B^{(4)}}_2]$.

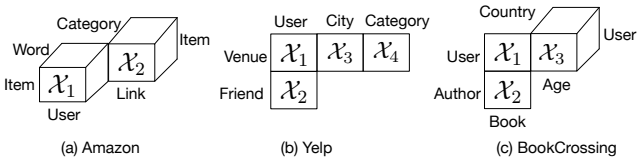
them all, i.e.,

$$\nabla\mathcal{R} = \begin{bmatrix} \mathrm{vec}(\frac{\partial\mathcal{R}}{\partial\mathbf{\Theta}^{(1)}}) \\ \vdots \\ \mathrm{vec}(\frac{\partial\mathcal{R}}{\partial\mathbf{\Theta}^{(M)}}) \\ \mathrm{vec}(\frac{\partial\mathcal{R}}{\partial\mathbf{b}^{(1,1)}}) \\ \vdots \\ \mathrm{vec}(\frac{\partial\mathcal{R}}{\partial\mathbf{b}^{(V,M)}}) \\ \mathrm{vec}(\frac{\partial\mathcal{R}}{\partial\mathbf{\Phi}}) \end{bmatrix} \tag{14}$$

Once we have the function, $\mathcal{R}$ and gradient, $\nabla\mathcal{R}$, we can use any gradient-based optimization algorithm to compute the factor matrices. For the results presented in this paper, we use the Adaptive Moment Estimation (Adam) optimization algorithm [21] for parameter updates. Adam is an adaptive version of gradient descent that controls individual adaptive learning rates for different parameters from estimates of first and second moments of the gradient. It combines the best properties of the AdaGrad [10], which works well with sparse gradients, and RMSProp [18], which works well in on-line and non-stationary settings. Readers can refer to [21] for details of the Adam optimization algorithm.

4.3 Efficient Computing with Relational Structures

In relational domains, we can often observe that feature vectors of the same entity repeatedly appear in the plain formatted feature matrix $\mathbf{X}$, where $\mathbf{X} = [\mathbf{X}^{(1)}; \cdots; \mathbf{X}^{(M)}] \in \mathbb{R}^{I \times N}$ and $\mathbf{X}^{(m)} \in \mathbb{R}^{I_m \times N}$ is the feature matrix in the m-th mode. Consider Fig. 3(a) as an example, where the parts highlighted in yellow in the forth mode



(a) Amazon (b) Yelp (c) BookCrossing

Figure 4: Schema of the structural views in each dataset.

(which represents the friends of the user) are repeatedly appear in the first three columns. Clearly, these repeating patterns stem from the relational structure of the same entity.

In the following, we show how the proposed SFM method can make use of relational structure of each mode, such that the learning and prediction can be scaled to predictor variables generated from relational data involving relations of high cardinality. We adopt the idea from [35] to avoid redundant computing on repeating patterns over a set of feature vectors.

Let $\mathcal{B} = \{(\mathbf{X}^{B^{(m)}}, \psi^{B^{(m)}})\}_{m=1}^{M}$ be the set of relational structures, where $\mathbf{X}^{B^{(m)}} \in \mathbb{R}^{I_m \times N_m}$ denotes the relational matrix of m-th mode, $\psi^{B^{(m)}} : \{1, \cdots, N\} \to \{1, \cdots, N_m\}$ denotes the mapping from columns in the feature matrix $\mathbf{X}$ to columns within $\mathbf{X}^{B^{(m)}}$. To shorten notation, the index B is dropped from the mapping ψ^B whenever it is clear which block the mapping belongs to. From $\mathcal{B}$, one can reconstruct $\mathbf{X}$ by concatenating the corresponding columns of the relational matrices using the mappings. For instance, the feature vector $\mathbf{x}_n$ of the n-th case in the plain feature matrix $\mathbf{X}$ is represented as $\mathbf{x}_n = [\mathbf{x}^{(1)}_{\psi(n)}; \cdots; \mathbf{x}^{(M)}_{\psi(n)}]$. Fig. 3(b) shows an example how the feature matrix can be represented in relational structures. Let $N_z(\mathbf{A})$ denote the number of non-zeros in a matrix $\mathbf{A}$. The space required for using relational structures to represent the input data is $|\mathcal{B}| = NM + \sum_m N_z(\mathbf{X}^{B^{(m)}})$, which is much smaller than $N_z(\mathbf{X})$ if there are repeating patterns in the feature matrix $\mathbf{X}$.

Now we can rewrite the predictive model in Eq. (8) as follows

$$f(\{\mathcal{X}^{(v)}_n\}) = \sum_{v=1}^{V} \boldsymbol{\phi}^v \prod_{m \in S_M(v)} * \left(\mathbf{h}^{B^{(m)}}_{\psi(n)} + \mathbf{b}^{(v,m)^{\mathrm{T}}}\right), \tag{15}$$

with the caches $\mathbf{H}^{B^{(m)}} = [\mathbf{h}^{B^{(m)}}_1, \cdots, \mathbf{h}^{B^{(m)}}_{N_m}]$ for each mode, where $\mathbf{h}^{B^{(m)}}_j = \mathbf{\Theta}^{(m)^{\mathrm{T}}} \mathbf{x}^{B^{(m)}}_j$, $\forall j \in [1 : N_m]$.

This directly shows how N samples can be efficiently predicted: (i) compute $\mathbf{H}^{B^{(m)}}$ in $O(RN_z(\mathbf{X}^{B^{(m)}}))$ for each mode, (ii) compute N predictions with Eq. (15) using caches in $O(RN(V + \sum_v M_v))$. With the help of relational structures, SFMs can learn the same parameters and make the same predictions but with a much lower runtime complexity.

5 EXPERIMENTS

5.1 Datasets

To evaluate the ability and applicability of the proposed SFMs, we include a spectrum of large datasets from different domains. The statistics for each dataset is summarized in Table 2, the schema of the structural views in each dataset is presented in Fig. 4, and the details are as follows:

Table 2: The statistics for each dataset. $N_z(X)$ and $N_z(\mathcal{B})$ are the number of non-zeros in plain formatted feature matrix and in relational structures, respectively. Game: Video Games, Cloth: Clothing, Shoes and Jewelry, Sport: Sports and Outdoors, Health: Health and Personal Care, Home: Home and Kitchen, Elec: Electronics.

Dataset	#Samples	Mode					Density	$N_z(X)$	$N_z(\mathcal{B})$
Amazon		#Users	#Items	#Words	#Categories	#Links			
Game	231,780	24,303	10,672	7,500	193	17,974	0.089%	32.9M	15.2M
Cloth	278,677	39,387	23,033	3,493	1,175	107,139	0.031%	25.6M	7.3M
Sport	296,337	35,598	18,357	5,202	1,432	73,040	0.045%	34.2M	10.2M
Health	346,355	38,609	18,534	5,889	849	80,379	0.048%	33.6M	12.1M
Home	551,682	66,569	28,237	6,455	970	99,090	0.029%	46.8M	19.4M
Elec	1,689,188	192,403	63,001	12,805	967	89,259	0.014%	161.5M	69M
		#Users	#Venues	#Friends	#Categories	#Cities			
Yelp	1,319,870	88,009	40,520	88,009	892	412	0.037%	70.5M	1.4M
		#Users	#Books	#Countries	#Ages	#Authors			
BX	244,848	24,325	45,074	57	8	17,178	0.022%	1.2M	163K

Amazon[2]: The first group of datasets are from Amazon.com recently introduced by [31]. This is among the largest datasets available that include review texts and metadata of items. Each top-level category of products on Amazon.com has been constructed as an independent dataset in [31]. In this paper, we take a variety of large categories as listed in Tabel 2.

Each sample in these datasets has five modes, *i.e.*, users, items, review texts, categories, and linkage. The user mode and item mode are represented by one-hot encoding. The ℓ_2-normalized TF-IDF vector representation of review text [3] of the item given by the user is used as the text mode. The category mode and linkage mode consists of all the categories and all the co-purchasing items of the item, which might be from other categories. The last two modes are ℓ_1-normalized.

Yelp[4]: It is a large-scale dataset consisting of venue reviews. Each sample in this dataset contains five modes, *i.e.*, users, venues, friends, categories and cities. The user mode and venue mode are represented by one-hot encoding. The friend mode consists of the friends' ids of users. The category mode and city mode consists of all the categories and the city of the venue. The last three modes are ℓ_1-normalized.

BookCrossing (BX)[5]: It is a book review dataset collected from the Book-Crossing community. Each sample in this dataset contains five modes, *i.e.*, users, books, countries, ages and authors. The ages are split in eight bins as in [13]. The country mode and age mode consist of the corresponding meta information of the user. The author modes represents the authors of the book. All the modes are represented by one-hot encoding.

The values of samples range within [1:5] in Amazon and Yelp datasets, and range within [1:10] in BX dataset.

5.2 Comparison Methods

In order to demonstrate the effectiveness of the proposed SFMs, we compare a series of state-of-the-art methods.

Matrix Factorization (MF) is used to validate that meta information is helpful for improving prediction performance. We use the LIBMF implementation [8] for comparison in the experiment.
Factorization Machine (FM) [34] is the state-of-the-art method in recommender systems. We compare with its higher-order extension [2] with up to second-order, and third-order feature interactions, and denote them as FM-2 and FM-3.
Polynomial Network (PolyNet) [27] is a recently proposed method that utilizes polynomial kernel on all features. We compare the augmented PolyNet (which adds a constant one to the feature vector [3]) with up to the second-order, and third-order kernel and denote them as PolyNet-2 and PolyNet-3.
Multi-View Machine (MVM) [6] is a tensor factorization based method that explores the latent representation embedded in the full-order interactions among all the modes.
Structural Factorization Machine (SFM) is the proposed model that learns the common latent spaces shared in multi-way data.

5.3 Experimental Settings

For each dataset, we randomly split 50%, 10%, and 40% of labeled samples as training set, validation set, and testing set, respectively. Validation sets are used for hyper-parameter tuning for each model. Each of the validation and testing sets does not overlap with any other set so as to ensure the sanity of the experiment. For simplicity and fair comparison, in all the comparison methods, the dimension of latent factors $R = 20$ and the maximum number of epochs is set as 400 and we use early stop to obtain the best results for each method. Forbenius norm regularizers are used to avoid overfitting. The regularization hyper-parameter is tuned from $\{10^{-5}, 10^{-4}, \cdots, 10^{0}\}$.

All the methods except MF are implemented in TensorFlow, and the parameters are initialized using scaling variance initializer [14]. We tune the scaling factor of initializer σ from $\{1, 2, 5, 10, 100\}$ and the learning rate η from $\{0.01, 0.1, 1\}$ using the validation sets. In the experiment, we set $\sigma = 2$ (default setting in TensorFlow) and $\eta = 0.01$ for these methods except MVM. We found that MVM is more sensitive to the configuration, because MVM will element-wisely multiply the latent factors of all the modes which leads to an extremely small value approaching zero. $\sigma = 10$ and $\eta = 0.1$ yielded the best performance for MVM.

[2] http://jmcauley.ucsd.edu/data/amazon/
[3] Stemming, lemmatization, removing stop-words and words with frequency less than 100 times, etc., are handled beforehand.
[4] https://www.yelp.com/dataset-challenge
[5] http://www2.informatik.uni-freiburg.de/~cziegler/BX/

Table 3: MSE comparison on all the datasets. The best results are listed in bold.

Dataset	(a) MF	(b) MVM	(c) FM-2	(d) FM-3	(e) PolyNet-2	(f) PolyNet-3	(g) SFM	Improvement of SFM verus b	min(c,d)	min(e,f)
Game	1.569 ± 0.005	0.753 ± 0.007	0.764 ± 0.006	0.749 ± 0.007	0.749 ± 0.004	0.748 ± 0.006	**0.723 ± 0.006**	4.06%	3.52%	3.35%
Cloth	1.624 ± 0.009	0.725 ± 0.046	0.678 ± 0.004	0.679 ± 0.004	0.678 ± 0.007	0.680 ± 0.005	**0.659 ± 0.013**	9.03%	2.82%	2.84%
Sport	1.290 ± 0.004	0.646 ± 0.019	0.638 ± 0.003	0.632 ± 0.007	0.631 ± 0.005	0.632 ± 0.005	**0.614 ± 0.011**	5.00%	2.91%	2.79%
Health	1.568 ± 0.007	0.807 ± 0.012	0.779 ± 0.004	0.778 ± 0.004	0.779 ± 0.005	0.776 ± 0.005	**0.763 ± 0.019**	5.47%	2.02%	1.77%
Home	1.591 ± 0.004	0.729 ± 0.067	0.714 ± 0.002	0.714 ± 0.004	0.690 ± 0.003	0.692 ± 0.005	**0.678 ± 0.008**	6.93%	5.00%	1.72%
Elec	1.756 ± 0.002	0.792 ± 0.042	0.776 ± 0.006	0.749 ± 0.007	0.760 ± 0.004	0.757 ± 0.001	**0.747 ± 0.006**	5.69%	0.27%	1.33%
Yelp	1.713 ± 0.003	1.2575 ± 0.013	1.277 ± 0.002	1.277 ± 0.002	1.272 ± 0.002	1.272 ± 0.002	**1.256 ± 0.010**	0.09%	1.58%	1.19%
BX	4.094 ± 0.025	2.844 ± 0.024	2.766 ± 0.012	2.767 ± 0.014	2.654 ± 0.013	2.658 ± 0.013	**2.541 ± 0.025**	10.66%	8.16%	4.27%
Average on all datasets								5.87%	3.29%	2.41%

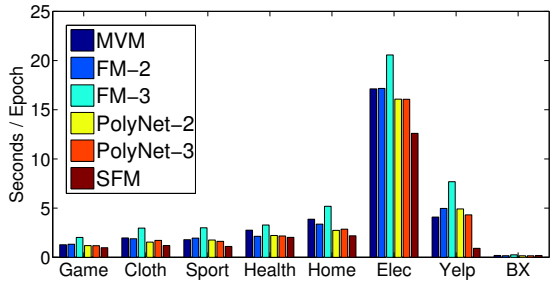


Figure 5: Training Time (Seconds/Epoch) Comparison.

To investigate the performance of comparison methods, we adopt mean squared error (MSE) on the test data as the evaluation metrics [30, 45]. The smaller value of the metric indicates the better performance. Each experiment was repeated for 10 times, and the mean and standard deviation of each metric in each data set were reported. All experiments are conducted on a single machine with Intel Xeon 6-Core CPUs of 2.4 GHz and equipped with a Maxwell Titan X GPU.

5.4 Performance Analysis

The experimental results are shown in Table 3. The best method of each dataset is in bold. For clarity, on the right of the tables we show the percentage improvement of the proposed SFM method over a variety of methods. From these results, we can observe that SFM consistently outperforms all the comparison methods. We also make a few comparisons and summarize our findings as follows.

Compared with MF, SFM performs better with an average improvement of nearly 50%. MF usually performs well in practice [26, 34], while in datasets which are extremely sparse, as is shown in our case, MF is unable to learn an accurate representation of users/items. Thus MF under-performs other methods which takes the meta information into consideration.

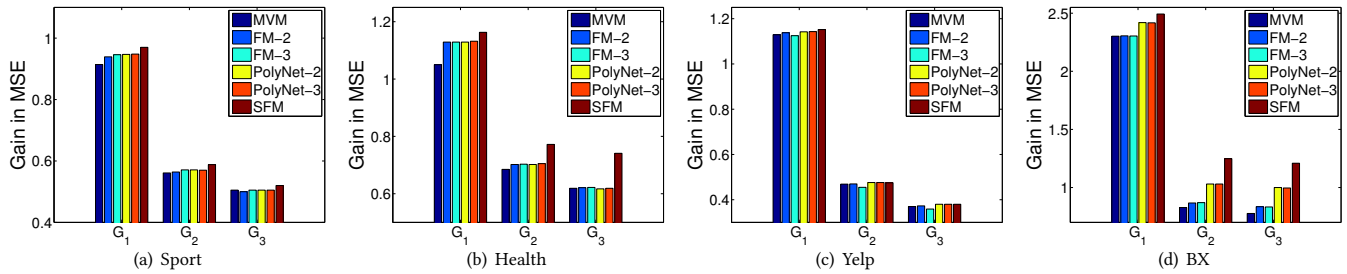
In both FM and PolyNet methods, the feature vectors from all the modes are concatenated as a single input feature vector. The major difference between these two methods is the choice of kernel applied [2]. The polynomial kernel used in PolyNet considers all monomials (the products of features), i.e., all combinations of features *with* replacement. The ANOVA kernel used in FM considers only monomials composed of distinct features, i.e., feature combinations *without* replacement. Compared with the best results obtained from FM methods and from PolyNet methods, SFM leads to an average improvement of 3.3% and 2.4% in MSE, respectively.

The primary reason behind the results is how the latent factors of each feature are learned. For any factorization based method, the latent factors of a feature are essentially learned from its interactions with other features observed in the data, as can be observed from its update rule. In FM and PolyNet, all the feature interactions are taken into consideration without distinguishing the features from different modes. As a result, important feature interactions (e.g., the interactions between the given user and her friend) would be easily buried in irrelevant feature interactions from the same modes (e.g., the interactions between the friends of the same user). Hence, the learned latent factors are less representative in FM and PolyNet, compared with the proposed SFM. Besides, we can find that including higher-order interactions in FM and PolyNet (i.e., FM-3 and PolyNet-3) does not always improve the performance. Instead, it may even degrade the performance, as shown in Cloth, Yelp, and BX datasets. This is probably due to overfitting, as they need to include more parameters to model the interactions in higher orders while the datasets are extremely sparse such that the parameters cannot be properly learned.

Compared to the MVM method, which models the full-order interactions among all the modes, our proposed SFM leads to an average improvement of 5.87%. This is because not all the modes are relevant, and some irrelevant feature interactions may introduce unexpected noise to the learning task. The irrelevant information can even be exaggerated after combinations, thereby degrading performance. This suggests that preserving the nature of relational structure is important in building predictive models.

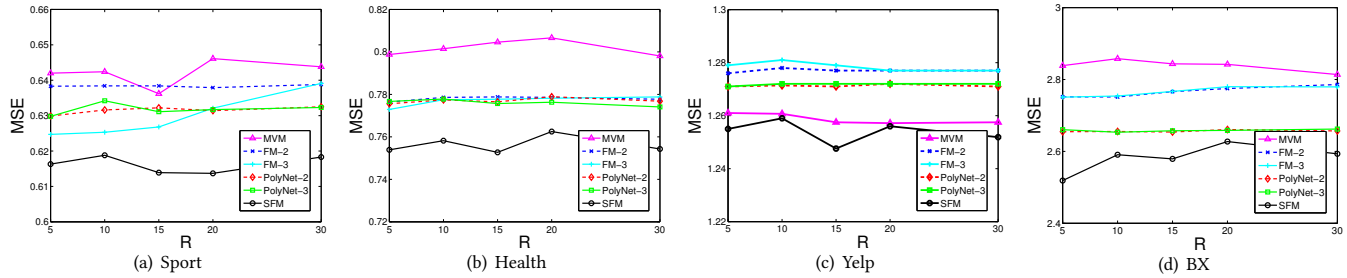
5.5 Computational Cost Analysis

Next, we investigate the computational cost for comparison methods. The averaged training time (seconds per epoch) required for each dataset is shown in Fig. 5. We can easily find that the proposed SFM requires much less computational cost on all the datasets, especially for the Yelp dataset (roughly 11% of computational cost required for training FM-3). The efficiency comes from the use of relational structure representation. As shown in Table 2, the number of non-zeros of the feature matrix $N_z(\mathbf{X})$ is much larger than the number of non-zeros of the relational structure representation $N_z(\mathcal{B})$. The amount of repeating patterns is much higher for the Yelp dataset than for the other dataset, because adding all the



(a) Sport (b) Health (c) Yelp (d) BX

Figure 6: Performance gain in MSE compared with MF for users with limited training samples. G_1, G_2, and G_3 are groups of users with $[1, 3]$, $[4, 6]$, and $[7, 10]$ observed samples in the training set, respectively.



(a) Sport (b) Health (c) Yelp (d) BX

Figure 7: Sensitivity analysis of the latent dimension R.

friends of a user significantly increases results in large repeating blocks in the plain feature matrix. Standard ML algorithms like the compared methods have typically at best a linear complexity in $N_z(\mathbf{X})$, while using the relational structure representation for SFM have a linear complexity in $N_z(\mathcal{B})$. This experiment substantiates the efficiency of the proposed SFM for large datasets.

5.6 Analysis of the Impact of Data Sparsity

We proceed by further studying the impact of data sparsity on different methods. As can be found in the experimental results, the improvement of SFM over the traditional collaborative filtering methods (e.g., MF) is significant for datasets that are sparse, mainly because the number of samples is too scarce to model the items and users adequately. We verify this finding by comparing the performance of comparison methods with MF on users with limited training data. Shown in Fig. 6 is the gain of each method compared with MF for users with limited training samples, where G_1, G_2, and G_3 are groups of users with $[1, 3]$, $[4, 6]$, and $[7, 10]$ observed samples in the training set. Due to space limit, we only report the results from two Amazon datasets (Sport and Health) while the observations still hold for the rest datasets. It can be seen that the proposed SFM gains the most in group G_1, in which the users have extremely few training items. The performance gain starts to decrease with the number of training items available for each user. The results indicate that including meta information can be valuable information especially when limited information available.

5.7 Sensitivity analysis

The number of latent factors R is an important hyperparameter for the factorization models. We analyze different values of R and report the averaged results in Fig. 7. The results again show that SFM consistently outperforms other methods with various values of R. In contrast to findings in other related factorization models [42] where prediction error can steadily get reduced with larger R, we observe that the performance of each method is rather stable even with the increasing of R. It is reasonable in a general sense, as the expressiveness of the model is enough to describe the information embedded in data. Although larger R renders the model with greater expressiveness, when the available observations regarding the target values are too sparse but the meta information is rich, only a few number of factors are required to fit the data well.

6 CONCLUSIONS

In this paper, we introduce a generic framework for learning structural data from heterogeneous domains, which can explore the high order correlations underlying multi-view multi-way data. We develop structural factorization machines (SFMs) that learn the common latent spaces shared in the multi-view tensors while automatically adjust the contribution of each view in the predictive model. With the help of relational structure representation, we further provide an efficient approach to avoid unnecessary computation costs on repeating patterns of the multi-view data. It was shown that the proposed SFMs outperform state-of-the-art factorization models on eight large-scale datasets in terms of prediction accuracy and computational cost.

ACKNOWLEDGMENTS

This work is supported in part by NSF through grants IIS-1526499, and CNS-1626432, and NSFC 61672313, 61503253 and NSF of Guangdong Province (2017A030313339).

REFERENCES

[1] Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. 2011. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv preprint arXiv:1105.3422* (2011).

[2] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-Order Factorization Machines. In *Advances in Neural Information Processing Systems*. 3351–3359.

[3] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. 2016. Polynomial Networks and Factorization Machines: New Insights and Efficient Training Algorithms. In *Proceedings of the 33nd International Conference on Machine Learning*. 850–858.

[4] Bokai Cao, Lifang He, Xiangnan Kong, Philip S. Yu, Zhifeng Hao, and Ann B. Ragin. 2014. Tensor-Based Multi-view Feature Selection with Applications to Brain Diseases. In *IEEE International Conference on Data Mining*. 40–49.

[5] Bokai Cao, Lei Zheng, Chenwei Zhang, Philip S Yu, Andrea Piscitello, John Zulueta, Olu Ajilore, Kelly Ryan, and Alex D Leow. 2017. DeepMood: Modeling Mobile Phone Typing Dynamics for Mood Detection. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*. 747–755.

[6] Bokai Cao, Hucheng Zhou, Guoqiang Li, and Philip S Yu. 2016. Multi-view Machines. In *ACM International Conference on Web Search and Data Mining*. 427–436.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *DLRS*. ACM, 7–10.

[8] Wei-Sheng Chin, Bo-Wen Yuan, Meng-Yuan Yang, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. 2016. LIBMF: A library for parallel matrix factorization in shared-memory systems. *The Journal of Machine Learning Research* 17, 1 (2016), 2971–2975.

[9] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *ACM Recommender Systems Conference (RecSys)*. ACM, 191–198.

[10] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12 (2011), 2121–2159.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *arXiv preprint arXiv:1703.04247* (2017).

[12] Hongyu Guo and Herna L Viktor. 2006. Mining relational data through correlation-based multiple view validation. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*. 567–573.

[13] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. 1026–1034.

[15] Lifang He, Xiangnan Kong, S Yu Philip, Ann B Ragin, Zhifeng Hao, and Xiaowei Yang. 2014. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. *matrix* 3, 1 (2014), 2.

[16] Lifang He, Chun-Ta Lu, Jiaqi Ma, Jianping Cao, Linlin Shen, and Philip S Yu. 2016. Joint community and structural hole spanner detection via harmonic modularity. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 875–884.

[17] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[18] G Hinton, N Srivastava, and K Swersky. 2012. RMSProp: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e* (2012).

[19] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *ACM International Conference on Information and Knowledge Management*. ACM, 2333–2338.

[20] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 43–50.

[21] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.

[23] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*. 426–434.

[24] Yehuda Koren. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4, 1 (2010), 1.

[25] Tingting Liang, Lifang He, Chun-Ta Lu, Liang Chen, Philip S. Yu, and Jian Wu. 2017. A Broad Learning Approach for Context-Aware Mobile Application Recommendation. In *2017 IEEE International Conference on Data Mining (ICDM)*. 955–960.

[26] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*. 105–112.

[27] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. 2014. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*. 855–863.

[28] Chun-Ta Lu, Lifang He, Weixiang Shao, Bokai Cao, and Philip S. Yu. 2017. Multilinear Factorization Machines for Multi-Task Multi-View Learning. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 701–709.

[29] Chun-Ta Lu, Sihong Xie, Weixiang Shao, Lifang He, and Philip S Yu. 2016. Item recommendation for emerging online businesses. In *Proceedings of International Joint Conference Artificial Intelligence*. 3797–3803.

[30] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. 165–172.

[31] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*. 785–794.

[32] Alexander Novikov, Mikhail Trofimov, and Ivan Oseledets. 2017. Exponential machines. In *International Conference on Learning Representations*.

[33] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 1149–1154.

[34] Steffen Rendle. 2012. Factorization machines with libFM. *Intelligent Systems and Technology* 3, 3 (2012), 57.

[35] Steffen Rendle. 2013. Scaling factorization machines to relational data. In *Proceedings of the VLDB Endowment*, Vol. 6. VLDB Endowment, 337–348.

[36] Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 81–90.

[37] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep Crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 255–262.

[38] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*. 650–658.

[39] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. *arXiv preprint arXiv:1708.05123* (2017).

[40] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *International Joint Conference on Artificial Intelligence*.

[41] Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *arXiv:1304.5634* (2013).

[42] Ling Yan, Wu-jun Li, Gui-Rong Xue, and Dingyi Han. 2014. Coupled Group Lasso for Web-Scale CTR Prediction in Display Advertising. In *International Conference on Machine Learning*. 802–810.

[43] Jingyuan Zhang, Chun-Ta Lu, Bokai Cao, Yi Chang, and Philip S. Yu. 2017. Connecting Emerging Relationships from News via Tensor Factorization. In *Proceedings of IEEE International Conference on Big Data*. IEEE.

[44] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *European conference on information retrieval*. Springer, 45–57.

[45] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 425–434.

[46] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Xiao Ma, Yanghui Yan, Xingya Dai, Han Zhu, Junqi Jin, Han Li, and Kun Gai. 2017. Deep Interest Network for Click-Through Rate Prediction. *arXiv preprint arXiv:1706.06978* (2017).

[47] Jie Zhu, Ying Shan, JC Mao, Dong Yu, Holakou Rahmanian, and Yi Zhang. 2017. Deep Embedding Forest: Forest-based Serving with Deep Embedding Features. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*.