# Multilinear Factorization Machines for Multi-Task Multi-View Learning

Chun-Ta Lu[†], Lifang He[‡]*, Weixiang Shao[†], Bokai Cao[†], and Philip S. Yu[†§]
[†]Department of Computer Science, University of Illinois at Chicago, IL, USA
[‡]Institute for Computer Vision, Shenzhen University, China
[§]Institute for Data Science, Tsinghua University, Beijing, China
{clu29,wshao4,caobokai}@uic.edu, lifanghescut@gmail.com, psyu@cs.uic.edu

## ABSTRACT

Many real-world problems, such as web image analysis, document categorization and product recommendation, often exhibit dual-heterogeneity: heterogeneous features obtained in multiple views, and multiple tasks might be related to each other through one or more shared views. To address these Multi-Task Multi-View (MTMV) problems, we propose a tensor-based framework for learning the predictive multilinear structure from the full-order feature interactions within the heterogeneous data. The usage of tensor structure is to strengthen and capture the complex relationships between multiple tasks with multiple views. We further develop efficient multilinear factorization machines (MFMs) that can learn the task-specific feature map and the task-view shared multilinear structures, without physically building the tensor. In the proposed method, a joint factorization is applied to the full-order interactions such that the consensus representation can be learned. In this manner, it can deal with the partially incomplete data without difficulty as the learning procedure does not simply rely on any particular view. Furthermore, the complexity of MFMs is linear in the number of parameters, which makes MFMs suitable to large-scale real-world problems. Extensive experiments on four real-world datasets demonstrate that the proposed method significantly outperforms several state-of-the-art methods in a wide variety of MTMV problems.

## CCS Concepts

**•Computing methodologies → Multi-task learning; Factorization methods; Learning latent representations;**

## Keywords

Multi-task; Multi-view; Tensor Factorization

*Corresponding author.

*WSDM 2017, February 06 - 10, 2017, Cambridge, United Kingdom*
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4675-7/17/02...$15.00
DOI: http://dx.doi.org/10.1145/3018661.3018716

## 1. INTRODUCTION

In the era of big data, it is becoming common to have heterogeneous data obtained in multiple views or extracted from multiple sources, known as "multi-view data". For example, in web image retrieval, the visual information of images and their textual descriptions can be regarded as two views; for scientific document categorization, each paper has word features and its citations. Multi-View Learning (MVL) was proposed to combine different views to obtain better performance than relying on just one single view [4, 25, 27, 26]. In addition, different learning tasks might be related with each other through shared features. For example, for product recommendation systems, different product domains can be viewed as different tasks and they might share certain word features in product reviews, *e.g.*, good, great and bad. Multi-Task Learning (MTL) was developed to learn multiple related tasks together so as to improve the performance of each task relative to learning them separately [1, 2, 7, 8, 11, 12, 13].

Existing MVL or MTL approaches only capture one type of heterogeneity, while real-world problems exhibit dual-heterogeneity (both feature heterogeneity and task heterogeneity) [14, 15, 28]. Consider the multi-task recommendation problem, reviews may also contain multi-view data such as words, emotion icons, images and web links to other products. Such type of problem is Multi-Task Multi-View (MTMV) learning. Though there are a wide variety of applications of MTMV learning, only a few works have addressed this problem. Recently, a graph-based iterative algorithm, IteM$^2$ [14], was first introduced for MTMV learning with applications to text classification, while it can only deal with nonnegative feature values. Assuming the predictive models should be consistent among different views, co-regularization based algorithms, such as regMVMT [28] and CSL-MTMV [15], imposed a regularization term to enforce the difference of the predictive functions among different views to be small.

Nonetheless, different views might not be consistent with each other especially for heterogeneous data; instead, they provide complementary information. For example, textual view (*e.g.*, bag-of-words) and topological view (*e.g.*, citations in the document categorization or web links in the product recommendation) usually provide complementary information. In contrast to building a distinct model for each view or each task, another direction is to mine the hidden interactions/correlations among MTMV features.

In this paper, we propose a general framework for learning the predictive multilinear structure from the complex relationships within the heterogeneous data. Specifically, we model the multimodal interactions among multiple tasks and multiple views as a tensor structure, by taking the tensor product of their respective feature spaces. As the interactions with different orders can reflect different but complementary insights [4, 22], in the proposed framework, the full-order interactions[1] observed in the heterogeneous data are used collectively to learn the consensus representation. In this manner, it can also deal with the partially incomplete data without difficulty because the learning procedure does not simply rely on any particular view.

Constructing the full-order tensor may not be realistic for real-world applications, as the model parameters can be exponential growth. We further propose multilinear factorization machines (MFMs) that can efficiently learn the task-specific feature map and the task-view shared multilinear structures, without physically building the tensor. A joint factorization is applied for the model parameters which makes parameter estimation more accurate under sparsity and renders the model with the capacity to avoid overfitting. Furthermore, the model complexity of the proposed MFMs is linear in the feature dimensionality, making it applicable to large-scale real-world applications.

The contributions of this paper are summarized as follows:

- The proposed framework is widely applicable to multiple types of heterogeneous machine learning problems, by learning predictive multilinear structures from the complex relationships within the heterogeneous data.
- We propose multilinear factorization machines (MFMs) that can efficiently learn the task-specific feature map and the task-view shared multilinear structures from full-order interactions.
- Extensive experiments on four real-world datasets demonstrate that the proposed MFM methods outperform several state-of-the-art methods in a wide variety of MTMV problems, including classification tasks and regression tasks.

The rest of this paper is organized as follows. We introduce the problem definition and some preliminary works in Section 2. We then propose the framework for learning predictive multilinear structures to address the MTMV problems, and develop the multilinear factorization machines (MFMs) in Section 3. The experimental results and parameter analysis are reported in Section 4. In Section 5, we briefly review related work on MVL, MTL and MTMV learning. Section 6 concludes this paper.

## 2. PRELIMINARIES

The key to this work is to apply the tensor structure to fuse all possible dependence relationships among different views and different tasks. We begin by introducing some related concepts and notation about tensor, and then state the problem of multi-task multi-view learning. Table 1 lists basic symbols that will be used throughout the paper.

[1]Full-order interactions range from the first-order interactions (i.e., contributions of single features in each task) to the highest-order interactions (i.e., contributions of the tensor product of features from each view and each task).

**Table 1: List of basic symbols.**

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

### 2.1 Tensor Basics and Notation

Tensors are higher order arrays that generalize the notion of vectors (first order) and matrices (second order). Following [16], an $M$-th order tensor is denoted by $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_M}$ and its elements by $x_{i_1,\cdots,i_M}$. An index is denoted by a lowercase letter, spanning the range from 1 to the uppercase letter of the index, *e.g.*, $i = 1, 2, \cdots, I$. 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. The inner product of two same-sized tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1\times\cdots\times I_M}$ is defined by $\langle\mathcal{X},\mathcal{Y}\rangle = \sum_{i_1=1}^{I_1}\cdots\sum_{i_1=1}^{I_M} x_{i_1,\cdots,i_M} y_{i_1,\cdots,i_M}$. The outer product of $M$ vectors $\mathbf{x}^{(m)} \in \mathbb{R}^{I_m}$ for $m \in [1:M]$ is an $M$-th order tensor and defined elementwise by $\left(\mathbf{x}^{(1)}\circ\cdots\circ\mathbf{x}^{(M)}\right)_{i_1,\cdots,i_M} = x_{i_1}^{(1)}\cdots x_{i_M}^{(M)}$ for all values of the indices. In particular, for $\mathcal{X} = \mathbf{x}^{(1)}\circ\cdots\circ\mathbf{x}^{(M)}$ and $\mathcal{Y} = \mathbf{y}^{(1)}\circ\cdots\circ\mathbf{y}^{(M)}$, it holds that

$$\langle\mathcal{X},\mathcal{Y}\rangle = \prod_{m=1}^{M}\langle\mathbf{x}^{(m)},\mathbf{y}^{(m)}\rangle = \prod_{m=1}^{M}\mathbf{x}^{(m)\mathrm{T}}\mathbf{y}^{(m)} \tag{1}$$

For a general tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_M}$, its CANDECOMP / PARAFAC (CP) factorization is

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

where for $m \in [1:M]$, $\mathbf{X}^{(m)} = [\mathbf{x}_1^{(m)},\cdots,\mathbf{x}_R^{(m)}]$ are factor matrices of size $I_m \times R$, $R$ is the number of factors, and $[\![\cdot]\!]$ is used for shorthand.

### 2.2 Problem Formulation

Suppose that the problem includes $T$ tasks and $V$ views. Let $N_t$ be the number of labeled instances in the task $t \in [1:T]$, thus we have $N = \sum_t N_t$ labeled instances in total. Let $I_v$ be the dimensionality of the view $v \in [1:V]$ and denote $I = \sum_v I_v$. Assuming the problem is associated with training data $\mathcal{D} = \left\{(\mathbf{X}_t^{(1)},\ldots,\mathbf{X}_t^{(V)}, \mathbf{y}_t) \mid t \in [1:T]\right\}$, where $\mathbf{X}_t^{(v)} \in \mathbb{R}^{I_v\times N_t}$ is the feature matrix in the $t$-th task for the $v$-th view; $\mathbf{y}_t$ is the vector of the responses of the $t$-th task. Our goal is to leverage the label information from all the tasks to help predict the unlabeled instances in each task, as well as to use the complementary among different views of a single task to improve the performance. Specifically, for the $t$-th task, we are interested in finding a predictive function $f_t : \mathcal{X}_t \to \mathcal{Y}_t$ that minimizes the expected loss, where $\mathcal{X}_t$ is the input space and $\mathcal{Y}_t$ is the output space.

Following the traditional supervised learning framework, we learn $T$ functions $\{f_t\}_{t=1}^{T}$ from the data that minimize the following regularized empirical risk:

$$\mathcal{R}(\{f_t\}_{t=1}^T) = \sum_{t=1}^{T}\left(\sum_{n=1}^{N_t}\frac{1}{N_t}\ell\left(f_t(\{\mathbf{x}_{t,n}^{(v)}\}), y_{t,n}\right) + \lambda\Omega(f_t)\right) \tag{3}$$

where $\ell$ is a prescribed loss function, $\Omega$ is the regularizer encoding the prior knowledge of $f_t$, and $\lambda > 0$ is the regularization hyperparameter that controls the trade-off between the empirical loss and the prior knowledge. In this formulation, we weight each task equally (by dividing the number of instances $N_t$) so that no task will dominate the others. One may also choose other weighting schemes. The empirical loss of the training data in the $t$-th task is

$$\mathcal{L}_t(f_t(\{\mathbf{X}_t^{(v)}\}), \mathbf{y}) = \frac{1}{N_t}\sum_{n=1}^{N_t}\ell\left(f_t(\{\mathbf{x}_{t,n}^{(v)}\}), y_{t,n}\right) \tag{4}$$

The choice of the loss function $\ell$ depends on learning tasks. To conduct regression, for example, one can use the squared loss, and for classification problems, one can use the logistic loss or the hinge loss. The regularizer is chosen based on our prior knowledge about the model parameters.

For solving MTMV problems, a straightforward approach is to concatenate feature vectors from different views and apply the multi-task learning algorithms. However, transforming a multi-view data into a single-view data would fail to leverage the underlying correlations between different views, wherein complementary information is contained. Through the employment of a nonlinear kernel, one can implicitly project data from the feature space into a more complex high-dimensional space, which allows modeling higher order interactions between features. However, as discussed in [22], all interaction parameters of nonlinear are completely independent. Besides, for multiple kernel learning [9, 26], the coefficients are learned based on the informativeness of the corresponding views, and thus inter-view correlations are only considered at the view-level. These approaches, however, fail to explore the explicit correlations between features across multiple views.

Currently, there are only a few researches on MTMV learning. The IteM$^2$ algorithm [14] projects any two tasks to a new RKHS based on the common views shared by the given two tasks, and has been shown empirically outperforming multiple kernel approaches. Assuming the predictive models should be consistent among different views, co-regularization based methods were later developed [15, 28]. These methods assume that all the views are similar to each other, while such assumption may not be appropriate especially for heterogeneous data. Furthermore, the co-regularization approaches involve pairwise comparison of the prediction from different views, which leads to high model complexity (the space and time complexity are quadratic and cubic in the total number of features, respectively) [15, 28]. Thus, they can hardly be applied to high-dimensional data.

In the following, we will introduce a general framework that intrinsically models the complex relationships in multi-modal interactions among multiple tasks and multiple views as a tensor structure.

## 3. MULTILINEAR STRUCTURE LEARNING FOR MTMV PROBLEMS

In this section, we first discuss how to design the multilinear predictive models for learning the full-order interactions among MTMV data. We then derive efficient Multilinear Factorization Machines (MFMs) that can learn the shared multilinear structure in linear complexity.

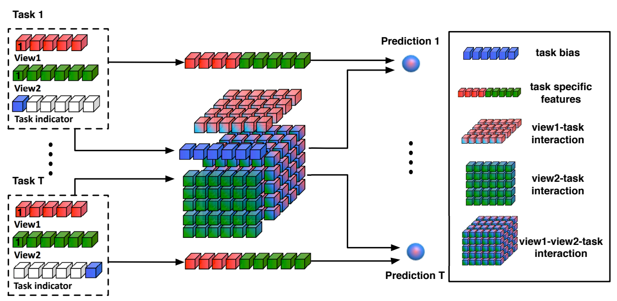


**Figure 1: Multilinear Predictive Models. The feature interactions in multi-task multi-view data are modeled in a full-order tensor. The prediction is learned from the task-specific feature space and the task-view shared multilinear feature space.**

### 3.1 Multilinear Predictive Models

Given an input vector $\mathbf{x} \in \mathbb{R}^I$, the linear model for the $t$-th task is given by

$$f_t(\mathbf{x}) = \sum_{i=1}^{I} w_t x_i = \mathbf{x}^{\mathrm{T}}\mathbf{w}_t \tag{5}$$

where $\mathbf{w}_t \in \mathbb{R}^I$ is the weight vector for linear effects. Let $\mathbf{W} \in \mathbb{R}^{I\times T}$ denote the weight matrix to be learned, whose columns are the vector $\mathbf{w}_t$. Most of the MTL algorithms aim at solving the following regularized problem

$$\min \sum_{t=1}^{T}\mathcal{L}_t\left(\mathbf{X}_t^{\mathrm{T}}\mathbf{w}_t, \mathbf{y}_t\right) + \lambda\Omega(\mathbf{W}) \tag{6}$$

Many different assumptions about how tasks are related have been proposed for MTL, leading to different regularization terms (e.g., $\ell_1/\ell_q$-norm regularization, trace norm regularization, and composite regularization) in the formulation [1, 2, 7, 8, 13].

In fact, the joint learning of multiple linear models for multiple tasks is essentially to learn the bilinear map for modeling the second-order interactions between input features and tasks. Let $\mathbf{e}_t \in \mathbb{R}^T$ denote the task indicator vector

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

We then have that

$$f_t(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{w}_t = \mathbf{x}^{\mathrm{T}}\mathbf{W}\mathbf{e}_t = \langle \mathbf{W}, \mathbf{x}\circ\mathbf{e}_t\rangle = f(\{\mathbf{x}, \mathbf{e}_t\}) \tag{7}$$

Similarly, we can form a multilinear function for modeling the higher-order interactions in MTMV data. Assume we are given two views, for example, we can learn the third-order interactions by

$$f_t(\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\}) = \mathbf{x}^{(1)^{\mathrm{T}}}\mathbf{W}_t\mathbf{x}^{(2)} = \left\langle \mathcal{W}, \mathbf{x}^{(1)}\circ\mathbf{x}^{(2)}\circ\mathbf{e}_t\right\rangle \tag{8}$$

where $\mathcal{W} \in \mathbb{R}^{I_1\times I_2\times T}$ is the weight tensor to be learned.

However, only the highest-order interactions are explored in this way, and such interactions are limited in sparse data, especially when one or more views are missing in some instances. In contrast, the lower-order (*e.g.*, pairwise) interactions can usually explain the data sufficiently [24, 22], and incorporating the lower-order interactions in the predictive models can further improve the performance [4, 22]. Hence, we consider nesting all interactions up to full-order:

$$f_t(\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\}) = w_t + \sum_{v=1}^{2} \mathbf{x}^{(v)\mathrm{T}} \mathbf{w}_t^{(v)} + \mathbf{x}^{(1)\mathrm{T}} \mathbf{W}_t \mathbf{x}^{(2)} \tag{9}$$

This can be done by adding an extra feature with constant value 1 to the vector $\mathbf{x}_t^{(v)}$, *i.e.*, $\mathbf{z}_t^{(v)} = [1; \mathbf{x}_t^{(v)}] \in \mathbb{R}^{1+I_v}$. Then Eq. (9) can be rewritten as

$$f_t(\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\}) = \left\langle \mathcal{W}, \mathbf{z}^{(1)} \circ \mathbf{z}^{(2)} \circ \mathbf{e}_t \right\rangle = \langle \mathcal{W}, \mathcal{Z}_t \rangle \tag{10}$$

We can easily extend Eq. (10) to the MTMV problems with more views. Formally, let $\mathcal{Z}_t = \mathbf{z}^{(1)} \circ \cdots \circ \mathbf{z}^{(V)} \circ \mathbf{e}_t \in \mathbb{R}^{(1+I_1)\times\cdots\times(1+I_V)\times T}$ be the full-order tensor, and let $\mathcal{W} = \{w_{i_1,\ldots,i_V,t}\} \in \mathbb{R}^{(1+I_1)\times\cdots\times(1+I_V)\times T}$ be the weight tensor to be learned. The multilinear map function can be defined as

$$f_t(\{\mathbf{x}^{(v)}\}) = \langle \mathcal{W}, \mathcal{Z}_t \rangle = \sum_{s=1}^{T} \sum_{i_1=0}^{I_1} \cdots \sum_{i_V=0}^{I_V} w_{i_1,\ldots,i_V,s} \left( e_{t,s} \prod_{v=1}^{V} z_{i_v}^{(v)} \right) \tag{11}$$

It is worth noting that $w_{i_1,\ldots,i_V,s}$ with some indexes satisfying $i_v = 0$ encodes lower-order interactions between views whose $i_{v'} > 0$.

So far, multiple tasks with multi-view features are able to be incorporated into an elegant tensor formulation, where the complex multiple relationships among tasks and views are embedded within the tensor structures. However, it could be too restrictive to constrain all tasks to share a common set of features [8, 13]. Thus, following the structural learning framework for MTL [1, 7], we consider learning a predictive function from both the original feature spaces and the multilinear feature interaction space:

$$f_t(\{\mathbf{x}^{(v)}\}) = \mathbf{x}^{\mathrm{T}} \mathbf{u}_t + \langle \mathcal{W}, \mathcal{Z}_t \rangle \tag{12}$$

where $\mathbf{x} = [\mathbf{x}^{(1)}; \ldots; \mathbf{x}^{(V)}] \in \mathbb{R}^I$ is the concatenated feature vector from multiple views, and $\mathbf{u}_t \in \mathbb{R}^I$ is the task-specific weight vector. Figure 1 illustrates the proposed multilinear predictive model with two views.

## 3.2 Multilinear Factorization Machines

Directly learning the weight tensor $\mathcal{W}$ leads to two drawbacks. First, transfer learning is not possible straight from the model since the weight parameters are learned independently for different tasks and different views. Second, the number of parameters in Eq. (11) is $T \prod_{v=1}^{V}(1 + I_v)$, which can make the model prone to overfitting and ineffective on sparse data. Hence, we assume that the effect of interactions has a low rank and $\mathcal{W}$ can be factorized as

$$\mathcal{W} = [\![\mathbf{\Theta}^{(1)}, \ldots, \mathbf{\Theta}^{(V)}, \mathbf{\Phi}]\!]$$

by CP factorization. The factor matrix $\mathbf{\Theta}^{(v)} \in \mathbb{R}^{(1+I_v)\times R}$ is the shared structure matrix for the $v$-th view and the $t$-th row $\boldsymbol{\phi}^t$ within $\mathbf{\Phi}$ is the task specific weight vector for the $t$-th task. Then Eq. (11) is transformed into

$$\begin{aligned} \langle \mathcal{W}, \mathcal{Z}_t \rangle &= \sum_{s=1}^{T} \sum_{i_1=0}^{I_1} \cdots \sum_{i_V=0}^{I_V} \left( \sum_{r=1}^{R} \phi_{s,r} \prod_{v=1}^{V} \theta_{i_v,r}^{(v)} \right) \left( e_{t,s} \prod_{v=1}^{V} z_{i_v}^{(v)} \right) \\ &= \sum_{r=1}^{R} \left( \sum_{s=1}^{T} \phi_{s,r} e_{t,s} \right) \sum_{i_1=0}^{I_1} \cdots \sum_{i_V=0}^{I_V} \left( \prod_{v=1}^{V} \theta_{i_v,r}^{(v)} z_{i_v}^{(v)} \right) \\ &= \sum_{r=1}^{R} \left\langle \boldsymbol{\theta}_r^{(1)} \circ \cdots \circ \boldsymbol{\theta}_r^{(V)} \circ \boldsymbol{\phi}_r \ , \ \mathbf{z}^{(1)} \circ \cdots \mathbf{z}^{(V)} \circ \mathbf{e}_t \right\rangle \end{aligned} \tag{13}$$

Because $e_{t,s} = 1$ only when $t = s$ and according to Eq. (1), we can further rewrite Eq. (13) into

$$\begin{aligned} \langle \mathcal{W}, \mathcal{Z}_t \rangle &= \sum_{r=1}^{R} \left( \mathbf{z}^{(1)\mathrm{T}} \boldsymbol{\theta}_r^{(1)} \right) \cdots \left( \mathbf{z}^{(V)\mathrm{T}} \boldsymbol{\theta}_r^{(V)} \right) \phi_{t,r} \\ &= \boldsymbol{\phi}^t \left( \left( \mathbf{z}^{(1)\mathrm{T}} \mathbf{\Theta}^{(1)} \right) * \cdots * \left( \mathbf{z}^{(V)\mathrm{T}} \mathbf{\Theta}^{(V)} \right) \right)^{\mathrm{T}} \\ &= \boldsymbol{\phi}^t \prod_{v=1}^{V} * \left( \mathbf{z}^{(v)\mathrm{T}} \mathbf{\Theta}^{(v)} \right)^{\mathrm{T}} \end{aligned} \tag{14}$$

where $*$ is the Hadamard (elementwise) product. It should be noted that the first row $\boldsymbol{\theta}^{(v),0}$ within $\mathbf{\Theta}^{(v)}$ is always associated with $z_0^{(v)} = 1$ and represents the bias factors of the $v$-th view. Through the bias factors, the lower-order interactions are explored in the predictive function.

By replacing the tensor inner product in Eq. (12) using Eq. (14), we then have

$$f_t(\{\mathbf{x}^{(v)}\}) = \mathbf{x}^{\mathrm{T}} \mathbf{u}_t + \boldsymbol{\phi}^t \prod_{v=1}^{V} * \left( \mathbf{z}^{(v)\mathrm{T}} \mathbf{\Theta}^{(v)} \right)^{\mathrm{T}} \tag{15}$$

We name this model as multilinear factorization machines (MFMs). Clearly, the parameters of the interactions between multiple tasks with multiple views are jointly factorized. The joint factorization 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. Further, since the lower-order interactions are modeled with the bias factors, this joint factorization model can easily deal with missing views and even incomplete views for multiple tasks.

Another appealing property of MFMs comes from the main characteristics of multilinear analysis. After factorizing the weight tensor $\mathcal{W}$, there is no need to construct the input tensor physically. From Eq. (15), we can find that the common subspace shared by multiple views can be discovered through the Hadamard product of the low-dimensional projection of each view, and its contribution to the $t$-th task is controlled by the weight vector $\boldsymbol{\phi}^t$. Moreover, the model complexity is linear in the number of original features. In particular, the model complexity is $O(R(V + I + T) + \sum_{t=1}^{T} I_t)$, where $I_t$ is the number of features in the $t$-th task. This multilinear property can help save memory and also speed up the learning procedure.

## 3.3 Learning Multilinear Factorization Machines

Following the regularization formulation in Eq. (3), we propose to estimate the model parameters by minimizing the following regularized empirical risk:

$$\min \mathcal{R}(\mathbf{\Phi}, \{\mathbf{\Theta}^{(v)}\}, \mathbf{U}) = \sum_{t=1}^{T} \mathcal{L}_t(f_t(\{\mathbf{X}_t^{(v)}\}), \mathbf{y}_t) + \lambda\Omega_\lambda(\mathbf{\Phi}, \{\mathbf{\Theta}^{(v)}\}) + \gamma\Omega_\gamma(\mathbf{U}) \quad (16)$$

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_T] \in \mathbb{R}^{I\times T}$. The regularization $\Omega_\lambda$ and $\Omega_\gamma$ can be Forbenius norm, $\ell_{2,1}$ norm, or other structural regularization. To optimize the objective function, we present the alternating block coordinate descent approach in the rest of this section.

With all other parameters fixed, the minimization over $\mathbf{\Theta}^{(v)}$ simply consists of learning the parameters $\mathbf{\Theta}^{(v)}$ by a regularization method, and the partial derivative of $\mathcal{R}$ w.r.t. $\mathbf{\Theta}^{(v)}$ is given by

$$\frac{\partial \mathcal{R}}{\partial \mathbf{\Theta}^{(v)}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial f_t} \frac{\partial f_t}{\partial \mathbf{\Theta}^{(v)}} + \lambda \frac{\partial \Omega_\lambda(\mathbf{\Theta}^{(v)})}{\partial \mathbf{\Theta}^{(v)}} \quad (17)$$

where $\frac{\partial \mathcal{L}_t}{\partial f_t} = \frac{1}{N_t}\left[\begin{array}{c} \frac{\partial \ell_{t,1}}{\partial f_t}, \cdots, \frac{\partial \ell_{t,N_t}}{\partial f_t} \end{array}\right]^{\mathrm{T}} \in \mathbb{R}^{N_t}$.

For convenience, we let $\boldsymbol{\pi} \in \mathbb{R}^R$ denote $\prod_{v=1}^{V} * \left(\mathbf{z}^{(v)\mathrm{T}} \mathbf{\Theta}^{(v)}\right)^{\mathrm{T}}$ and $\boldsymbol{\pi}^{(-v)} \in \mathbb{R}^R$ denote $\prod_{v'=1, v'\neq v}^{V} * \left(\mathbf{z}^{(v')\mathrm{T}} \mathbf{\Theta}^{(v')}\right)^{\mathrm{T}}$. Let $\mathbf{\Pi} = [\boldsymbol{\pi}_1, \cdots, \boldsymbol{\pi}_N]^{\mathrm{T}}$ and $\mathbf{\Pi}^{(-v)} = [\boldsymbol{\pi}_1^{(-v)}, \cdots, \boldsymbol{\pi}_N^{(-v)}]^{\mathrm{T}}$. We then have that

$$\begin{aligned} \frac{\partial \mathcal{L}_t}{\partial f_t} \frac{\partial f_t}{\partial \mathbf{\Theta}^{(v)}} &= \frac{1}{N_t} \sum_{n=1}^{N_t} \frac{\partial \ell_{t,n}}{\partial f_{t,n}} \frac{\partial f_{t,n}}{\partial \mathbf{\Theta}^{(v)}} \\ &= \sum_{n=1}^{N_t} \mathbf{z}_{t,n}^{(v)} \left( \left( \frac{1}{N_t} \frac{\partial \ell_{t,n}}{\partial f_{t,n}} \boldsymbol{\phi}^t \right) * \boldsymbol{\pi}_{t,n}^{(-v)} \right) \\ &= \mathbf{Z}_t^{(v)} \left( \left( \frac{\partial \mathcal{L}_t}{\partial f_t} \boldsymbol{\phi}^t \right) * \mathbf{\Pi}_t^{(-v)} \right) \end{aligned} \quad (18)$$

With all other variables fixed, the minimization over $\mathbf{\Phi}$ w.r.t the empirical loss simply consists of learning the parameters $\boldsymbol{\phi}^t$ independently. The partial derivative of $\mathcal{R}$ w.r.t. $\mathbf{\Phi}$ is given by

$$\frac{\partial \mathcal{R}}{\partial \mathbf{\Phi}} = \left[\begin{array}{c} \frac{\partial \mathcal{L}_1}{\partial f_1} \frac{\partial f_1}{\partial \boldsymbol{\phi}^1} \; ; \; \cdots ; \; \frac{\partial \mathcal{L}_T}{\partial f_T} \frac{\partial f_T}{\partial \boldsymbol{\phi}^T} \end{array}\right] + \lambda \frac{\partial \Omega_\lambda(\mathbf{\Phi})}{\partial \mathbf{\Phi}} \quad (19)$$

Following the derivation in Eq. (18), we have that

$$\frac{\partial \mathcal{L}_t}{\partial f_t} \frac{\partial f_t}{\partial \boldsymbol{\phi}^t} = \left(\frac{\partial \mathcal{L}_t}{\partial f_t}\right)^{\mathrm{T}} \mathbf{\Pi}_t \quad (20)$$

With all other variables fixed, the partial derivative of $\mathcal{R}$ w.r.t. $\mathbf{U}$ is given by

$$\frac{\partial \mathcal{R}}{\partial \mathbf{U}} = \left[\begin{array}{c} \mathbf{X}_1 \frac{\partial \mathcal{L}_1}{\partial f_1} \; , \; \cdots , \; \mathbf{X}_T \frac{\partial \mathcal{L}_T}{\partial f_T} \end{array}\right] + \gamma \frac{\partial \Omega_\gamma(\mathbf{U})}{\partial \mathbf{U}} \quad (21)$$

where $\mathbf{X}_t = [\mathbf{X}_t^{(1)}; \ldots ; \mathbf{X}_t^{(V)}] \in \mathbb{R}^{I\times N_t}$ is the concatenated feature matrix for the $t$-th task.

The optimization procedure is summarized in Algorithm 1. In the experiment, we apply AdaGrad [10], an adaptive gradient-based optimization approach that automatically determines a per-parameter learning rate, for parameter updates. The speed bottleneck of this algorithm is in computing the predicted values of all the training instances. The time complexity of computing Eq. (15) for an instance is $O(RV + (R+1)I_t))$, while the computation can be done in parallel. It can be further reduced under sparsity, where most of the elements in $\mathbf{x}_{t,n}$ are 0 and the sums have only to be computed over the non-zero elements. Let $N_z(\mathbf{x}_{t,n})$ denote the number of non-zero elements, then the time complexity of the computation for each instance is $O(RV + (R+1)N_z(\mathbf{x}_{t,n}))$.

**Algorithm 1** Learning Multilinear Factorization Machines

```
Input: Training data $\mathcal{D}$, number of factors $R$, regularization parameter $\lambda, \gamma$ and standard deviation $\sigma$
Output: Model parameters $\{\mathbf{\Theta}^{(v)}\}, \mathbf{\Phi}, \mathbf{U}$
1: Initialize $\{\mathbf{\Theta}^{(v)}\}, \mathbf{\Phi}, \mathbf{U} \sim \mathcal{N}(0, \sigma)$
2: repeat
3:     for $v := 1$ to $V$ do
4:         Fixing $\{\mathbf{\Theta}^{(v')}\}_{v'\neq v}$, $\mathbf{\Phi}$ and $\mathbf{U}$, update $\mathbf{\Theta}^{(v)}$
5:     end for
6:     Fixing $\{\mathbf{\Theta}^{(v)}\}$ and $\mathbf{U}$, update $\mathbf{\Phi}$
7:     Fixing $\{\mathbf{\Theta}^{(v)}\}$ and $\mathbf{\Phi}$, update $\mathbf{U}$
8: until convergence
```

## 4. EXPERIMENTS

### 4.1 Datasets

To evaluate the performance of the proposed MFMs, we conduct extensive experiments on the following four datasets (two of them are for classification tasks, and two of them are for regression tasks):

**FOX**[2]: This dataset was crawled from FOX web news [21]. The category for each news article can be viewed as the class label. For each task (category), the documents from one category are regarded as positive samples. Each instance can be represented in two views, the text view and image view. The text view consists of $\ell_2$-normalized TF-IDF vector representation extracted from titles, abstracts, and text body contents[3]. The image view consists of seven groups of color features and five textural features.

**DBLP**[4]: This dataset was extracted from the DBLP [17]. The research areas (DB, DM, AI, IR, CV, and ML) that an author has published papers in can be viewed as class labels. For each task (research area), the authors who have published papers in that research area are regarded as positive samples. Each instance is represented in two views, the text view and the linkage view. The $\ell_2$-normalized TF-IDF vector representation of all the paper titles published by the author is used as the text view data. Since each instance represents an author, the linkage feature of an instance is the binary vector of the co-authors' ids.

**MovieLens**[5]: Regression tasks for rating prediction are studied on this MovieLens dataset. We select the top 10 genres with the most movies as tasks. Each rating in this dataset has three views, *i.e.*, users, movies and tags. The user view consists of binary feature vectors for user ids, and thus for each rating there is only one non-zero feature in the user view, *i.e.*, the associated user id; the same for the movie view. The tags of the movies are used for the tag view.

**Amazon**[6]: This dataset contains item reviews from Amazon [19]. We select ratings from the Clothing, Jewelry, Nov-

[2]https://sites.google.com/site/qianmingjie/home/datasets/
[3]Stemming, lemmatization, removing stop-words and words with frequency less than 1%, etc., are handled beforehand for all the text view mentioned in this paper
[4]http://dblp.uni-trier.de/db/
[5]http://grouplens.org/datasets/movielens/
[6]http://jmcauley.ucsd.edu/data/amazon/

**Table 2: The statistics for each dataset.**

| Classification | #Feature | $T$ | $N_p$ | $N_n$ | Partial View Missing? |
|---|---|---|---|---|---|
| FOX | image(996), text(2,711) | 4 | 178∼635 | 888∼1,345 | No |
| DBLP | linkage(4,638), text(687) | 6 | 635∼1,950 | 2,688∼3,985 | No |
| Regression | #Feature | $T$ | $N$ | Density | Partial View Missing? |
| MovieLens | users(943), movies(1,599), tags(1,065) | 10 | 758∼39,895 | 6.3% | No |
| Amazon | users(1,805,364), items(192,978), text(83,143) | 5 | 349,038∼1,015,189 | 0.001% | Yes |

**Table 3: Performance comparison on the FOX dataset. The best two results are listed in bold.**

| Training Ratio | Measure | rMTFL | FM | TF | IteM$^2$ | CSL-MTMV | MFM-T | MFM-F | MFM-F-S |
|---|---|---|---|---|---|---|---|---|---|
| 10% | ACC | 0.8816±0.011 | 0.7883±0.011 | 0.8460±0.035 | 0.4052±0.076 | 0.8986±0.011 | 0.9259±0.019 | **0.9343±0.012** | **0.9364±0.011** |
| | F1 | 0.6911±0.035 | 0.2930±0.046 | 0.6362±0.044 | 0.3598±0.030 | 0.7335±0.029 | 0.7799±0.053 | **0.8076±0.038** | **0.8119±0.027** |
| | AUC | 0.9109±0.013 | 0.7764±0.018 | 0.8681±0.038 | 0.5326±0.036 | 0.9342±0.011 | 0.9678±0.015 | **0.9763±0.008** | **0.9777±0.009** |
| 20% | ACC | 0.9039±0.013 | 0.8087±0.011 | 0.8546±0.025 | 0.5091±0.078 | 0.9264±0.005 | 0.9551±0.005 | **0.9569±0.010** | **0.9612±0.005** |
| | F1 | 0.7654±0.026 | 0.3764±0.050 | 0.6632±0.051 | 0.3306±0.068 | 0.8004±0.012 | 0.8721±0.012 | **0.8769±0.027** | **0.8882±0.014** |
| | AUC | 0.9353±0.016 | 0.8260±0.012 | 0.8751±0.029 | 0.4954±0.043 | 0.9705±0.003 | 0.9883±0.003 | **0.9885±0.006** | **0.9922±0.002** |
| 30% | ACC | 0.9314±0.005 | 0.8255±0.007 | 0.8767±0.082 | 0.4289±0.134 | 0.9390±0.004 | 0.9641±0.007 | **0.9709±0.003** | **0.9697±0.004** |
| | F1 | 0.8051±0.015 | 0.4448±0.026 | 0.7302±0.132 | 0.3314±0.056 | 0.8341±0.012 | 0.9000±0.018 | **0.9185±0.010** | **0.9149±0.010** |
| | AUC | 0.9709±0.005 | 0.8393±0.012 | 0.9010±0.091 | 0.5365±0.039 | 0.9812±0.003 | 0.9916±0.003 | **0.9949±0.001** | **0.9949±0.001** |

elty, Accessories and Shoes item categories for the study of large-scale regression tasks, where each category is viewed as a task. Users and items with less than 5 ratings are filtered. Each rating in this dataset has three views, *i.e.*, users, items and text. The user view and item view are constructed using the same way as in the MovieLens dataset. The $\ell_2$-normalized TF-IDF vector representation of all summaries of the item is used as the text view.

The statistics for each dataset is summarized in Table 2, where $T$ denotes the number of tasks, $N_p$, $N_n$ and $N$ denote the number of positive, negative and all the samples in each task, respectively. The density in Table 2 means the density of the user-item matrix in the dataset. Note that the text view is partially missing in the Amazon dataset, where 0.62% reviews have no text.

## 4.2 Comparisons

We compare the proposed MFM method with five state-of-the-art methods. **Factorization Machine (FM)** explores pairwise interactions between all features. We apply the FM in the setting of MTMV learning by concatenating the task indicator and all the feature vectors from multiple views as the input feature vector. The preliminary study shows it performs better than training each task separately. **Robust Multi-Task Feature Learning (rMTFL)** is a representative MTL algorithm that uses composite regularization for joint feature learning [13]. **Tensor Factorization (TF)** is a generalization of matrix factorization to higher orders. Since TF only considers the highest-order of the given tensor, we use Eq. (8) to model the MTMV data for TF and factorize the weight tensor. **IteM$^2$** is a transductive MTMV learning algorithm with applications to classification problems [14]. Since IteM$^2$ can only handle nonnegative feature values, we add a positive constant to the feature values to guarantee its nonnegativity. **CSL-MTMV** is an inductive MTMV learning algorithm [15] that assumes the predictions of different views within a single task are consistent.

**Multilinear Factorization Machine (MFM)** is the proposed model that learns the predictive multilinear structure. We compare three variations of MFM for studying the effects of each component in the model. Forbenius norm regularizers are used as the default for all the parameters to avoid overfitting, if not specified. **MFM-F** and **MFM-F-S** both denote the variations that use the proposed predictive model in Eq. (15), while **MFM-F-S** uses $\ell_{2,1}$ norm regularization on $\mathbf{U}$ for joint feature selection. **MFM-T** denotes the variation that uses only the tensor inner product as the predictive function, *i.e.*, the task-specific weight vector $\mathbf{u}_t$ is always set to be zeros. For all three MFM methods, we use squared loss for regression tasks and logistic loss for classification tasks.

## 4.3 Model Construction and Evaluation

For each dataset we randomly select $n$%, 10%, and 40% of labeled samples for each task as training set, validation set, and testing set, respectively, where $n$ is varying in the range $[10, 30]$ with the increment of 10. Validation sets are used for hyperparameter 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 our experiment. For all the methods, the dimension of latent factors $R = 20$, the learning rate $\eta = 0.1$, the initialization $\sigma = 1$, the maximum number of iterations are all set as 200. We apply grid searching to identify optimal values for each regularization hyperparameter from $\{10^{-5}, 10^{-4}, \cdots, 10^{5}\}$ for all the comparison methods.

To investigate the performance of comparison methods, we adopt accuracy (ACC), F1-score, and the area under the receiver-operator characteristic curve (AUC) on the test data as the evaluation metrics [4, 14, 15]. Overall accuracy, F1-score and AUC are averaged over all tasks. The larger value of each metric indicates the better performance. For regression tasks, we adopt the mean absolute error (MAE) and root mean squared error (RMSE) on the test data as the evaluation metrics [23]. Overall MAE and RMSE are the averaged over all tasks. The smaller value of each 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 machines with Intel Xeon 6-Core CPUs of 2.4 GHz and 64 GB RAM.

## 4.4 Classification Tasks

Table 3 and Table 4 show the performance of all the com-

**Table 4: Performance comparison on the DBLP dataset. The best two results are listed in bold.**

| Training Ratio | Measure | rMTFL | FM | TF | IteM$^2$ | CSL-MTMV | MFM-T | MFM-F | MFM-F-S |
|---|---|---|---|---|---|---|---|---|---|
| 10% | ACC | 0.8057±0.004 | 0.7264±0.004 | 0.7471±0.011 | 0.6223±0.004 | 0.7290±0.005 | 0.8008±0.004 | **0.8058±0.004** | **0.8062±0.005** |
| | F1 | 0.5395±0.015 | 0.0732±0.019 | 0.5606±0.011 | 0.3176±0.007 | 0.4402±0.004 | 0.5278±0.018 | **0.5469±0.014** | **0.5471±0.015** |
| | AUC | 0.7888±0.007 | 0.6264±0.023 | 0.7723±0.009 | 0.5310±0.007 | 0.6890±0.006 | 0.8039±0.010 | **0.8113±0.010** | **0.8120±0.009** |
| 20% | ACC | 0.8319±0.004 | 0.7628±0.007 | 0.7878±0.007 | 0.6309±0.003 | 0.7760±0.002 | 0.8346±0.004 | **0.8374±0.004** | **0.8371±0.004** |
| | F1 | 0.6447±0.008 | 0.2680±0.038 | 0.6247±0.014 | 0.3494±0.006 | 0.5295±0.007 | 0.6274±0.013 | **0.6499±0.012** | **0.6508±0.012** |
| | AUC | 0.8374±0.005 | 0.7548±0.022 | 0.8200±0.010 | 0.5550±0.006 | 0.7655±0.005 | 0.8531±0.006 | **0.8658±0.005** | **0.8632±0.005** |
| 30% | ACC | 0.8412±0.004 | 0.7978±0.005 | 0.8191±0.008 | 0.6256±0.003 | 0.8037±0.003 | 0.8501±0.004 | **0.8527±0.004** | **0.8535±0.004** |
| | F1 | 0.6796±0.010 | 0.4312±0.021 | 0.6670±0.021 | 0.3569±0.009 | 0.5869±0.007 | 0.6800±0.013 | **0.6891±0.012** | **0.6892±0.009** |
| | AUC | 0.8590±0.005 | 0.8351±0.010 | 0.8498±0.009 | 0.5563±0.006 | 0.8083±0.006 | 0.8757±0.005 | **0.8866±0.006** | **0.8866±0.006** |

**Table 5: Performance comparison on the MovieLens dataset. The best two results are listed in bold.**

| Training Ratio | Measure | rMTFL | FM | TF | CSL-MTMV | MFM-T | MFM-F | MFM-F-S |
|---|---|---|---|---|---|---|---|---|
| 10% | RMSE | 1.1861±0.008 | 1.0251±0.003 | 1.5679±0.099 | 1.05013±0.005 | 1.0078±0.005 | **1.0069±0.005** | **0.9976±0.004** |
| | MAE | 0.8516±0.004 | 0.8422±0.004 | 1.2497±0.088 | 0.8516±0.004 | 0.8142±0.005 | **0.8082±0.005** | **0.8022±0.004** |
| 20% | RMSE | 1.0631±0.005 | 0.9898±0.003 | 1.2519±0.069 | 1.0214±0.004 | **0.9877±0.003** | 0.9977±0.003 | **0.9857±0.003** |
| | MAE | 0.8539±0.005 | 0.7997±0.004 | 0.9801±0.053 | 0.8294±0.004 | **0.7987±0.003** | 0.8023±0.003 | **0.7927±0.004** |
| 30% | RMSE | 0.9917±0.003 | **0.9765±0.003** | 1.2066±0.061 | 1.0082±0.003 | 0.9795±0.003 | 0.9887±0.004 | **0.9785±0.003** |
| | MAE | 0.8159±0.003 | **0.7815±0.003** | 0.9380±0.045 | 0.8189±0.003 | 0.7885±0.002 | 0.7823±0.004 | **0.7789±0.004** |

parison methods on the FOX and DBLP datasets. From these results, we have the several observations. First, most of the methods achieve better performance when the training size increases, and the proposed MFM methods consistently outperform all the other methods on both datasets. This is mainly because MFM can effectively learn the predictive multilinear structure from the full-order interactions of MTMV data. Moreover, by combining the task-specific feature map with the task-view shared multilinear feature map, MFM-F and MFM-F-S can further improve the performance. Further, it can be found that MFM-F-S almost always perform better than MFM-F, which empirically shows the effectiveness of learning features with sparse constraints.

Besides, among all the factorization based methods (*i.e.*, FM, TF and MFM), FM performs the worst. Such poor performance indicates that FM cannot discriminate important features (*e.g.*, the task indicators) for different tasks. Because all the pairwise interactions between all the features are considered in FM, the interactions between the task indicators and important features are buried by many redundant intra-view feature interactions. The ability to distinguish different tasks is critical for multi-task classification problems, since the labels of a classification task are usually dissimilar or even opposite to other classification tasks. On the other hand, by fusing multi-view information into tensor structures, the tensor-based methods can achieve relatively better performance. In addition, we find that TF performs much worse than the MFM-T. This confirms that learning only from the highest-order interactions is limited and incorporating the lower-order interactions in the predictive models can help provide more information [4, 22].

In addition, the rMTFL method, which learns important features for all the tasks but does not distinguish features from different views, can achieve comparative or even better performance than the state-of-the-art MTMV learning methods, *i.e.*, CSL-MTMV. This is mainly because CSL-MTMV enforces the prediction results of each view to be consistent with each other, while the text view and the image view (or the linkage view) are not similar. In contrast, the proposed MFM methods can achieve better performance by exploring the complementary information of multiple views.

## 4.5 Regression Tasks

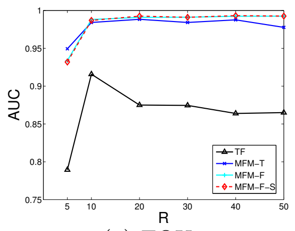
Table 5 and Table 6 report the performance comparison on the MovieLens and Amazon datasets. Note that IteM$^2$ is not compared since it can only work for classification tasks. Besides, due to the high memory complexity CSL-MTMV and rMTFL cannot be applied to the large-scale Amazon dataset. From these results, we can observe that the proposed MFM methods outperform most of the comparison methods on both datasets, especially when the training data is limited. This is because when less instances are available, some users and items may hardly appear during training, making it harder to learn the model parameters for the user view and the item view. By incorporating the bias terms for each view in the full-order tensor, the proposed MFM models can explore the information from other complementary views (*e.g.*, the tag view or the text view) to alleviate the issue and to improve the performance. We also notice that FM achieves much more competitive performance in regression tasks than in classification tasks. This is due to the fact that the regression tasks in the experiments are more similar and thus it is less important to learn task-specific features. Furthermore, we observe that the performance of TF on the Amazon dataset is unacceptably low, which is due to the fact that some instances do not have any text features. Because TF can only learn from the highest-order interactions, its performance is significantly impacted by the existence of the partially missing view. In contrast, by taking the full-order interactions into consideration, MFM methods can easily deal with the partially missing view.

## 4.6 Hyperparameter Analysis

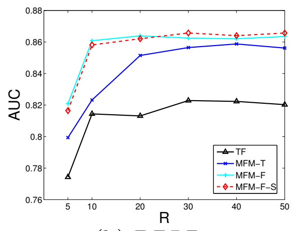
The number of latent factors $R$ is an important hyperparameter for the proposed MFM methods involving the CP factorization. We analyze different values of $R$ and report the results in Fig. 2. Due to space limit, we only report AUC for classification tasks and RMSE for regression tasks using 20% of instances as training set. Compared with the other tensor-based method (TF), the performance of MFM methods is much more stable when $R \geq 20$ for all four datasets. It validates the importance of including lower-order interac-

Table 6: **Performance comparison on the Amazon dataset. The best two results are listed in bold. Due to the memory overhead, rMTFL and CSL-MTMV are not compared.**

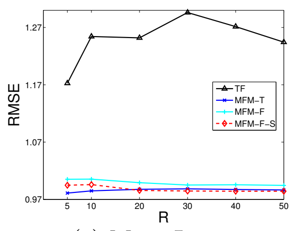
| Training Ratio | Measure | FM | TF | MFM-T | MFM-F | MFM-F-S |
|---|---|---|---|---|---|---|
| 10% | RMSE | 0.9834±0.001 | 3.6044±0.003 | **0.9775±0.001** | 0.9857±0.001 | **0.9825±0.002** |
| | MAE | 0.7420±0.001 | 3.4574±0.005 | 0.7249±0.001 | **0.7158±0.002** | **0.7129±0.001** |
| 20% | RMSE | 0.9814±0.001 | 3.5611±0.018 | **0.9764±0.001** | 0.9845±0.001 | **0.9775±0.001** |
| | MAE | 0.7343±0.002 | 3.3965±0.030 | 0.7255±0.001 | **0.7112±0.001** | **0.7086±0.001** |
| 30% | RMSE | 0.9782±0.002 | 3.4962±0.018 | **0.9705±0.002** | 0.9841±0.001 | **0.9733±0.001** |
| | MAE | 0.7257±0.002 | 3.2945±0.034 | **0.7001±0.001** | 0.7115±0.001 | **0.7078±0.001** |



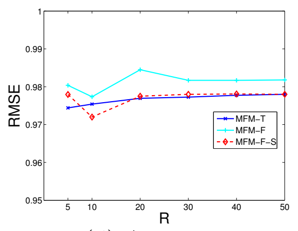
(a) FOX



(b) DBLP

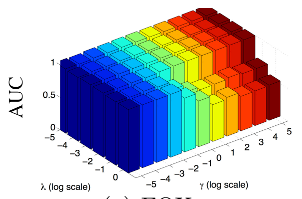


(c) MovieLens

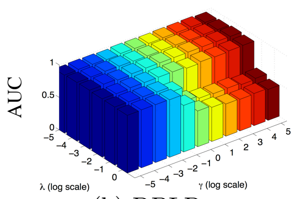


(d) Amazon

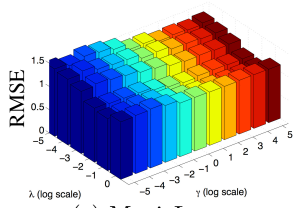
**Figure 2: Sensitivity analysis of number of latent factor.**



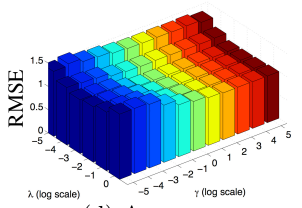
(a) FOX



(b) DBLP



(c) MovieLens



(d) Amazon

**Figure 3: Sensitivity analysis of regularization hyperparameters.**

tions in the predictive model.

To explore the effects of the regularization hyperparameters on the performance, we run with different values for $\lambda$ and $\gamma$. Since there are no much differences between the results of MFM-F-S and those of MFM-F, we only report the results using MFM-F-S. From Fig. 3(a) and Fig. 3(b), we can clearly see that the performance of classification is fairly stable for most cases. However, when $\lambda$ and $\gamma$ are both large, the model parameters can be very small, which decreases the value of AUC. From Fig. 3(c) and Fig. 3(d), we can observe that for regression tasks, the RMSE is much lower when the value of $\gamma$ is large and the value of $\lambda$ is in the range from $10^{-3}$ to $10^{-1}$. This could indicate that the task-specific weight vector $\mathbf{u}_t$ is less important for the prediction. This confirms our observation that the tasks in the regression problem are much more similar than in the classification problem.

## 5. RELATED WORK

**Multi-Task Learning:** MTL aims to improve the performance of related tasks by learning a model which is able to capture intrinsic relatedness across tasks [5]. The current work mainly focuses on joint feature selection and feature learning, where the task relatedness is explicitly expressed as a shared part of the tasks [1, 2, 11, 12, 20]. Most of these methods make the assumption that different tasks share a common representation/structure. However, in practical applications it is too restrictive to constrain all tasks to share a common set of features, due to the heterogeneity of tasks. Some recent methods proposed to capture different types of relationships using a composite regularization [8, 13]. In particular, the alternating structure optimization (ASO) algorithm [1] and its convex version cASO algorithm [7] decompose the predictive model into the task-specific and task-shared feature mapping, which can be cast as special cases of our framework when $V = 1$.

**Multi-View Learning:** MVL concerns about exploiting different views on the same object to make a more accurate learning. In essence, MVL utilizes experiential inputs to explore diverse representations so that an increasingly variety of problems may be recognized, formulated and solved [6]. In this fashion, the strengths of each view are amplified and the weaknesses are alleviated. There are currently a plethora of studies available for MVL. Interested readers are referred to [27] for a comprehensive survey of these techniques and applications. The most related work to ours is that of [3, 4] who introduced and explored the tensor product operator to integrate different views together in a joint tensor. The advantage of tensor representation is that it enables not only to record the information from multiple views, but also strengthen and capture the relationships between them [18]. In addition, various tensor factorizations [16] can be modelled to learn dependencies between variables, each of which imply different hypotheses about the data. In this study, we employ the CP factorization to facilitate the learning process, but it can be easily extended also to other types of factorizations.

**Multi-Task Multi-View Learning:** The IteM$^2$ algorithm [14] was first proposed by He *et al.* for MTMV learning. Since IteM$^2$ is a transductive learning method, it is unable to generate predictive models on independent, unknown testing samples. Besides, it can only deal with classification problems with nonnegative feature values. Assuming the predictive models should be consistent among different views, co-regularization based methods were later developed [15, 28]. Specifically, Zhang *et al.* proposed regMVMT algorithm [28] that minimizes the difference of the predictive models for different tasks on the same view; Jin *et al.* proposed a more generalized algorithm, CSL-MTMV [15], which assumes a low-dimensional subspace is shared among multiple related tasks that have common views. These methods assume that all the views are similar to each other, while such assumption may not be appropriate especially for heterogeneous data. It also makes these methods have difficulty in learning from partially observed views.

## 6. CONCLUSIONS

In this paper, we present efficient multilinear factorization machines (MFMs) for MTMV problems. The proposed MFMs can learn the task-specific features and the common latent spaces embedded within the multimodal interactions among the multiple tasks and the multiple views. Because full-order interactions are collectively used during learning procedure, MFMs can deal with the partially incomplete data without difficulty. Moreover, the complexity of MFMs is linear in the number of features, which make MFMs suitable to large-scale real-world problems. Extensive experiments on four real-world datasets demonstrate that our proposed MFMs outperform several state-of-the-art methods in a wide variety of MTMV learning problems.

## Acknowledgments

This work is supported in part by NSF through grants IIS-1526499, CNS-1626432, NSFC (61472089, 61503253, 61672357), NSFC-Guangdong Joint Found (U1501254), and the Science Foundation of Guangdong Province (2014A030313556). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

## 7. REFERENCES

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, 2005.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[3] B. Cao, L. He, X. Kong, P. S. Yu, Z. Hao, and A. B. Ragin. Tensor-based multi-view feature selection with applications to brain diseases. In *ICDM*, pages 40–49, 2014.

[4] B. Cao, H. Zhou, G. Li, and P. S. Yu. Multi-view machines. In *WSDM*, 2016.

[5] R. Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[6] M. Ceci, A. Appice, H. L. Viktor, D. Malerba, E. Paquet, and H. Guo. Transductive relational classification in the co-training paradigm. In *Workshop on MLDM*, pages 11–25. Springer, 2012.

[7] J. Chen, L. Tang, J. Liu, and J. Ye. A convex formulation for learning shared structures from multiple tasks. In *ICML*, pages 137–144, 2009.

[8] J. Chen, J. Zhou, and J. Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *SIGKDD*, pages 42–50, 2011.

[9] C. Cortes, M. Mohri, and A. Rostamizadeh. Learning non-linear combinations of kernels. In *NIPS*, pages 396–404, 2009.

[10] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[11] A. Evgeniou and M. Pontil. Multi-task feature learning. *NIPS*, 19:41, 2007.

[12] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *SIGKDD*, pages 109–117, 2004.

[13] P. Gong, J. Ye, and C. Zhang. Robust multi-task feature learning. In *SIGKDD*, pages 895–903, 2012.

[14] J. He and R. Lawrence. A graph-based framework for multi-task multi-view learning. In *ICML*, pages 25–32, 2011.

[15] X. Jin, F. Zhuang, S. Wang, Q. He, and Z. Shi. Shared structure learning for multiple tasks with multiple views. In *ECML/PKDD*, pages 353–368, 2013.

[16] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[17] X. Kong, X. Shi, and S. Y. Philip. Multi-label collective classification. In *SDM*, volume 11, pages 618–629. SIAM, 2011.

[18] Q. Li, J. Wang, F. Wang, P. Li, L. Liu, and Y. Chen. The role of social sentiment in stock markets: a view from joint effects of multiple information sources. *Multimedia Tools and Applications*, pages 1–31, 2016.

[19] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *SIGKDD*, pages 785–794, 2015.

[20] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252, 2010.

[21] M. Qian and C. Zhai. Unsupervised feature selection for multi-view clustering on text-image web news data. In *CIKM*, pages 1963–1966, 2014.

[22] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.

[23] S. Rendle. Factorization machines with libFM. *Intelligent Systems and Technology*, 3(3):57, 2012.

[24] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90, 2010.

[25] V. Sindhwani and D. S. Rosenberg. An rkhs for multi-view learning and manifold co-regularization. In *ICML*, pages 976–983, 2008.

[26] M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML*, pages 1065–1072, 2009.

[27] C. Xu, D. Tao, and C. Xu. A survey on multi-view learning. *arXiv:1304.5634*, 2013.

[28] J. Zhang and J. Huan. Inductive multi-task learning with multiple view data. In *SIGKDD*, pages 543–551, 2012.