Multi-view Machines

Bokai Cao[†,*], Hucheng Zhou[‡], Guoqiang Li and Philip S. Yu[†]
[†]Department of Computer Science, University of Illinois at Chicago, IL, USA
[‡]Microsoft Research, Beijing, China
caobokai@uic.edu, huzho@microsoft.com, psyu@cs.uic.edu

ABSTRACT

With rapidly growing amount of data available on the web, it becomes increasingly likely to obtain data from different perspectives for multi-view learning. Some successive examples of web applications include recommendation and target advertising. Specifically, to predict whether a user will click an ad in a query context, there are available features extracted from user profile, ad information and query description, and each of them can only capture part of the task signals from a particular aspect/view. Different views provide complementary information to learn a practical model for these applications. Therefore, an effective integration of the multi-view information is critical to facilitate the learning performance.

In this paper, we propose a general predictor, named multi-view machines (MVMs), that can effectively explore the full-order interactions between features from multiple views. A joint factorization is applied for the interaction parameters which makes parameter estimation more accurate under sparsity and renders the model with the capacity to avoid overfitting. Moreover, MVMs can work in conjunction with different loss functions for a variety of machine learning tasks. The advantages of MVMs are illustrated through comparison with other methods for multi-view prediction, including support vector machines (SVMs), support tensor machines (STMs) and factorization machines (FMs).

A stochastic gradient descent method and a distributed implementation on Spark are presented to learn the MVM model. Through empirical studies on two real-world web application datasets, we demonstrate the effectiveness of MVMs on modeling feature interactions in multi-view data. A 3.51% accuracy improvement is shown on MVMs over FMs for the problem of movie rating prediction, and 0.57% for ad click prediction.

[*]This work was done while the author was an intern at Microsoft Research.

Keywords

Multi-view Learning, Feature Interaction, Factorization

1. INTRODUCTION

Web data is available not only in great volume but also in multiple representations/views from a variety of sources or feature subsets. Generally, different views provide complementary information to learn an effective model for web-scale applications. Thus, multi-view learning can facilitate the learning performance and is prevalent in a wide range of application domains. For example, for the business on the web, it is critical to estimate the probability that the display of an ad to a specific user when s/he searches for a query will lead to a click. This process involves three entities: users, ads, and queries. An effective integration of features describing these different entities is directly related to precise targeting of an advertising system.

One of the key challenges of multi-view learning is to model the interactions/correlations between different views, wherein complementary information is contained. Conventionally, multi-kernel learning algorithms combine kernels associated with respective views to improve the learning performance [9]. Basically, coefficients are learned based on the usefulness/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.

In contrast to modeling on views, another direction for modeling multi-view data is to directly consider the abundant correlations between features from different views. Feature interactions with different orders can reflect different but complementary insights. Assume that we have obtained a latent factor representing wealth/price-related attributes for each entity (*i.e.*, users, ads and queries) in the advertising system of a search engine, as illustrated in Table 1. For example, users who have high purchase power (*i.e.*, a positive latent factor, $a_{user} > 0$) may have interests in luxury products ($a_{ad} > 0$). However, a thoughtful recommender system should not always recommend luxury products to these users regardless of the query context. In Table 1, it is unreasonable to recommend a luxury bag ($a_{ad} > 0$) to the user when s/he searches for a disease ($a_{query} < 0$), in which case some relevant medicines or medical books ($a_{ad} < 0$) would seem better choices. We can observe that, in such scenarios, only the third-order interactions contribute positively to the recommendation of medicines and negatively to that of luxury bags, while the first-order and the second-order interactions insist to recommend something inappropriate in the specific

Table 1: An example showing the discrepancy between feature interactions with different orders. $\#1 = user + ad + query$, $\#2 = user \times ad + user \times query + ad \times query$, $\#3 = user \times ad \times query$.

User	Ad	Query	#1	#2	#3
1.20 (+)	1.80 (+)	0.50 (+)	3.50 (+)	3.66 (+)	1.08 (+)
1.20 (+)	1.80 (+)	-0.50 (-)	2.50 (+)	0.66 (+)	-1.08 (-)
1.20 (+)	-1.80 (-)	-0.50 (-)	-1.10 (-)	-1.86 (-)	1.08 (+)

context. Note here that we do not claim the higher order interactions can work as the best indicator by their own in all problems. Nevertheless, integrating their contributions into the decision function in an efficient manner is critical.

S. Rendle pioneers the concept of factorization machines (FMs) [10] which are now the state-of-the-art approach to model feature interactions and inspire this work. However, the practical implementations of FMs are usually limited to the second-order interactions, *i.e.*, pairwise correlations. This is partially due to the fact that a separate set of latent factors (parameters to be learned) is introduced for each order of interactions in FMs. That is to say, a feature has a latent representation when it is considered for the second-order interactions, while the same feature has a different representation for the third-order interactions. Moreover, the global bias and the first-order interaction terms in FMs are not factorized and independent from the latent factors for higher order interactions. These bias terms and latent factors for different orders altogether compose inconsistent representations of input features and thus compromise the model interpretability. In addition, independent factorization of interactions with different orders results in a large set of model parameters to be learned which makes the training process challenging.

The major challenge of including higher order interactions is that observations with such interactions become sparser with higher orders. Therefore, parameters representing higher order interactions can hardly be learned from their limited observations, especially from the extremely sparse data, *e.g.*, recommender systems. We suggest a common latent subspace for all features that is shared by different orders of interactions. In this manner, the full-order interactions observed in the data can collectively be used to learn a consistent representation in the latent feature space.

In this paper, we propose a novel model for multi-view prediction, called multi-view machines (MVMs). The main advantages of MVMs are outlined as follows:

- MVMs include the global bias and the full-order interactions between features from multiple views, ranging from the first-order interactions (*i.e.*, contributions of single features) to the highest-order interactions (*i.e.*, contributions of combinations of features from each view).
- MVMs jointly factorize the interaction parameters for different orders to allow accurate parameter estimation under sparsity and avoid overfitting via the effect of bias factors.
- MVMs are a general predictor that can work with different loss functions (*e.g.*, square error, hinge loss, logit loss) for a variety of machine learning tasks.

Table 2: Symbols.

Symbol	Definition and description
s	each lowercase letter represents a scale
$\mathbf{v}$	each boldface lowercase letter represents a vector
$\mathbf{M}$	each boldface capital letter represents a matrix
$\mathcal{T}$	each calligraphic letter represents a tensor, set or space
$\langle \cdot, \cdot \rangle$	denotes inner product
$\circ$	denotes tensor product or outer product
$\times_k$	denotes mode-k product
$\lvert \cdot \rvert$	denotes absolute value
$\lVert \cdot \rVert_F$	denotes (Frobenius) norm of vector, matrix or tensor

To empirically analyze and understand these advantages, we have the MVM model and other baselines implemented in a distributed environment, GraphX [4], which is a component of Spark [17]. Extensive experiments are conducted on real-world web application datasets, for regression and classification tasks, respectively. A 3.51% accuracy improvement is shown on MVMs over FMs for the problem of movie rating prediction, and 0.57% for ad click prediction.

2. BACKGROUND

In this section, we first state the problem of multi-view prediction and briefly review the adaptation of existing methods for multi-view prediction, including support vector machines (SVMs), support tensor machines (STMs) and factorization machines (FMs).

2.1 Multi-view Prediction

Suppose each instance has representations in m different views, *i.e.*, $\mathbf{x}^T = \left(\mathbf{x}^{(1)^T}, ..., \mathbf{x}^{(m)^T}\right)$, where $\mathbf{x}^{(p)} \in \mathbb{R}^{I_p}$, I_p is the dimensionality of the p-th view. Let $d = \sum_{p=1}^{m} I_p$, so $\mathbf{x} \in \mathbb{R}^d$. Considering the problem of click through rate (CTR) prediction for advertising display, for example, an instance corresponds to an *impression* which involves a user, an ad, and a query. Therefore, suppose $\mathbf{x}^T = \left(\mathbf{x}^{(1)^T}, \mathbf{x}^{(2)^T}, \mathbf{x}^{(3)^T}\right)$ is an impression, $\mathbf{x}^{(1)}$ contains information of the user profile, $\mathbf{x}^{(2)}$ is associated with the ad information, and $\mathbf{x}^{(3)}$ is the description from the query aspect. The result of an impression is *click* or *non-click*.

Given a training set with n labeled instances represented from m views: $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, ..., n\}$, in which $\mathbf{x}_i^T = \left(\mathbf{x}_i^{(1)^T}, ..., \mathbf{x}_i^{(m)^T}\right)$ and $y_i \in \{-1, 1\}$ is the class label of the i-th instance. For CTR prediction problem, $y = 1$ denotes *click* and $y = -1$ denotes *non-click* in an impression. The task of multi-view classification is to learn a function $f : \mathbb{R}^{I_1} \times \cdots \times \mathbb{R}^{I_m} \to \{-1, 1\}$ that correctly predicts the label of a test instance. Alternatively, if $y_i \in \mathbb{R}$, it is a multi-view regression problem, *e.g.*, rating prediction.

Table 2 lists some basic symbols that will be used throughout the paper. In addition, we introduce the concept of tensors which are higher order arrays that generalize the notions of vectors (the first-order tensors) and matrices (the second-order tensors), whose elements are indexed by more than two indexes. Definitions of tensor product and mode-k product are given which will be used to formulate our proposed model.

Definition 2.1 (Tensor Product or Outer Product). *The tensor product* $\mathcal{X} \circ \mathcal{Y}$ *of a tensor* $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_m}$ *and*

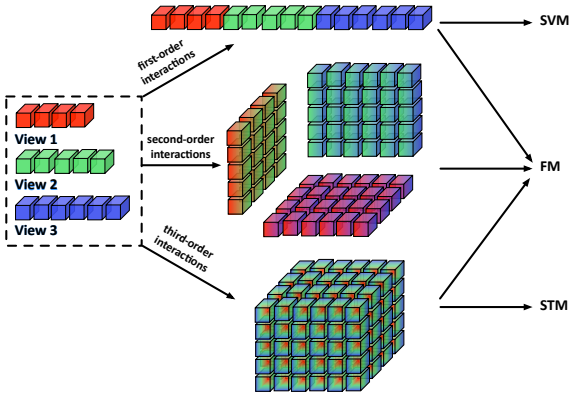


Figure 1: Related work (and variations) on modeling feature interactions in multi-view data.

another tensor $\mathcal{Y} \in \mathbb{R}^{I'_1 \times \cdots \times I'_{m'}}$ *is defined by*

$$(\mathcal{X} \circ \mathcal{Y})_{i_1,\ldots,i_m,i'_1,\ldots,i'_{m'}} = x_{i_1,\ldots,i_m} y_{i'_1,\ldots,i'_{m'}} \tag{1}$$

for all index values.

Definition 2.2 (Mode-k Product). *The mode-k product* $\mathcal{X} \times_k \mathbf{M}$ *of a tensor* $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_m}$ *and a matrix* $\mathbf{M} \in \mathbb{R}^{I'_k \times I_k}$ *is defined by*

$$(\mathcal{X} \times_k \mathbf{M})_{i_1,\ldots,i_{k-1},j,i_{k+1},\ldots,i_m} = \sum_{i_k=1}^{I_k} x_{i_1,\ldots,i_m} m_{j,i_k} \tag{2}$$

for all index values.

2.2 SVM Model

Vapnik introduced support vector machines (SVMs) [14] based on the maximum-margin principle. Essentially, SVMs integrate the hinge loss and the L2-norm regularization. The decision function with a linear kernel is as follows[1]:

$$\hat{y} = w_0 + \sum_{i=1}^{d} w_i x_i = w_0 + \sum_{p=1}^{m} \sum_{i_p=1}^{I_p} w_{i_p}^{(p)} x_{i_p}^{(p)} \tag{3}$$

where $\mathbf{x}$ is simply a concatenation of features from different views in the multi-view setting, *i.e.*, $\mathbf{x}^T = \left(\mathbf{x}^{(1)^T}, ..., \mathbf{x}^{(m)^T}\right)$, as shown in Figure 1.

Obviously, no interactions between views are explored in Eq. (3). Through the employment of a nonlinear kernel, SVMs can implicitly project data from the feature space into a more complex high-dimensional space, which allows SVMs to model higher order interactions between features. However, as discussed in [10], all interaction parameters of nonlinear SVMs are completely independent.

For nonlinear SVMs, there must be enough instances $\mathbf{x} \in \mathcal{D}$ where $x_{i_p}^{(p)} \neq 0$ and $x_{i_q}^{(q)} \neq 0$ to reliably estimate the second-order interaction parameter $w_{i_p,i_q}^{(p,q)}$. The instances with either $x_{i_p}^{(p)} = 0$ or $x_{i_q}^{(q)} = 0$ cannot be used for estimating $w_{i_p,i_q}^{(p,q)}$. That is to say, on a sparse dataset where there are too few or even no cases for some higher order interactions, nonlinear SVMs are likely to degenerate into linear SVMs. Therefore, factorizing and projecting higher order interactions into a consistent latent space would facilitate parameter estimation under sparsity.

[1]The sign function is omitted, because the analysis and conclusions can easily extend to other generalized linear models, *e.g.*, logistic regression.

2.3 STM Model

Cao et al. investigated multi-view classification by modeling features interactions across views as a tensor, *i.e.*, $\mathcal{X} = \mathbf{x}^{(1)} \circ \cdots \circ \mathbf{x}^{(m)} \in \mathbb{R}^{I_1 \times \cdots \times I_m}$ [2] and solved the problem in the framework of support tensor machines (STMs) [13]. Basically, as shown in Figure 1, only the highest-order interactions are explored:

$$\hat{y} = \sum_{i_1=1}^{I_1} \cdots \sum_{i_m=1}^{I_m} w_{i_1,\ldots,i_m} \left(\prod_{p=1}^{m} x_{i_p}^{(p)} \right) \tag{4}$$

where $w_{i_1,\ldots,i_m} = \prod_{p=1}^{m} w_{i_p}^{(p)}$, *i.e.*, a rank-one decomposition of the weight tensor $\mathcal{W} \in \mathbb{R}^{I_1 \times \cdots \times I_m}$ [2].

However, estimating a lower order interaction (*e.g.*, a pairwise one) reliably is easier than estimating a higher order one, and lower order interactions can usually explain the data sufficiently [12, 1]. Thus, it motivates us to include the lower order interactions into the model. Moreover, instead of a rank-one decomposition, it is desirable to apply a higher rank decomposition of the weight tensor to capture more latent factors and thereby achieving a better approximation to the original interaction parameters.

2.4 FM Model

Rendle introduced factorization machines (FMs) [10] that combine the advantages of SVMs with factorization models. The model equation of a 2-way FM is as follows:

$$\hat{y} = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{5}$$

where $d = \sum_{p=1}^{m} I_p$ and $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} v_{j,f}$.

Note that pairwise interactions between all features are included in FMs without consideration of the view segmentation. In the multi-view setting, there can be redundant correlations between features within the same view, *i.e.*, intra-view correlations, which are thereby unworthy of consideration. Field-aware FMs [6] integrate the field/view concept into the FM model where the extension is limited to the second-order feature interactions. The coupled group lasso model [16] is essentially an application of the 2-way FMs in multi-view classification. Let MVFM denote the multi-view variation of FMs with a decision function as follows:

$$\begin{aligned} \hat{y} = w_0 &+ \sum_{p=1}^{m} \sum_{i_p=1}^{I_p} w_{i_p}^{(p)} x_{i_p}^{(p)} + \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \left\langle \mathbf{v}_{i_1}^{(1)}, \mathbf{v}_{i_2}^{(2)} \right\rangle x_{i_1}^{(1)} x_{i_2}^{(2)} \\ &+ \cdots + \sum_{i_{m-1}=1}^{I_{m-1}} \sum_{i_m=1}^{I_m} \left\langle \mathbf{v}_{i_{m-1}}^{(m-1)}, \mathbf{v}_{i_m}^{(m)} \right\rangle x_{i_{m-1}}^{(m-1)} x_{i_m}^{(m)} \end{aligned} \tag{6}$$

The pairwise interaction parameter $w_{i_p,i_q}^{(p,q)} = \left\langle \mathbf{v}_{i_p}^{(p)}, \mathbf{v}_{i_q}^{(q)} \right\rangle$ in Eq. (6) indicates that $w_{i_p,i_q}^{(p,q)}$ can be learned from instances with $x_{i_p}^{(p)} \neq 0$ and some $x_{i_{q'}}^{(q')} \neq 0$ (sharing $\mathbf{v}_p$), or $x_{i_q}^{(q)} \neq 0$ and some $x_{i_{p'}}^{(p')} \neq 0$ (sharing $\mathbf{v}_q$). It makes MVFM (and

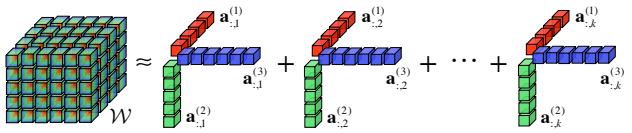


Figure 2: CP factorization. The third-order ($m = 3$) tensor $\mathcal{W}$ is approximated by k rank-one tensors. The f-th factor tensor is the tensor product of three vectors, *i.e.*, $\mathbf{a}^{(1)}_{:,f} \circ \mathbf{a}^{(2)}_{:,f} \circ \mathbf{a}^{(3)}_{:,f}$.

FMs) more effective under sparsity than SVMs where only instances with $x^{(p)}_{i_p} \neq 0$ and $x^{(q)}_{i_q} \neq 0$ can be used to learn the second-order feature interaction $w^{(p,q)}_{i_p,i_q}$.

However, the interaction parameters for different orders are completely independent in MVFM (and FMs), *e.g.*, the first-order interaction parameter, $w^{(p)}_{i_p}$, and the second-order interaction parameter, $\mathbf{v}^{(p)}_{i_p}$, in Eq. (6). Furthermore, as illustrated in Figure 1, additional sets of model parameters will be introduced when we consider higher order feature interactions in MVFM (and FMs) which makes the learning process harder. A more effective strategy is needed when including the higher order interactions.

3. MULTI-VIEW MACHINE MODEL

3.1 Model Formulation

The key challenge of multi-view prediction is to model the interactions between features from different views, wherein complementary information is contained. Here, we consider nesting all interactions up to the mth-order between m views:

$$\hat{y} = \underbrace{\beta_0}_{\text{global bias}} + \underbrace{\sum_{p=1}^{m}\sum_{i_p=1}^{I_p} \beta^{(p)}_{i_p} x^{(p)}_{i_p}}_{\text{first-order interactions}} + \underbrace{\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2} \beta^{(1,2)}_{i_1,i_2} x^{(1)}_{i_i} x^{(2)}_{i_2} + \cdots + \sum_{i_{m-1}=1}^{I_{m-1}}\sum_{i_m=1}^{I_m} \beta^{(m-1,m)}_{i_{m-1},i_m} x^{(m-1)}_{i_{m-1}} x^{(m)}_{i_m}}_{\text{second-order interactions}} + \cdots + \underbrace{\sum_{i_1=1}^{I_1}\cdots\sum_{i_m=1}^{I_m} \beta_{i_1,\ldots,i_m}\left(\prod_{p=1}^{m} x^{(p)}_{i_p}\right)}_{m\text{th-order interactions}} \tag{7}$$

Let us add an extra feature with constant value 1 to the feature vector $\mathbf{x}^{(p)}$, *i.e.*, $\mathbf{z}^{(p)^T} = (\mathbf{x}^{(p)^T}, 1) \in \mathbb{R}^{I_p+1}, \forall p = 1, ..., m$. Then, Eq. (7) can be compactly rewritten as:

$$\hat{y} = \sum_{i_1=1}^{I_1+1}\cdots\sum_{i_m=1}^{I_m+1} w_{i_1,\ldots,i_m}\left(\prod_{p=1}^{m} z^{(p)}_{i_p}\right) \tag{8}$$

where $w_{I_1+1,\ldots,I_m+1} = \beta_0$ and $w_{i_1,\ldots,i_m} = \beta_{i_1,\ldots,i_m}, \forall i_p \leq I_p$. For $w_{i_1,\ldots,i_m}$ with some indexes satisfying $i_p = I_p + 1$, it encodes lower order interaction between views whose $i_p \leq I_p$. Hereinafter, let $w^{(p)}_{i_p}$ denote $w_{i_1,\ldots,i_m}$ where only $i_p \leq I_p$ and $i_q = I_q + 1, q \neq p$, and let $w^{(p,q)}_{i_p,i_q}$ denote $w_{i_1,\ldots,i_m}$ where $i_p \leq I_p$, $i_q \leq I_q$ and $i_r = I_r + 1, r \notin \{p, q\}$, *etc.*

The number of parameters in Eq. (8) is $\prod_{p=1}^{m}(I_p + 1)$, which can make the model prone to overfitting and ineffective on sparse data. Therefore, we assume that the effect of interactions has a low rank and the mth-order weight tensor $\mathcal{W} = \{w_{i_1,\ldots,i_m}\} \in \mathbb{R}^{(I_1+1)\times\cdots\times(I_m+1)}$ can be factorized into k factors:

$$\mathcal{W} = \mathbf{C} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_m \mathbf{A}^{(m)} \tag{9}$$

where $\mathbf{A}^{(p)} \in \mathbb{R}^{(I_p+1)\times k}$, and $\mathbf{C} \in \mathbb{R}^{k\times\cdots\times k}$ is the identity tensor, *i.e.*, $c_{i_1,\ldots,i_m} = \delta(i_1 = \cdots = i_m)$. Basically, Eq. (9) is a CANDECOMP/PARAFAC (CP) factorization [7] as shown in Figure 2, with element-wise notation $w_{i_1,\ldots,i_m} = \sum_{f=1}^{k}\prod_{p=1}^{m} a^{(p)}_{i_p,f}$. The number of model parameters is reduced to $k\sum_{p=1}^{m}(I_p + 1) = k(m + d)$. It transforms Eq. (8) into:

$$\hat{y} = \sum_{i_1=1}^{I_1+1}\cdots\sum_{i_m=1}^{I_m+1}\left(\prod_{p=1}^{m} z^{(p)}_{i_p}\right)\left(\sum_{f=1}^{k}\prod_{p=1}^{m} a^{(p)}_{i_p,f}\right) \tag{10}$$

We name this model in Eq. (10) as multi-view machines (MVMs). As shown in Figure 3, the full-order interactions between multiple views are modeled in a tensor, and they are factorized collectively. The model parameters that have to be estimated are:

$$\mathbf{A}^{(p)} \in \mathbb{R}^{(I_p+1)\times k},\ p = 1, ..., m \tag{11}$$

where the i_p-th row $\mathbf{a}^{(p)^T}_{i_p} = (a^{(p)}_{i_p,1}, ..., a^{(p)}_{i_p,k})$ within $\mathbf{A}^{(p)}$ describes the i_p-th feature in the p-the view with k factors.

Definition 3.1 (Bias Factor). *The bias factor is a collection of bias from each factor. In MVMs, the last row of $\mathbf{A}^{(p)}$, i.e., $\mathbf{a}^{(p)^T}_{I_p+1}$, represents the bias factor of the p-th view, and it is always associated with $z^{(p)}_{I_p+1} = 1$ in Eq. (10).*

Hence,

$$w_{I_1+1,\ldots,I_m+1} = \sum_{f=1}^{k}\prod_{p=1}^{m} a^{(p)}_{I_p+1,f} \tag{12}$$

is the *global bias*, denoted as w_0 hereinafter.

3.2 Time Complexity

Next, we show how to compute the decision function of MVMs efficiently. The straightforward time complexity of Eq. (10) is $O(k\prod_{p=1}^{m}(I_p + 1))$. However, we observe that there is no model parameter which directly depends on feature interactions, due to the joint factorization. Therefore, the time complexity can be largely reduced.

Lemma 3.1. *The model equation of MVMs can be computed in $O(k(m + d))$.*

Proof. The feature interactions in Eq. (10) can be reformulated as:

$$\begin{aligned}
&\sum_{i_1=1}^{I_1+1}\cdots\sum_{i_m=1}^{I_m+1}\left(\prod_{p=1}^{m} z^{(p)}_{i_p}\right)\left(\sum_{f=1}^{k}\prod_{p=1}^{m} a^{(p)}_{i_p,f}\right)\\
=&\sum_{f=1}^{k}\sum_{i_1=1}^{I_1+1}\cdots\sum_{i_m=1}^{I_m+1}\left(\prod_{p=1}^{m} z^{(p)}_{i_p} a^{(p)}_{i_p,f}\right)\\
=&\sum_{f=1}^{k}\left(\sum_{i_1=1}^{I_1+1} z^{(1)}_{i_1} a^{(1)}_{i_1,f}\right)\cdots\left(\sum_{i_m=1}^{I_m+1} z^{(m)}_{i_m} a^{(m)}_{i_m,f}\right)
\end{aligned} \tag{13}$$

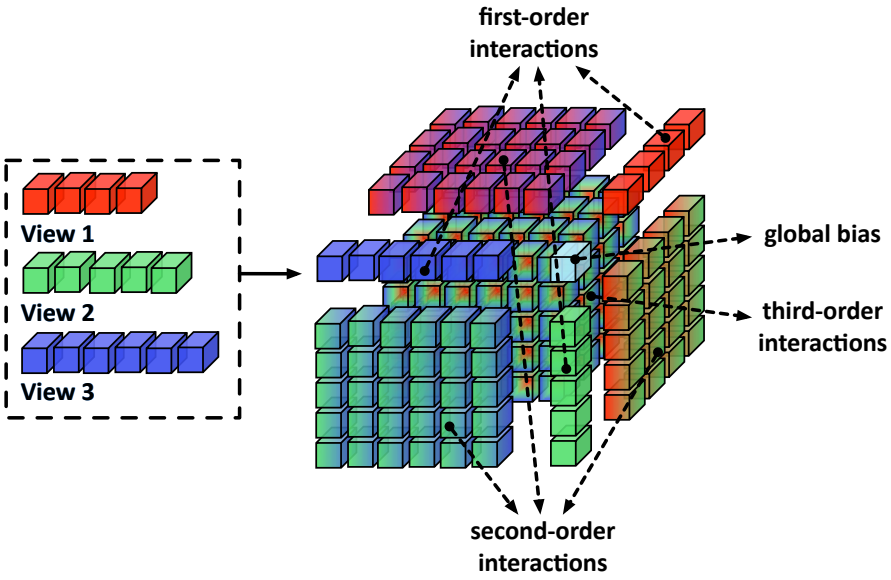


Figure 3: Multi-view machines. The full-order feature interactions in multi-view data are modeled in a tensor and jointly factorized into a common latent subspace.

This equation has only linear complexity in both k and I_p. Thus, its time complexity is $O(k(m+d))$, which is in the same order of the number of model parameters. □

3.3 Discussion

The joint factorization of the global bias and the full-order interactions is important for MVMs. Thus, dependencies exist when interactions share the same feature. It benefits MVMs for parameter estimation under sparsity, since the latent factor $\mathbf{a}_{i_p}^{(p)}$ can be learned from any instances whose $x_{i_p}^{(p)} \neq 0$, which allows the second-order interaction $w_{i_p,i_q}^{(p,q)}$ can be approximated from instances whose $x_{i_p}^{(p)} \neq 0$ *or* $x_{i_q}^{(q)} \neq 0$ rather than instances whose $x_{i_p}^{(p)} \neq 0$ *and* $x_{i_q}^{(q)} \neq 0$ as in nonlinear SVMs. Therefore, the interaction parameters in MVMs can be effectively learned without direct observations of such interactions in a training set of sparse data.

The main difference between FMs and MVMs is that the interaction parameters for different orders are completely independent in FMs, *e.g.*, the first-order interaction $w_{i_p}^{(p)}$ and the second-order interaction $\mathbf{v}_{i_p}^{(p)}$ in Eq. (6). On the contrary, in MVMs, all orders of interactions share the same set of latent factors $\mathbf{a}_{i_p}^{(p)}$ in Eq. (10). For example, the combination of $\mathbf{a}_{i_p}^{(p)}$ and the bias factors from other $m-1$ views, *i.e.*, $\mathbf{a}_{I_1+1}^{(1)}, ..., \mathbf{a}_{I_{p-1}+1}^{(p-1)}, \mathbf{a}_{I_{p+1}+1}^{(p+1)}, ..., \mathbf{a}_{I_m+1}^{(m)}$, approximates the first-order interaction $w_{i_p}^{(p)}$. Similarly, we can obtain the second-order interaction $w_{i_p,i_q}^{(p,q)}$ by combining $\mathbf{a}_{i_p}^{(p)}$, $\mathbf{a}_{i_q}^{(q)}$ and other $m-2$ bias factors. Therefore, compared to MVMs, FMs are partially and independently factorized. Such difference is more significant for higher order FMs. As summarized in Table 3, assuming the same number of factors for different orders of interactions, the model complexity of an m-way FM is $O(kmd)$ which can be much larger than $O(k(m+d))$ of MVMs.

3.4 Extensions

MVMs are flexible in the interactions of interests. That is to say, when there are too many views available for a learning task and interactions between some of them may obviously be physically meaningless, or sometimes the very high order interactions may not be intuitively interpretable, it is not desirable to include these potentially redundant interactions in the model. In such scenarios, one can (1) partition (overlapping) groups of views, (2) construct multiple MVMs on these view groups where the full-order interactions within each group are included, and (3) implement a coupled matrix/tensor factorization [5]. This strategy excludes the inter-group feature interactions.

On the other hand, in scenarios where the view segmentation is not given, one may be aggressive to consider interactions between all features, which becomes the problem setting of the original FMs. To achieve this purpose, we can simply repeat the same feature set in multiple views. Overall, MVMs are applicable with either conservative or radical strategies. Although MVMs can be easily adapted to include/exclude interactions between any features, that is outside the scope of this paper; our focus is on investigating how to effectively explore the full-order feature interactions from a given set of views.

4. LEARNING MULTI-VIEW MACHINES

To learn model parameters in MVMs, we consider the following regularization framework:

$$\underset{\Theta}{\operatorname{argmin}} \sum_{(\mathbf{x},y)\in\mathcal{D}} \mathcal{L}(\hat{y}(\mathbf{x}|\Theta), y) + \lambda\Omega(\Theta) \tag{14}$$

where $\Theta = \{\mathbf{A}^{(p)} | \ p = 1, ..., m\}$ represents all model parameters, $\mathcal{L}(\cdot)$ is the loss function, $\Omega(\cdot)$ is the regularization term, and λ is the trade-off between the empirical loss and the risk of overfitting.

Table 3: Summary of related work. Model complexity refers to both the number of parameters in the model and the time complexity to compute the decision function.

Method	Model complexity	Feature interactions	Parameter factorization
Support vector machines (SVMs) [14]	$O(d)$	first-order	none
Support tensor machines (STMs) [13]	$O(kd)$	highest-order	factorized ($k = 1$ [2])
Factorization machines (FMs) [10]	$O(kmd)$	up to full-order	partially and independently factorized
Multi-view machines (MVMs)	$O(k(m+d))$	full-order	fully and jointly factorized

Importantly, MVMs can be used to perform a variety of machine learning tasks, depending on the choices of the loss function. For example, to conduct regression, one can use the square error:

$$\mathcal{L}^S(\hat{y}(\mathbf{x}|\Theta), y) = (\hat{y}(\mathbf{x}|\Theta) - y)^2 \tag{15}$$

and for classification problems, the logit loss can be used:

$$\mathcal{L}^L(\hat{y}(\mathbf{x}|\Theta), y) = \log(1 + \exp(-y \cdot \hat{y}(\mathbf{x}|\Theta))) \tag{16}$$

or the hinge loss.

The regularization term is chosen based on our prior knowledge about the model parameters. Typically, we can apply L2-norm:

$$\Omega^{L2}(\Theta) = \|\Theta\|_2^2 = \sum\nolimits_i \theta_i^2 \tag{17}$$

4.1 Gradient Descent

The model can be learned efficiently by alternating least square (ALS), stochastic gradient descent (SGD), L-BFGS, *etc.* From Eq. (13), the gradient of the MVM model is:

$$\begin{aligned}\frac{\partial \hat{y}(\mathbf{x}|\Theta)}{\partial \theta} =& z_{i_p}^{(p)} \left(\sum_{i_1=1}^{I_1+1} z_{i_1}^{(1)} a_{i_1,f}^{(1)} \right) \cdots \left(\sum_{i_{p-1}=1}^{I_{p-1}+1} z_{i_{p-1}}^{(p-1)} a_{i_{p-1},f}^{(p-1)} \right) \\ & \left(\sum_{i_{p+1}=1}^{I_{p+1}+1} z_{i_{p+1}}^{(p+1)} a_{i_{p+1},f}^{(p+1)} \right) \cdots \left(\sum_{i_m=1}^{I_m+1} z_{i_m}^{(m)} a_{i_m,f}^{(m)} \right)\end{aligned} \tag{18}$$

where $\theta = a_{i_p,f}^{(p)}$, and $z_{i_p}^{(p)} = 1$ if $i_p = I_p + 1$, otherwise $z_{i_p}^{(p)} = x_{i_p}^{(p)}$. It validates that MVMs possess the multilinearity property [11], because the gradient along θ is independent of the value of θ itself.

Note that in Eq. (18), the sum $\sum_{i_p=1}^{I_p+1} z_{i_p}^{(p)} a_{i_p,f}^{(p)}$ and their product can be precomputed for updating the f-th factor of all features. Hence, each gradient can be computed in constant time $O(1)$. In an iteration, including the precomputation time, all parameters can be updated in $O(k(m+d))$. It can be even reduced under sparsity, where most of the elements in $\mathbf{x}$ (or $\mathbf{z}$) are 0 and thus, the sums have only to be computed over the non-zero elements, and only non-zero parameters need to be updated according to Eq. (18).

It is straightforward to embed Eq. (18) into the gradient of the loss functions *e.g.*, Eqs. (15)-(16), for direct optimization, as follows:

$$\frac{\partial \mathcal{L}^S(\hat{y}(\mathbf{x}|\Theta), y)}{\partial \theta} = 2(\hat{y}(\mathbf{x}|\Theta) - y) \cdot \frac{\partial \hat{y}(\mathbf{x}|\Theta)}{\partial \theta} \tag{19}$$

$$\frac{\partial \mathcal{L}^L(\hat{y}(\mathbf{x}|\Theta), y)}{\partial \theta} = \frac{-y}{1 + \exp(y \cdot \hat{y}(\mathbf{x}|\Theta))} \cdot \frac{\partial \hat{y}(\mathbf{x}|\Theta)}{\partial \theta} \tag{20}$$

Moreover, the gradient of the regularization term $\Omega(\Theta)$ can be derived:

$$\frac{\partial \Omega^{L2}(\Theta)}{\partial \theta} = 2\theta \tag{21}$$

The SGD optimization method for MVMs is summarized in Algorithm 1, where the model parameters are first initialized from a zero-mean normal distribution with standard deviation σ, and the gradients in line 8 can be computed according to Eqs. (19)-(21).

Algorithm 1 Stochastic Gradient Descent for MVMs

Input: Training data $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, ..., n\}$, number of factors k, regularization parameter λ, learning rate η, standard deviation σ
Output: Model parameters $\Theta = \{\mathbf{A}^{(p)} \in \mathbb{R}^{(I_p+1)\times k} \mid p = 1, ..., m\}$

```
1: Initialize A^(p) ~ N(0, σ)
2: repeat
3:   for (x, y) ∈ D do
4:     for p := 1 to m do
5:       for i_p := 1 to I_p + 1 do
6:         if z_{i_p}^(p) ≠ 0 then
7:           for f := 1 to k do
8:             θ ← θ − η(∂L(ŷ(x|Θ),y)/∂θ + λ ∂Ω(Θ)/∂θ) where θ is a_{i_p}^(p)
9:           end for
10:        end if
11:      end for
12:    end for
13:  end for
14: until convergence
```

4.2 Distributed Implementation

Web-scale applications in the real world always contain a huge number of entities represented in multiple views, *e.g.*, users, movies, ads, queries, with millions of instances, *e.g.*, ratings, impressions. In this section, we introduce a design for scalable learning and its implementation on top of GraphX [4], which is a component of Spark [17] for graphs and graph-parallel computation and provides high performance, scalability and fault-tolerance for the learning process.

The training data is represented as a graph that contains two types of vertices, *i.e.*, instance vertices and feature vertices. A directed edge from a feature vertex to an instance vertex exists if the feature is non-zero in the instance. The graph representation is efficient due to the inherent sparsity of the training data. The factor vector (or weight coefficient that is not factorized in some baselines) of a feature is represented as attributes of the corresponding feature vertex,

Table 4: The statistics and parameters for each dataset. The number in braces indicates the dimensionality of the corresponding view.

Dataset	Views	n	η	λ	k	#iterations
MovieLens	users (138,493), movies (27,278), impl. (27,278)	20,000,263	0.1	0.01	20	200
BingAds	queries (958,426), ad URLs (1,935,510), impressions (18)	28,622,281	0.1	0.01	20	200

the label information of an instance is represented as the attribute of the corresponding instance vertex, and the feature value is represented as the edge attribute. For distributed learning, the graph is partitioned and scheduled to different computing nodes for execution by the underlying distributed graph framework. In this manner, both data parallelism and model parallelism are achieved.

Each iteration in Algorithm 1 is composed of two major steps, *i.e.*, feed-forward and back-propagation. In the feed-forward process, messages are sent from feature vertices to instance vertices following the edges which are arrays $\mathbf{b} = \mathbb{R}^k$ where $b_f = z_{i_p}^{(p)} * a_{i_p,f}^{(p)}$. An instance vertex receives all messages from its connected feature vertices and sums them in view-wise. The predicted value is then computed accordingly based on Eq. (13). In the back-propagation process, messages are sent from instance vertices to feature vertices which are arrays $\mathbf{c} = \mathbb{R}^k$ where each element represents a gradient. A feature vertex averages the gradients received from its connected instance vertices and updates the factor vector accordingly based on Algorithm 1.

5. EXPERIMENTS

5.1 Experimental Setup

Data collections. To evaluate the performance of multi-view prediction, we conduct extensive experiments on the MovieLens dataset for movie rating prediction (regression) and the BingAds dataset for CTR prediction (classification), respectively.

- **MovieLens dataset**[2]. A regression task for rating prediction is studied on the public dataset, MovieLens. Ratings are made on a 5-star scale, with half-star increments. Each rating in this dataset has three views, *i.e.*, users, movies and implicit user feedback. 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 implicit feedback view is constructed following SVD++ [8] to capture users' history information. Specifically, it consists of all movies the user has ever rated and it is normalized. Hence, this view makes use of implicit feedback information and indicates users' preference. For this problem, the performance is measured by root mean square error (RMSE).

- **BingAds dataset**[3]. A classification task for CTR prediction is investigated on a dataset collected from ad impression logs of Bing, comprising three views: queries, ad URLs and impression information. Each instance is labeled as 1 if the impression is clicked and -1 otherwise. The query view consists of unigrams of user query words[4]. The ad URL view includes URLs corresponding to the shown ads. The impression view is composed of impression locations and matched types. All features are hashed as integer ids and represented by binary values. There are multiple non-zero features in the query view, only one non-zero feature in the ad URL view, and 2 non-zero features in the impression view. Area under the curve (AUC) is used as the evaluation metric.

See Table 4 for more information about the statistics and parameters used for each dataset.

Compared models. In order to demonstrate the effectiveness of modeling feature interactions in multi-view data, we compare the following models:

- **Linear regression/logistic regression (LR)**. We implement linear regression for regression tasks, *e.g.*, rating prediction, and logistic regression for classification tasks, *e.g.*, CTR prediction. They are essentially representative linear models (including linear SVMs), but with different loss functions, *e.g.*, the square error and the logit loss, respectively. It is discussed in the form of SVMs in detail in Section 2.2.

- **Tensor factorization (TF)** is a generalization of matrix factorization to higher orders. We can directly use tensors to model the multi-view data and factorize the weight tensor [2]. When the hinge loss is used, it can be solved in the framework of support tensor machines (STMs) [13]. When there are two views with categorical features, TF is reduced to conventional matrix factorization without bias terms. It is introduced as the STM model in Section 2.3.

- **Factorization machine (FM)** explores pairwise interactions between all features without consideration of the view segmentation [10]. Its adaptation in the multi-view setting, denoted as mvFM, considers feature interactions across views with the decision function in Eq. (6). This FM variation is specifically reviewed in Section 2.4. In addition to the popular 2-way FM model, we also implemented 3-way FMs to include higher order interactions, denoted as mvFM-3d, where feature interactions with different orders are modeled but with separate sets of parameters[5]. Moreover, we regularized the second-order and the third-order interactions sharing the same latent factors and assigned

[2] `http://grouplens.org/datasets/movielens`

[3] The dataset is used internally in the Bing Ads team for model experiments rather than training product models.

[4] Stemming, lemmatization, removing stop-words, *etc.*, are handled beforehand.

[5] In experiments, the rank k in mvFM-3d is set to 10 for both the second-order and the third-order interactions, so that the number of model parameters stays the same as other factorization baselines.

Table 5: Prediction accuracy. ↓ indicates the smaller the value the better the performance; ↑ indicates the larger the value the better the performance.

Dataset	MovieLens (RMSE) ↓	BingAds (AUC) ↑
MVM	0.8376	0.7917
FM	0.8681	0.7872
mvFM	0.8447	0.7729
mvFM-3d	0.9060	0.7201
mvFM-reg	0.9807	0.6947
TF	0.8572	0.6645
LR	1.0017	0.7450

the global bias and the first-order interactions with independent weights that are not factorized, denoted as mvFM-reg.

- **Multi-view machine (MVM)** is our proposed model to explore the full-order interactions embedded within multi-view data, where feature interactions with different orders are jointly factorized and thereby sharing the same set of latent factors.

Configuration. All compared models are implemented on top of GraphX in Spark and trained with iterative forward and backward steps as in introduced in Section 4.2. The stochastic gradient descent with adaptive (sub)gradient [3] is used as the optimization method. The code has been made available at GitHub[6].

For a fair comparison, the same parameter setting in Table 4 is used for all compared models. A deterministic data sampling is applied on both datasets so that 80% data is used for training and the other 20% for test. All models are trained with the same hardware configuration, where 10 homogeneous computing nodes are connected via 40Gbps Infiniband network and each node has 16 2.40GHz Intel(R) Xeon(R) CPU E5-2665 cores and 128GB memory. There are 1 driver configured with 25GB memory and 10 workers configured with 100GB memory. The data is partitioned into 160 partitions based on node degree [15] to balance the load in each core and reduce the communication among cores.

5.2 Multi-view Prediction Accuracy

The experimental results are shown in Table 5. On the MovieLens dataset, the smaller RMSE, the better an algorithm. We can observe that LR is a simple baseline because as a conventional linear model, it neglects any interactions between features. However, such feature interactions can be critical in the sparse data, which explains much better performance achieved by FM through including pairwise feature interactions. We further find that mvFM is able to outperform FM by excluding intra-view correlations. In our case of movie rating prediction on SVD++ data [8], intra-view correlations indicate interactions between movies the user has rated before which do not have direct influence on the user's preference of the current movie. However, inter-view correlations include interactions between the current movie and those movies rated by the user which are critical by matching the latent factor of the current movie and that of rated movies in the past. It validates the importance of introducing the view concept to learn an effective model in many problems.

The two variations of mvFM, mvFM-3d and mvFM-reg, add the third-order feature interactions in addition to the 2-way mvFM. The difference is that mvFM-reg uses the same set of latent factors for the second-order and the third-order interactions, while mvFM-3d introduces different parameters for interactions with different orders. It seems that the inclusion of higher order interactions fails to bring us any accuracy improvement, but TF manages to perform well by solely relying on the highest-order interactions. It might imply that the consensus and complementary information between lower order and higher order interactions need to be better taken care of, which leads to our MVM model. Overall, we can observe from Table 5 that MVM achieves the best performance through joint factorization of feature interactions with different orders.

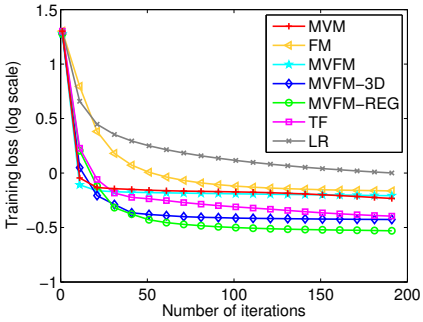
On the BingAds dataset, FM shows better performance than mvFM implying intra-view correlations might be important for this problem. Consider the impression view comprising 2 non-zero features for each instance, *i.e.*, impression location and matched type. Feature interactions between impression locations and matched types are not included in mvFM, whose variations and TF are even defeated by the linear model, LR.

5.3 Convergence Efficiency

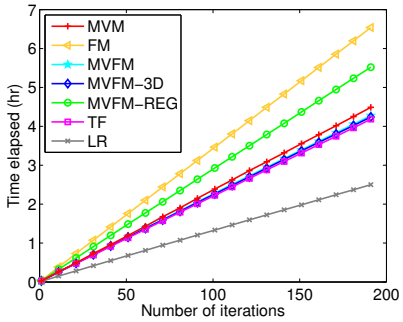
In this section, we show more details about the training procedure of compared models. Figure 4(a) illustrates the training loss on the MovieLens dataset where results are plotted in a log scale for better resolution on final convergence in the late stage. We observe that several compared models can fit themselves very well on the training data. For example, the final converged training RMSE of mvFM-reg is as small as 0.5879; however, its test RMSE turns out to be 0.9807. It is clear that these models are easily prone to overfitting. On the contrary, our MVM model fits the training data with a moderate training RMSE = 0.7855, and achieves the best test RMSE = 0.8376, as shown in Table 5.

Figure 5(a) shows similar observations on the BingAds dataset where the overfitting problem is more significant. There is a possible reasoning about the capacity of MVMs to avoid overfitting. The joint factorization of the global bias, the first-order interactions, the second-order and higher order interactions plays a key role through the effect of bias factors $\mathbf{a}^{(p)}_{I_p+1}$. The bias factor of each view will be updated by all instances, since it is always associated with a non-zero feature. Considering lower order interactions, other factor vectors contribute to the decision function by combining with bias factors of other views, as shown in Eq. (10). Therefore, bias factors are frequently retrieved and updated, making themselves relatively sensitive among model parameters. Our initial experiments found that the MVM model would suffer an unstable convergence process without the use of adaptive gradient [3]. Fortunately, this problem can be greatly alleviated by adaptively choosing an appropriate learning rate, as illustrated by the monotonic convergence process shown in Figure 4(a) and Figure 5(a). With this problem solved, bias factors bring MVMs with the capacity to avoid overfitting by storing and providing the *global knowledge*, because each training instance will update them and each test instance will be predicted based on them. Such

[6] `https://github.com/cloudml/zen/tree/mvm_opt/ml/src/main/scala/com/github/cloudml/zen/ml/recommendation`

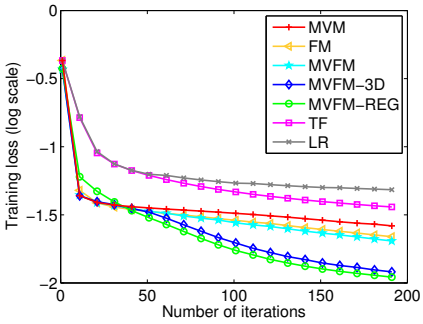


(a) Training loss.

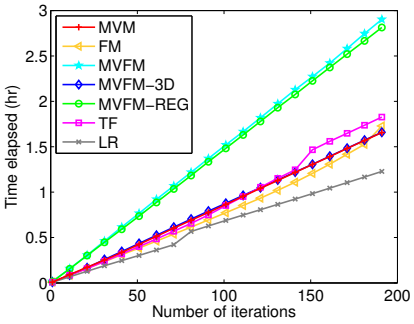


(b) Time cost.

Figure 4: The training procedure on the MovieLens dataset.



(a) Training loss.



(b) Time cost.

Figure 5: The training procedure on the BingAds dataset.

global knowledge is critical to model the bias information per view per factor and thus makes MVMs a robust model. In contrast, other compared methods (*e.g.*, LR, FM) use a single model parameter, *i.e.*, the global bias, for the purpose of the global knowledge, which is insufficient.

Figure 4(b) and Figure 5(b) compare the time cost of each model on the MovieLens dataset and the BingAds dataset, respectively. We find that our MVM model has the best system performance among models that consider high order feature interactions without bringing too much system overhead than the linear model, LR. The steep rise of LR and TF in Figure 5(b) appears because of fault occurrence during training and automatic recovery by Spark.

5.4 Hyperparameter Sensitivity

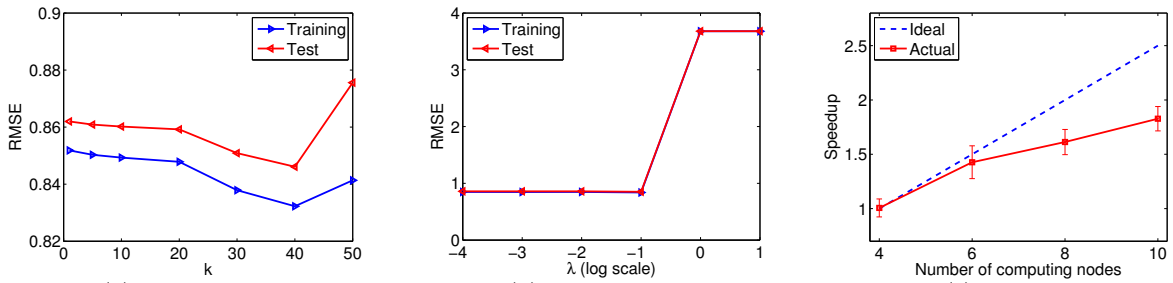
In all experiments, the parameter η is heuristically set to 0.1 for MVMs and other baseline models, since the performance is insensitive to the initial learning rate by using the adaptive gradient [3]. In this section, we study the influence of the other two key hyperparameters, k and λ, in our MVM model. Due to the space limit, only results on MovieLens dataset are presented.

Experiments are conducted for different k and the results are shown in Figure 6(a). In contrast to findings in other related models based on latent factors [16, 11] where accuracy can steadily get improved with larger k, we observe that both the converged training loss and test loss turn out to be better with the increasing of k and reach a peak at $k = 40$, after which the accuracy will obviously decrease. It is reasonable in a general sense, because larger k renders the model with greater expressiveness which also leads to higher risk of overfitting. When the expressiveness of the model overqualifies the information embedded in data, it is likely that the model will fit the training data very well yet fail on the test data. On the other hand, larger k leads to more model parameters which make it harder to learn an effective model within limited iterations. In general, Figure 6(a) indicates that the performance of our MVM model in Table 5 can be further improved with $k = 40$ at the cost of a linear increase in both runtime and memory.

Moreover, we investigate the influence of the regularization parameter λ and present the results in Figure 6(b). We observe that our MVM model is insensitive to λ in a relatively large range and performs well and steadily when $\lambda \leq 0.1$. It makes sense because large λ will let the regularization term override the effect of the loss function and thus dominate the objective.

5.5 Scalability

To investigate the scalability of our distributed learning framework introduced in Section 4.2, we compute the speedup factor relative to the time cost with 4 nodes by varying the number of computing nodes from 4 to 10. The number of training data partitions is always configured to be the number of cores. Experiments are repeated 10 iterations, and the average speedup factors with standard deviations are reported in Figure 6(c). We can observe that the speedup appears to be close to linear and close to the ideal speedup factors. Therefore, our distributed implementation of the MVM model is very scalable for web-scale applications.



(a) Influence of k. (b) Influence of λ. (c) Speedup.

Figure 6: Sensitivity analysis of hyperparameters and the speedup of the distributed learning framework.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a multi-view machine (MVM) model and presented a stochastic gradient descent (SGD) learning method with a distributed implementation on Spark. In general, the model is particularly designed for data that is composed of features from multiple views, between which the full-order interactions are effectively explored. In contrast to other models that include only partial feature interactions or factorize different orders of interactions separately, MVMs jointly factorize the full-order feature interactions and thereby benefiting parameter estimation under sparsity and rendering the model with the capacity to avoid overfitting. Moreover, MVMs can be applied to a variety of supervised machine learning tasks, including classification and regression. Empirical studies on real-world web application datasets demonstrate the effectiveness of MVMs on modeling feature interactions in multi-view data, which outperform baseline models for multi-view prediction.

The MVM model can be further investigated in several directions for future work. For example, in addition to SGD, we are interested in implementation of other learning algorithms in a distributed environment to facilitate convergence efficiency, *e.g.*, alternating least square (ALS) and Markov Chain Monto Carlo (MCMC) for MVMs. It is also interesting to have our model applied to other multi-view prediction problems. Moreover, defining an evaluation metric for an effective view segmentation would be critical for the subsequent multi-view learning.

7. REFERENCES

[1] Yuanzhe Cai, Miao Zhang, Dijun Luo, Chris Ding, and Sharma Chakravarthy. Low-order tensor decompositions for social tagging recommendation. In *WSDM*, pages 695–704. ACM, 2011.

[2] Bokai Cao, Lifang He, Xiangnan Kong, Philip S. Yu, Zhifeng Hao, and Ann B. Ragin. Tensor-based multi-view feature selection with applications to brain diseases. In *ICDM*, pages 40–49. IEEE, 2014.

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

[4] Joseph E Gonzalez, Reynold S Xin, Ankur Dave, Daniel Crankshaw, Michael J Franklin, and Ion Stoica. GraphX: Graph processing in a distributed dataflow framework. In *OSDI*, pages 599–613. USENIX, 2014.

[5] Liangjie Hong, Aziz S Doumith, and Brian D Davison. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *WSDM*, pages 557–566. ACM, 2013.

[6] Yu-Chin Juan, Yong Zhuang, and Wei-Sheng Chin. *LIBFFM: A Library for Field-aware Factorization Machines*, 2015. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libffm`.

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

[8] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434. ACM, 2008.

[9] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

[10] Steffen Rendle. Factorization machines. In *ICDM*, pages 995–1000. IEEE, 2010.

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

[12] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90. ACM, 2010.

[13] Dacheng Tao, Xuelong Li, Weiming Hu, Stephen Maybank, and Xindong Wu. Supervised tensor learning. In *ICDM*, pages 8–pp. IEEE, 2005.

[14] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2000.

[15] Cong Xie, Ling Yan, Wu-Jun Li, and Zhihua Zhang. Distributed power-law graph computing: Theoretical and empirical analysis. In *NIPS*, pages 1673–1681, 2014.

[16] Ling Yan, Wu-jun Li, Gui-Rong Xue, and Dingyi Han. Coupled group lasso for web-scale CTR prediction in display advertising. In *ICML*, pages 802–810, 2014.

[17] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, pages 2–2. USENIX, 2012.