Deep Multi-View Concept Learning

Cai Xu†, Ziyu Guan†*, Wei Zhao†, Yunfei Niu†, Quan Wang‡, Zhiheng Wang‡
†State Key Lab of ISN, School of Computer Science and Technology, Xidian University
‡School of Computer Science and Technology, Xidian University
§College of Computer Science and Technology, Henan Polytechnic University
{cxu_3@stu., zyguan@, ywzhao@mail., yfniu@stu., qwang@, zhiheng@xidian.edu.cn, wzhenry@eyou.com

Abstract

Multi-view data is common in real-world datasets, where different views describe distinct perspectives. To better summarize the consistent and complementary information in multi-view data, researchers have proposed various multi-view representation learning algorithms, typically based on factorization models. However, most previous methods were focused on shallow factorization models which cannot capture the complex hierarchical information. Although a deep multi-view factorization model has been proposed recently, it fails to explicitly discern consistent and complementary information in multi-view data and does not consider conceptual labels. In this work we present a semi-supervised deep multi-view factorization method, named Deep Multi-view Concept Learning (DMCL). DMCL performs nonnegative factorization of the data hierarchically, and tries to capture semantic structures and explicitly model consistent and complementary information in multi-view data at the highest abstraction level. We develop a block coordinate descent algorithm for DMCL. Experiments conducted on image and document datasets show that DMCL performs well and outperforms baseline methods.

1 Introduction

Multi-view data is prevalent in many real-world applications. For instance, the same news can be obtained from various language sources; an image can be described by different low level visual features. These views often represent diverse and complementary information of the same data. Integrating multiple views is helpful to boost the performance of data mining tasks. We are concerned with representation learning by synthesizing multi-view data. In recent years, a lot of multi-view representation learning algorithms were proposed based on different techniques (e.g. matrix factorization [Guan et al., 2015; Deng et al., 2015], transfer learning [Xu and Sun, 2012]).

*Corresponding author

As a particularly useful family of techniques in data analysis, matrix factorization is a successful representation learning technique over a variety of areas, e.g. recommendation [Wang et al., 2016a; 2016b], image clustering [Tirges et al., 2017]. Recently, Nonnegative Matrix Factorization (NMF), a specific form of matrix factorization, has received significant attention in multi-view representation learning [Zong et al., 2017; Guan et al., 2015; Liu et al., 2015] due to its intuitive parts-based interpretation [Lee and Seung, 2001]. Given a data matrix \( X \in \mathbb{R}^{D \times N} \) for \( N \) items, NMF seeks two nonnegative matrices \( \mathbf{U} \in \mathbb{R}^{D \times K} \) and \( \mathbf{V} \in \mathbb{R}^{K \times N} \) such that \( X \approx UV \). \( U/V \) is called the basis/encoding matrix. The Multi-view Concept Learning (MCL) algorithm proposed in [Guan et al., 2015] is a typical semi-supervised method which explicitly discards consistent and complementary information in multi-view data to generate conceptual representations. However, a common draw-
back of the above methods is that they fail to capture complex hierarchical structures of real-world data.

In order to learn a better representation by capturing the hierarchical structures, different deep matrix factorization techniques were proposed recently. Trigeorgis et al. [Trigeorgis et al., 2017] proposed the Deep Semi-NMF method for representation learning. It has an interpretation of clustering according to different attributes of a given dataset. Nevertheless, it only deals with the single view case. Zhao et al. [Zhao et al., 2017] extended Deep Semi-NMF to the multi-view case. Although it is a multi-view deep factorization method, it neither considers label information of data nor explicitly models consistent and complementary information.

In this paper, we propose a new multi-view deep factorization method, named Deep Multi-view Concept Learning (DMCL). As shown in Figure 1, the deep model factorizes the data matrices \( \{X^{(v)}\}_{v=1}^V \) iteratively to get the final representation \( V_M \). For each view, each layer is a NMF which takes the representation obtained from the previous layer as its data matrix. The final representation \( V_M \) is shared across different views. We impose graph embedding regularization [Yan et al., 2007] on \( V_M \) to capture data conceptual structures. We also require the basis matrices \( \{U^{(v)}_M\} \) of the final layer to be sparse in term of columns to explicitly model consistency and complementarity among different views.

The major contribution of this work is a novel semi-supervised deep NMF method for conceptual representation learning from multi-view data. Conceptual features can reflect semantic relationships between data items; they are connected to different views in a flexible fashion, i.e. some conceptual features are described by all the views (consistency), while others may only be associated with some of the views (complementarity). We design the optimization problem to encourage \( V_M \) and \( \{U^{(v)}_M\} \) to comply with these properties. The second contribution is that we propose a block coordinate decent algorithm to optimize DMCL. We also design a pre-training scheme for DMCL. Thirdly, we empirically evaluate DMCL on two real world datasets and show its superiority over state-of-art baseline methods.

## 2 Related Work

Our work falls into the area of multi-view representation learning, which is concerned with how to embed inputs from different views of the same set of data items to a new common latent space for better data representation. A recent survey for this area can be found in [Li et al., 2016]. One direction stemmed from Canonical Correlation Analysis (CCA) [Chaudhuri et al., 2009] is based on the principle of maximizing correlations in the common latent space. However, it is nontrivial to extend those methods to deal with multiple views. Another popular idea is to find a shared latent representation across different views. Many methods in this direction were based on (nonnegative) matrix factorization, e.g. [Zong et al., 2017; Liu et al., 2015; Guan et al., 2015]. However, researchers were mainly focused on consistency among different views, while complementarity is rarely explicitly modeled. There were some works that explicitly consider complementarity. Guan et al. proposed a NMF-based flexible method where group sparsity constraints were imposed on basis matrices to learn flexible association patterns between encoding dimensions and views [Guan et al., 2015].

Nevertheless, traditional models are intrinsically shallow models and may not well handle intricate natural data. Inspired by deep learning [Bengio and others, 2009], different deep models were proposed recently for multi-view representation learning. Srivastava and Salakhutdinov [Srivastava and Salakhutdinov, 2012] proposed to learn joint representation of images and texts by Deep Boltzmann Machines. Ngiam et al. [Ngiam et al., 2011] explored extracting shared representations by training a bimodal deep autoencoder. Deep matrix factorization techniques that factorize complex natural data into multiple levels of factors will also increase representational and modeling power [Sharma et al., 2017; Trigeorgis et al., 2017]. Those methods can be viewed as a decoder network that produces a reconstruction \( X = g(V_M) \). Compared to deep matrix factorization, deep neural networks are harder to approximate global optima and lack interpretability. A closely related work is [Zhao et al., 2017] where multi-layer matrix factorization is performed for multi-view data clustering. Our DMCL is different from their method in that we not only incorporate label information but also explicitly learn consistency and complementarity among multiple views, trying to capture conceptual features hidden in the data.

## 3 The Method

Our DMCL is a deep extension of the MCL method. In this section, we review MCL briefly and then present DMCL, together with its optimization algorithm.

### 3.1 A Brief Review of MCL

We use \( X^{(v)} \in \mathbb{R}^{D_v \times N} \) to denote the \( v \)-th view of data, where \( D_v \) is the dimensionality of the \( v \)-th view. The dataset is described by \( H \) views: \( \{X^{(v)}\}_{v=1}^H \). The basis matrix \( U^{(v)} \in \mathbb{R}^{D_v \times K} \) denotes the linear connection between \( X^{(v)} \) and \( V \), the common encoding matrix. The data matrix of each view is separated into labeled and unlabeled parts: \( X^{(v)} = [X^{(v), L} \; X^{(v), U}] \). Correspondingly, the encoding matrix becomes \( V = [V^{L} \; V^{U}] \). We use \( N^{L}/N^{U} \) to denote the number of labeled/unlabeled items, respectively. The optimization problem of MCL is formulated as [Guan et al., 2015]

\[
\min_{\{U^{(v)}\}_{v=1}^V, V} \frac{1}{2} \sum_{v=1}^H \|X^{(v)} - U^{(v)}_L V\|^2_F + \alpha \sum_{v=1}^H \|U^{(v)}\|_{1,\infty} + \\
\frac{\beta}{2} \left\{ tr\left[ V^{L} L^{P} (V^{L})^T \right] - tr\left[ V^{L} L^{P} (V^{L})^T \right] \right\} + \\
\gamma \|V\|_{1,1} \\
\text{s.t. } U^{(v)}_{ik} \geq 0, 1 \geq V_{kj} \geq 0, \forall i, j, k, v.
\]

The upper bound 1 for \( V_{kj} \) is used to guarantee the problem is well lower bounded [Guan et al., 2015]. The first term is the reconstruction criterion. The second term contains the group
sparseres constraints imposed on the basis matrix of each view. For view $v_i$, it is defined as

$$
\|U^{(v)}\|_{1,\infty} = \max_{1 \leq i \leq M} |T^{(v)}_{ik}|
$$

(2)

It means we encourage some basis vectors of $U^{(v)}$ to be completely 0, so that the corresponding dimensions in $V$ are not associated with this view. These dimensions could represent complementary information in multi-view data.

The third term is the graph embedding criterion for regularizing $V$ where $tr(\cdot)$ denotes matrix trace. $L^a$ and $L^p$ represent the graph Laplacian matrices of within-class affinity graph $G^a$ and between-class penalty graph $G^p$ with their weighted adjacency matrices defined as

$$
W^a_{ij} = \left\{ \begin{array}{ll}
\frac{1}{N_i} - \frac{1}{N^t}, & \text{if } c_i = c_j \\
0, & \text{otherwise}
\end{array} \right.
$$

and

$$
W^p_{ij} = \left\{ \begin{array}{ll}
\frac{1}{N_i}, & \text{if } c_i \neq c_j \\
0, & \text{otherwise}
\end{array} \right.
$$

where $c_i$ denotes the label of item $i$, $N^t_i$ is the total number of items with label $c_i$. The graph embedding term intrinsically forces within-class items to be near while keeps between-class items away from each other. With simple algebra transformation, we have

$$
\sum_{i,j} W^a_{ij} |V^i - V^j|^2_2 = tr[V^T L^a (V^i)^T]
$$

and

$$
\sum_{i,j} W^p_{ij} |V^i - V^j|^2_2 = tr[V^T L^p (V^i)^T].
$$

The fourth term is a simple $L_1$ norm regularizer on $V$ since an item should not have too many conceptual features.

MCL not only tries to capture semantic structures of the data by semi-supervision, but also models consistency and complementarity among different views by group sparsity constraints. However, the shallow computation (i.e. one-step factorization) in MCL may not be able to well handle complex real world data.

### 3.2 Deep Multi-view Concept Learning

In order to obtain a more expressive representation, DML decomposes each of the data matrices $\{X^{(v)}\}_{v=1}^H$ iteratively to obtain the high-level representation:

$$
X^{(v)} \approx U_1^{(v)} U_2^{(v)} \cdots U_M^{(v)} V_M
$$

where $U_1^{(v)} \in \mathbb{R}_{+}^{D_v \times P_{1}}$, $U_2^{(v)} \in \mathbb{R}_{+}^{P_{1} \times P_2}$, $\ldots$, $U_M^{(v)} \in \mathbb{R}_{+}^{P_{M-1} \times P_M}$ denote $M$ basis matrices and $V_M \in \mathbb{R}_{+}^{P_M \times N}$ denotes the final common encoding. The optimization problem of DML is

$$
\min_{\{U^{(v)}_{m}\}, V_M} \frac{1}{2} \sum_{v=1}^{H} \left\| X^{(v)} - U_1^{(v)} U_2^{(v)} \cdots U_M^{(v)} V_M \right\|_F^2
$$

$$
+ \beta \frac{1}{2} \left\{ tr \left[ V_M^T L^a (V_M^i)^T \right] - tr \left[ V_M^T L^p (V_M^i)^T \right] \right\}
$$

$$
+ \alpha \sum_{v=1}^{H} \left\| U^{(v)}_{m} \right\|_{1,\infty} + \gamma \left\| V_M \right\|_{1,1}
$$

s.t. $(U_{m}^{(v)})_{ik} \geq 0, 1 \geq (V_M)_{kj} \geq 0, \forall i,j,k,v,m$.

(6)

Here we only apply the graph embedding constraints and the encoding sparseness constraint to $V_M$, since the intermediate encodings are near low-level features, so they would not well represent high-level conceptual features. The group sparsity constraints used to learn the structures of consistency and complementarity are only imposed on $\{U^{(v)}_{m}\}$, since they can only be used where a common encoding $(V_M)$ is reached. Note although the overall factorization is equivalent to a linear operation, as in [Zhao et al., 2017], the multi-layer computation can still help to better represent data items hierarchically by seeking a better local optimum [Zhao et al., 2017].

### 3.3 Optimization

(6) is not convex in both $\{U^{(v)}_{m}\}$ and $V_M$. Therefore, we can only find its local minima. To improve the quality of the solution and speedup learning, we initialize the model parameters using unsupervised greedy pre-training, similar to layerwise pre-training in deep learning [Hinton and Salakhutdinov, 2006]. Specifically, for each view $v$ we first decompose $X^{(v)}$ as $X^{(v)} = U_1^{(v)} V_1^{(v)}$ using NMF. Then, we treat the learned $V_1^{(v)}$ as the “data matrix” for layer 2 and continue to factorize it iteratively until the final layer. An exception is $V_M$. For layer $M$, we obtain a set of encoding matrices $\{V^{(v)}_M\}$ from the above initialization scheme. However, it is difficult to use them to initialize $V_M$ since elements in the same position of the encoding vectors for an item may represent different meanings in different views and so they are not comparable. Hence, we choose to initialize $V_M$ randomly. Preliminary experiments also confirmed its effectiveness.

Afterwards, the variables of (6) are separated into three groups: $\{U^{(v)}_{m}\}_{m \neq M}$, $\{U^{(v)}_{m}\}_{v=1}^H$ and $V_M$. (6) is convex in one group when the other two are fixed. Therefore, we solve DML by block coordinate descent [Lin, 2007] which each time optimizes one group of variables while keeping the other groups fixed. The procedure is depicted in Algorithm 1. At line 2, we have $V_0^{(v)} := X^{(v)}$. Next, we describe the detailed ideas for addressing the three subproblems.

#### Updating $V_M$

Since $V_M$ is randomly initialized, we optimize it firstly. The subproblem for $V_M$ is:

$$
\min_{V_M} \psi(V_M) := \sum_{v=1}^{H} \left\| X^{(v)} - U^{(v)} M V\right\|_F^2 + \gamma \left\| V_M \right\|_{1,1}
$$

$$
+ \frac{\beta}{2} \left\{ tr \left[ V_M^T L^a (V_M^i)^T \right] - tr \left[ V_M^T L^p (V_M^i)^T \right] \right\}
$$

s.t. $1 \geq (V_M)_{kj} \geq 0, \forall k,j.$

(7)

where $\hat{U}^{(v)}_M = \prod_{m=1}^{M} U^{(v)}_{m}$.

We can decompose (7) into two subproblems in which the variables are $U^{(v)}_M$ and $V_M$, labeled part and unlabeled part of $V_M$, respectively. The update rules can be similarly derived as in [Guan et al., 2015]. Here we only give the equations

\[\text{where }\]
Algorithm 1: Optimization of DMCL

Input: \( \{X^{(v)}\}_{v=1}^{H}; \alpha; \beta; \gamma; L^p; L^q \); layer sizes \( \{p_m\} \)

Output: \( \{U^{(v)}\}_{v=1}^{H}; \forall v, m; V_M \)

1. for \( m = 1 \) to \( M \), \( v = 1 \) to \( H \) do
2. \( U^{(v)}; V^{(v)} \leftarrow \text{NMF}(V^{(v)}_{m-1}; p_m) \)
3. end
4. Randomly initialize \( V_M \)
5. repeat
6. Fix other variables, optimize (7) w.r.t \( V_M \).
7. Fix other variables, optimize (15) w.r.t \( \{U^{(v)}\}_{v=1}^{H}; m \neq M \).
8. Fix other variables, optimize (18) w.r.t \( \{U^{(v)}\}_{v=1}^{H}; m = M \).
9. until convergence or max no. iterations reached

due to space limitation:

\[
(V^{l}_M)_{kj} \leftarrow \min \left\{ 1, \frac{-B_{kj} + \sqrt{B_{kj}^2 + 4A_{kj}C_{kj}}}{2A_{kj}} (V^{l}_M)_{kj} \right\}
\]

(8)

\[
(V^{u}_M)_{kj} \leftarrow \min \left\{ 1, \frac{-(\gamma - Q^{l}_k) + |\gamma - Q^{l}_k|}{2(P^u v^{l}_j)} (V^{u}_M)_{kj} \right\}
\]

(9)

where \( P, Q^u, A_{kj}, B_{kj} \) and \( C_{kj} \) are

\[
P = \sum_{v=1}^{H} (U^{(v)}_{M})^T \tilde{U}^{(v)}_M
\]

(10)

\[
Q^u = \sum_{v=1}^{H} (U^{(v)}_{M})^T X^{(v)} u
\]

(11)

\[
A_{kj} = (P^v v^{l}_j)_k + \beta ((D^u + W^p)^v v^{l}_j)_k
\]

(12)

\[
B_{kj} = \gamma - Q^{l}_k
\]

(13)

\[
C_{kj} = \beta ((D^p + W^u)^v v^{l}_j)_k
\]

(14)

Here \( v^{l}_j \) and \( v^{u}_j \) denote the \( j \)-th column vector and the \( k \)-th row vector of \( V^{l}_M \), respectively. \( D^u \) and \( D^p \) are diagonal matrices with \( D^u_{ii} = \sum_{j=1}^{N^l} W^u_{ij}, D^p_{ii} = \sum_{j=1}^{N^l} W^p_{ij} \).

Updating \( \{U^{(v)}\}_{v=1}^{H}; m \neq M \)

Fixing other variables, we get the subproblem for \( \{U^{(v)}\} \)

\[
\min_{\{U^{(v)}\}_{v=1}^{H}; m \neq M} \Gamma = \frac{1}{2} \sum_{v=1}^{H} \|X^{(v)} - U^{(v)}_1 U^{(v)}_2 \cdots U^{(v)}_M V_M\|_F^2
\]

s.t. \( (U^{(v)})_{ik} \geq 0, \forall i, k, v; m = 1, \cdots, M - 1 \).

(15)

For \( m = 1 \), the updating is similar as in NMF. Otherwise, the gradient of \( U^{(v)}_m \) is derived as:

\[
\frac{\partial \Gamma}{\partial U^{(v)}_m} = (C^{(v)}_m)^T (\xi^{(v)}_m U^{(v)}_m \phi^{(v)}_m - \phi^{(v)}_m)^T X^{(v)} \phi^{(v)}_m - \phi^{(v)}_m)^T + \frac{\partial \Gamma}{\partial U^{(v)}_m}
\]

where \( \xi^{(v)}_m = \prod_{i=1}^{m-1} U^{(v)}_i \) and \( \phi^{(v)}_m = \prod_{i=m+1}^{M} U^{(v)}_i V_M \).

Thus the additive update for \( U^{(v)}_m \) can be given as:

\[
(U^{(v)}_m)_{ik} \leftarrow (U^{(v)}_m)_{ik} - \eta \left( \frac{\partial \Gamma}{\partial U^{(v)}_m} \right)_{ik}
\]

(16)

Inspired by [Lee and Seung, 2001], to obtain a multiplicative update rule, \( \eta \) is set as \( \frac{(\xi^{(v)}_m)^T X^{(v)} \phi^{(v)}_m}{(\xi^{(v)}_m)^T \phi^{(v)}_m (\phi^{(v)}_m)^T} \).

Meanwhile, the convergence is guaranteed. Correspondingly, the multiplicative update rule is:

\[
(U^{(v)}_m)_{ik} \leftarrow \left( (U^{(v)}_m)_{ik} \left( \frac{(\xi^{(v)}_m)^T X^{(v)} \phi^{(v)}_m}{(\xi^{(v)}_m)^T \phi^{(v)}_m (\phi^{(v)}_m)^T} \right)_{ik} \right)
\]

(17)

Updating \( \{U^{(v)}\}_{v=1}^{H}; m = M \)

When other variables are fixed, the \( U^{(v)}_m \) of different views are independent with each other and their subproblems are identical. For clarity, we just focus on one view and omit the superscript \( (v) \) temporally:

\[
\min_{U_M} \phi(U_M) := \frac{1}{2} \|X - \tilde{U}_{M-1} U_M V_M\|_F^2 + \alpha \|U_M\|_{1,\infty}
\]

s.t. \( (U_M)_{ik} \geq 0, \forall i, k, M \).

\[
\phi(U_M) \text{ is the sum of a differentiable function and a general closed convex function. It can be solved using composite gradient mapping [Nesterov, 2013] which was proposed for minimizing such composite functions. The central idea is to iteratively minimize an auxiliary function } m_L(U_M) \text{ and adjust the guess of the Lipschitz constant of the first term of } \phi(U_M) \text{ so that } \phi(U_M) \text{ decreases by the minimizer of } m_L(U_M). \text{ Denote } f(U_M) = \frac{1}{2} \|X - \tilde{U}_{M-1} U_M V_M\|_F^2 \text{ and } U^*_M \text{ as the value of } U_M \text{ at the } t \text{-th iteration. The auxiliary function is set as}
\]

\[
m_L(U_M; U_M) = f(U_M) + \alpha \|U_M\|_{1,\infty}
\]

\[
+ \frac{L}{2} \|U_M - U_t^M\|_F^2 + tr( \nabla f(U_M^T)^T (U_M - U_t^M) )
\]

(19)

where \( L \) is the guess of \( L_f \), the Lipschitz constant of \( f(U_M) \), and \( \nabla f(U_M) \) is the gradient of \( f(U_M) \) at \( U_t^M \):

\[
\nabla f(U_t^M) = \tilde{U}_{M-1} V_M (U_M^T - \tilde{U}_{M-1} V_M)^T
\]

Then we minimize (19) to get a candidate for \( U_t^{M+1} \):

\[
\tilde{U}_{M+1} = \arg \min_{U_{M+1}} m_L(U_{M+1}; U_M)
\]

(20)

We have \( \phi(U_t^M) = m_L(U_t^M: U_t^M) \geq m_L(U_{M+1}; U_t^{M+1}) \) from (20). Furthermore, it has been proved that for \( L \geq L_f \)
### Algorithm 2: Composite Gradient Mapping

**Input:** $\eta_u > 1, \eta_d > 1$: scaling parameters for $L$

1. **begin**
2. Initialize $L_0 : 0 < L_0 \leq L_f$.
3. $t = 0$
4. **repeat**
   5. $L = L_t$
   6. Optimize (20) to get $\hat{U}^{t+1}_M$
   7. **if** $\phi(\hat{U}^{t+1}_M) > m_L(U^{t+1}_M; \hat{U}^{t+1}_M)$ **then**
   8. $L = L_t \eta_u$
   9. **end**
10. **until** $\phi(\hat{U}^{t+1}_M) \leq m_L(U^{t+1}_M; \hat{U}^{t+1}_M)$
11. $U^{t+1}_M = \hat{U}^{t+1}_M$
12. $L_{t+1} = \max(L_0, L_t / \eta_d)$
13. $t = t + 1$
14. **until convergence**
15. **end**

---

we have $m_L(U^{t+1}_M; \hat{U}^{t+1}_M) \geq \phi(\hat{U}^{t+1}_M)$ [Nesterov, 2013], leading to an acceptable $U^{t+1}_M$. Meanwhile, $L$ is inversely proportional to the step size (i.e. $\|\hat{U}^{t+1}_M - U^{t+1}_M\|$), so it cannot be set to too large values. Now the problem is to find a suitable $L$ in each iteration. We starts with an estimate $L_0$ such that $0 < L_0 \leq L_f$, and in each iteration adjusts $L$ until we get $\phi(\hat{U}^{t+1}_M) \leq m_L(U^{t+1}_M; \hat{U}^{t+1}_M)$. The algorithm for optimizing $U_M$ is shown in Algorithm 2.

The remaining problem is how to optimize (20) efficiently. The solution is similar to that in [Guan et al., 2015]. We omit the details due to space limitation.

### 3.4 Time Complexity

The DMCL model is composed of pre-training and fine-turning stages, so we analyze them separately. For simplifying, we set the input feature dimensionalities for all views to $D$; the dimensionalities of all layers are set to $p$. We use $\tau$ to denote the number of iterations for all iterative procedures.

The computational cost for the pre-training stage is $O(\tau H (DNp + MNp^2))$. For fine-turning, the time complexity consists of three subparts. For optimizing $\{U^{(v)}_m\}_{m \neq M}$, the cost is $O(\tau HM(DNp^2 + NP^2 + DNP + MP^3))$. For optimizing $\{U^{(v)}_M\}_{v = 1}$, we need to run Algorithm 2. Here we give the result: $O(HD((2(\tau + 1) + \log_2 L_t / \eta_U)P + \tau K + 2 + MP^3 + DP^2))$. Here $\tau$ denotes the number iterations of the outer loop of Algorithm 2. For $V_M$, the time complexity is $O(\tau (NP + NP^2 + (N^2)p^2) + H(MP^3 + DP^2))$. The time cost of DMCL is linear in feature dimension $D$, item number $N$ and view number $H$. However, it it not linear in layer number $M$ and layer size $p$. Therefore, it is essential to set layer number and layer sizes of DMCL properly for high performance and low time consumption.

### 4 Experiments

We evaluate the performance of DMCL on document and image datasets in terms of classification and clustering metrics. Important statistics are summarized in Table 1 and a brief introduction of the datasets is presented below.

**Reuters** [Amini et al., 2009]. It consists of 111740 documents written in five languages of 6 categories represented as TFIDF vectors. We utilize documents written in English and Italian as two views. For each categories, we randomly choose 200 documents. Totally 1200 documents are used.

**ImageNet** [Deng et al., 2009]. It is a well known real-world image database that contains roughly 15 million images organized according to the WordNet hierarchy. We randomly select 50 leaf synsets in the hierarchy as categories and sample 240 images from each one. Three kinds of features are extracted as different views, i.e., 64D HSV histogram, 512D GIST descriptors, and 1000D bag of SIFT visual words.

We compare DMCL with the following baseline algorithms: Deep NMF (DNMF) [Song et al., 2015] is a deep matrix factorization method for single view data. We apply it on each view and report the best performance. Concatenation NMF (ConcatNMF) concatenates feature vectors of different views and then applies DNMF. Deep Multiview Semi-NMF (DMSNMF) [Zhao et al., 2017] is an unsupervised matrix factorization method synthesizing multiview data to capture a uniform representation. Multi-view Concept Learning (MCL) [Guan et al., 2015] is a semi-supervised shallow method for multi-view data.

We use the holdout method [Han et al., 2011] for evaluation and tune model parameters by cross-validation on the training set. For each dataset, we randomly split the data items for each category and use 50% for training while the remaining 50% are reserved for test. We use the learned representation of these methods for classification and clustering. Note that the label information of the training set is utilized in semi-supervised methods, i.e., MCL and DMCL. For classification, the training items are fed into the kNN classifier ($k=9$) and Accuracy is calculated using the test set. For clustering, k-means is applied to the test set with $k$ set to the actual number of classes. Accuracy and Normalized Mutual Information (NMI) are used to evaluate clustering performance [Trigeorgis et al., 2017]. To account for runtime randomness in evaluation, we run each test case 10 times and calculate the averaged performance and standard derivation.

#### 4.1 Results

Table 2 and Figure 2 show the classification and clustering performance of DMCL and baseline methods. First, DNMF is the worst. It is outperformed by all the other methods. This reveals the importance of using multiple views. Second, semi-supervised methods outperforms unsupervised.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th># of categories</th>
<th>Dimensionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reuters</td>
<td>1200</td>
<td>6</td>
<td>21536/15506</td>
</tr>
<tr>
<td>ImageNet</td>
<td>12000</td>
<td>50</td>
<td>64/512/1000</td>
</tr>
</tbody>
</table>

Table 1: Dataset summary.
This is intuitive since by exploiting the partial label information, we can build a more discriminative representation.

Third, our proposed DMCL consistently outperforms MCL on the two datasets, which indicates that the deep model could generate better representation by hierarchical modeling. We use t-test with significance level 0.05 to test the significance of performance difference. Results show that DMCL significantly outperforms all the baseline methods.

### 4.2 Analysis

In this subsection, we will analyze DMCL from two perspectives, i.e., parameter setting and convergence analysis.

#### Parameter analysis

Parameters of DMCL include $\alpha$, $\beta$, $\gamma$, and layer number and sizes. Here we explore their impact to performance by cross-validation on the training set and only show representative results due to space limitation. We first focus on the former three parameters. $\alpha$ and $\gamma$ control the degree of sparseness, while $\beta$ controls the impact of the graph embedding regularization. We vary one parameter each time and fix the other two to explore its influence. Fig 3 shows the results on Reuters. We find a general pattern: the performance curves first go up and then go down when increasing the parameters. This means the sparseness and graph embedding terms are useful for learning good representations. Based on the results, we set $\alpha = 100$, $\beta = 0.015$, and $\gamma = 0.005$ in other experiments.

Regarding layer sizes, previous work on multi-view deep factorization [Zhao et al., 2017] has found that $p_M$, the size of the final layer, usually plays a more important role than the sizes of the other layers. Hence, we vary $p_M$ under different layer numbers and fix the sizes of the other layers empirically. Fig 4 shows the NMI results on ImageNet under 3 layer number settings. The full settings are $\{200 p_M, 300 200 p_M, 500 300 200 p_M\}$. As can be seen, the performance increases with the layer number, which indicates deep factorization really helps to find better representations. Considering both performance and efficiency, we choose 3 layers for all the experiments. The layer sizes are set to $[300 200 125]$.

#### Convergence analysis

Fig 5 shows the curves of objective function value and NMI against the number of iterations for DMCL and MCL. We find at the beginning the objective function value drops and the performance increases rapidly. The optimization procedure of DMCL typically converges in around 40 iterations on the ImageNet dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>ImageNet</th>
<th>Reuters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNMF</td>
<td>17.66±0.75</td>
<td>59.75±1.65</td>
</tr>
<tr>
<td>ConcatDNMF</td>
<td>25.28±0.81</td>
<td>61.25±1.93</td>
</tr>
<tr>
<td>DMSNMF</td>
<td>27.39±0.76</td>
<td>64.53±1.72</td>
</tr>
<tr>
<td>MCL</td>
<td>30.31±0.62</td>
<td>70.85±1.53</td>
</tr>
<tr>
<td>DMCL</td>
<td>32.41±0.67</td>
<td>73.17±1.61</td>
</tr>
</tbody>
</table>

Table 2: Classification performance on different datasets (accuracy±std dev,%).
by group sparseness constraints. Experimental results on document/image datasets for both classification and clustering tasks confirmed the effectiveness of DMCL compared to competitive multi-view factorization models.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant Nos. 61522206, 61373118, 61672409), the Major Basic Research Project of Shaanxi Province (Grant No. 2017ZDJC-31), the Science and Technology Plan Program in Shaanxi Province of China (Grant No. 2017KJXX-80), the Fundamental Research Funds for the Central Universities, and the Innovation Fund of Xidian University.

References


[Han et al., 2011] Jiawei Han, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.


