A probabilistic framework for multi-view feature learning with many-to-many associations via neural networks

Akifumi Okuno [1 2] **Tetsuya Hada** [3] **Hidetoshi Shimodaira** [1 2]

Abstract

A simple framework Probabilistic Multi-view Graph Embedding (PMvGE) is proposed for multi-view feature learning with many-to-many associations so that it generalizes various existing multi-view methods. PMvGE is a probabilistic model for predicting new associations via graph embedding of the nodes of data vectors with links of their associations. Multi-view data vectors with many-to-many associations are transformed by neural networks to feature vectors in a shared space, and the probability of new association between two data vectors is modeled by the inner product of their feature vectors. While existing multi-view feature learning techniques can treat only either of many-to-many association or non-linear transformation, PMvGE can treat both simultaneously. By combining Mercer's theorem and the universal approximation theorem, we prove that PMvGE learns a wide class of similarity measures across views. Our likelihood-based estimator enables efficient computation of non-linear transformations of data vectors in large-scale datasets by minibatch SGD, and numerical experiments illustrate that PMvGE outperforms existing multi-view methods.

1. Introduction

With the rapid development of Internet communication tools in past few decades, many different types of data vectors become easily obtainable these days, advancing the development of multi-view data analysis methods (Sun, 2013; Zhao et al., 2017). Different types of vectors are called as "views", and their dimensions may be different depending on the view. Typical examples are data vectors of images, text tags, and user attributes available in Social Networking Services (SNS). However, we cannot apply standard data analysis methods, such as clustering, to multi-view data vectors, because data vectors from different views, say, images and text tags, are not directly compared with each other. In this paper, we work on multi-view Feature Learning for transforming the data vectors from all the views into new vector representations called "feature vectors" in a shared euclidean subspace.

Table 1. Comparison of PMvGE with existing multi-view / graph-embedding methods. (Nv): Number of views, (MM): Many-to-many, (NL): Non-linear, (Ind): Inductive, (Lik): Likelihood-based. PMvGE has all the properties. Nv $= D$ represents that the method can deal with arbitrary number of views.

	(Nv)	(MM)	(NL)	(Ind)	(Lik)
CCA	2			✓	
DCCA	2		✓	✓	
MCCA	D			✓	
SGE	0	✓			
LINE	0	✓			✓
LPP	1	✓		✓	
CvGE	2	✓		✓	
CDMCA	D	✓		✓	
DeepWalk	0	✓			✓
SBM	1	✓		✓	✓
GCN	1	✓	✓		✓
GraphSAGE	1	✓	✓	✓	✓
IDW	1	✓	✓	✓	✓
PMvGE	D	✓	✓	✓	✓

One of the best known approaches to multi-view feature learning is Canonical Correlation Analysis (Hotelling, 1936, CCA) for two-views, and Multiset CCA (Kettenring, 1971, MCCA) for many views. CCA considers pairs of related data vectors $\{(\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)})\}_{i=1}^n \subset \mathbb{R}^{p_1} \times \mathbb{R}^{p_2}$. For instance, $\boldsymbol{x}_i^{(1)} \in \mathbb{R}^{p_1}$ may represent an image, and $\boldsymbol{x}_i^{(2)} \in \mathbb{R}^{p_2}$ may represent a text tag. Their dimension p_1 and p_2 may be different. CCA finds linear transformation matrices $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}$ so that the sum of inner products $\sum_{i=1}^n \langle \boldsymbol{A}^{(1)\top} \boldsymbol{x}_i^{(1)}, \boldsymbol{A}^{(2)\top} \boldsymbol{x}_i^2 \rangle$ is maximized under a variance constraint. The obtained linear transformations compute feature vectors $\boldsymbol{y}_i^{(1)} := \boldsymbol{A}^{(1)\top} \boldsymbol{x}_i^{(1)}, \boldsymbol{y}_i^{(2)} :=$

[1]Graduate School of Informatics, Kyoto University, Kyoto, Japan [2]RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan [3]Recruit Technologies Co., Ltd., Tokyo, Japan. Correspondence to: Akifumi Okuno <okuno@sys.i.kyoto-u.ac.jp>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

$\boldsymbol{A}^{(2)\top}\boldsymbol{x}_i^{(2)} \in \mathbb{R}^K$ where $K \leq \min\{p_1, p_2\}$ is the dimension of the shared space of feature vectors from the two views. However, the linear transformations may not capture the underlying structure of real-world datasets due to its simplicity.

To enhance the expressiveness of transformations, CCA has been further extended to non-linear settings, as Kernel CCA (Lai and Fyfe, 2000) and Deep CCA (Andrew et al., 2013; Wang et al., 2016, DCCA) which incorporate kernel methods and neural networks to CCA, respectively. These methods show drastic improvements in performance in face recognition (Zheng et al., 2006) and image-text embedding (Yan and Mikolajczyk, 2015). However, these CCA-based approaches are limited to multi-view data vectors with one-to-one correspondence across views.

Real-world datasets often have more complex association structures among the data vectors, thus the whole dataset is interpreted as a large graph with nodes of data vectors and links of the associations. For example, associations between images $\{\boldsymbol{x}_i^{(1)}\}_{i=1}^{n_1}$ and their tags $\{\boldsymbol{x}_j^{(2)}\}_{j=1}^{n_2}$ may be many-to-many relationships, in the sense that each image has multiple associated tags as well as each tag has multiple associated images. The weight $w_{ij} \geq 0$, which we call "link weight", is defined to represent the strength of association between data vectors $\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_j^{(2)}$ $(i = 1, 2, \ldots, n_1; j = 1, 2, \ldots, n_2)$. The number of data vectors n_1, n_2 in each view may be different.

To fully utilize the complex associations represented by $\{w_{ij}\}$, Cross-view Graph Embedding (Huang et al., 2012, CvGE) and its extension to more than three views called Cross-Domain Matching Correlation Analysis (Shimodaira, 2016, CDMCA) are proposed recently, by extending CCA to many-to-many settings. 2-view CDMCA (= CvGE) obtains linear transformation matrices $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}$ so that the sum of inner products $\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} w_{ij}\langle \boldsymbol{A}^{(1)\top}\boldsymbol{x}_i^{(1)}, \boldsymbol{A}^{(2)\top}\boldsymbol{x}_j^{(2)}\rangle$ is maximized under a variance constraint.

CDMCA includes various existing multi-view / graph-embedding methods as special cases. For instance, 2-view CDMCA obviously includes CCA as a special case $w_{ij} = \delta_{ij}$, where δ_{ij} is Kronecker's delta. By considering 1-view setting, where $\boldsymbol{x}_i \in \mathbb{R}^p$ is a 1-view data vector and $w_{ij} \geq 0$ is a link weight between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, CDMCA reduces to Locality Preserving Projections (He and Niyogi, 2004; Yan et al., 2007, LPP). LPP also reduces to Spectral Graph Embedding (Chung, 1997; Belkin and Niyogi, 2001, SGE), which is interpreted as "0-view" graph embedding, by letting $\boldsymbol{x}_i \in \{0, 1\}^n$ be 1-hot vector with 1 at i-th entry and 0 otherwise.

Although CDMCA shows a good performance in word and image embeddings (Oshikiri et al., 2016; Fukui et al., 2016), its expressiveness is still limited due to its linearity. There has been a necessity of a framework, which can deal with many-to-many associations and non-linear transformations simultaneously. Therefore, in this paper, we propose a non-linear framework for multi-view feature learning with many-to-many associations. We name the framework as Probabilistic Multi-view Graph Embedding (PMvGE). Since PMvGE generalizes CDMCA to non-linear setting, PMvGE can be regarded as a generalization of various existing multi-view methods as well.

PMvGE is built on a simple observation: many existing approaches to feature learning consider the inner product similarity of feature vectors. For instance, the objective function of CDMCA is the weighted sum of the inner product $\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle$ of the two feature vectors $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ in the shared space. Turning our eyes to recent 1-view feature learning, Graph Convolutional Network (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017), and Inductive DeepWalk (Dai et al., 2018) assume that the inner product of feature vectors $\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle$ approximates link weight $w_{ij} \geq 0$.

Inspired by these existing studies, for D-view feature learning $(D \geq 1)$, PMvGE transforms data vectors $\boldsymbol{x}_i \in \mathbb{R}^{p_{d_i}}$ from view $d_i \in \{1, \ldots, D\}$ by neural networks $\boldsymbol{x}_i \mapsto \boldsymbol{y}_i := \mathrm{NN}^{(d_i)}(\boldsymbol{x}_i) \in \mathbb{R}^K$ $(i = 1, 2, \ldots, n)$ so that the function $\exp(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle)$ approximates the weight w_{ij} for $i, j = 1, \ldots, n$. We introduce a parametric model of the conditional probability of $\{w_{ij}\}$ given $\{(\boldsymbol{x}_i, d_i)\}$, and thus PMvGE is a non-linear and probabilistic extension of CDMCA. This leads to very efficient computation of the Maximum Likelihood Estimator (MLE) with minibatch SGD.

Our contribution in this paper is summarized as follows:

(1) We propose PMvGE for multi-view feature learning with many-to-many associations, which is non-linear, efficient to compute, and inductive. Comparison with existing methods is shown in Table 1. See Section 2 for the description of these methods.

(2) We show in Section 3 that PMvGE generalizes various existing multi-view methods, at least approximately, by considering the Maximum Likelihood Estimator (MLE) with a novel probabilistic model.

(3) We show in Section 4 that PMvGE with large-scale datasets can be efficiently computed by minibatch SGD.

(4) We prove that PMvGE, yet very simple, learns a wide class of similarity measures across views. By combining Mercer's theorem and the universal approximation theorem, we prove in Section 5.1 that the inner product of feature vectors can approximate arbitrary continuous positive-definite similarity measures via sufficiently large neural networks. We also prove in Section 5.2 that MLE will actu-

ally learn the correct parameter value for sufficiently large number of data vectors.

2. Related works

0-view feature learning: There are several graph embedding methods related to PMvGE without data vectors. We call them as 0-view feature learning methods. Given a graph, Spectral Graph Embedding (Chung, 1997; Belkin and Niyogi, 2001, SGE) obtains feature vectors of nodes by considering the adjacency matrix. However, SGE requires time-consuming eigenvector computation due to its variance constraint. LINE (Tang et al., 2015) is very similar to 0-view PMvGE, which reduces the time complexity of SGE by proposing a probabilistic model so that any constraint is not required. DeepWalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016) obtain feature vectors of nodes by applying skip-gram model (Mikolov et al., 2013) to the node-series computed by random-walk over the given graph, while PMvGE directly considers the likelihood function.

1-view feature learning: Locality Preserving Projections (He and Niyogi, 2004; Yan et al., 2007, LPP) incorporates linear transformation of data vectors into SGE. For introducing non-linear transformation, the graph neural network (Scarselli et al., 2009, GNN) defines a graph-based convolutional neural network. ChebNet (Defferrard et al., 2016) and Graph Convolutional Network (GCN) (Kipf and Welling, 2017) reduce the time complexity of GNN by approximating the convolution. These GNN-based approaches are highly expressive but not *inductive*. A method is called *inductive* if it computes feature vectors for newly obtained vectors which are not included in the training set. GraphSAGE (Hamilton et al., 2017) and Inductive DeepWalk (Dai et al., 2018, IDW) are inductive as well as our proposal PMvGE, but probabilistic models of these methods are different.

Multi-view feature learning: HIMFAC (Nori et al., 2012) is mathematically equivalent to CDMCA. Factorization Machine (Rendle, 2010, FM) incorporates higher-order products of multi-view data vectors to linear-regression. It can be used for link prediction across views. If only the second terms in FM are considered, FM is approximately the same as PMvGE with linear transformations. However, FM does not include PMvGE with neural networks.

Another study Stochastic Block Model (Holland et al., 1983; Nowicki and Snijders, 2001, SBM) is a well-known probabilistic model of graphs, whose links are generated with probabilities depending on the cluster memberships of nodes. SBM assumes the cluster structure of nodes, while our model does not.

3. Proposed model and its parameter estimation

3.1. Preliminaries

We consider an undirected graph consisting of n nodes $\{v_i\}_{i=1}^n$ and link weights $w_{ij} \geq 0 \quad (i,j = 1,2,\ldots,n)$ satisfying $w_{ij} = w_{ji}$ for all i,j, and $w_{ii} = 0$. Let $D \in \mathbb{N}$ be the number of views. For D-view feature learning, node v_i belongs to one of views, which we denote as $d_i \in \{1,2,\ldots,D\}$. The data vector representing the attributes (or side-information) at node v_i is denoted as $\boldsymbol{x}_i \in \mathbb{R}^{p_{d_i}}$ for view d_i with dimension p_{d_i}. For 0-view feature learning, we formally let $D = 1$ and use the 1-hot vector $\boldsymbol{x}_i \in \{0,1\}^n$. We assume that we obtain $\{w_{ij}\}_{i,j=1}^n, \{\boldsymbol{x}_i, d_i\}_{i=1}^n$ as observations. By taking w_{ij} as a random variable, we consider a parametric model of conditional probability of w_{ij} given the data vectors. In Section 3.2, we consider the probability model of w_{ij} with the conditional expected value

$$\mu_{ij} = E(w_{ij} | \{\boldsymbol{x}_i, d_i\}_{i=1}^n)$$

where $\{\boldsymbol{x}_i, d_i\}_{i=1}^n$ is given, for all $1 \leq i < j \leq n$. In Section 3.3, we then define PMvGE by specifying the functional form of μ_{ij} via feature vectors.

3.2. Probabilistic model

For deriving our probabilistic model, we first consider a random graph model with fixed n nodes. At each time-point $t = 1,2,\ldots,T$, an unordered node pair (v_i, v_j) is chosen randomly with probability

$$\mathbb{P}(e_t = (v_i, v_j)) = \frac{\mu_{i'j'}}{\sum_{1 \leq i < j \leq n} \mu_{ij}}$$

where $i' := \min\{i,j\}, j' := \max\{i,j\}$ and e_t represents the undirected link at time t. The parameters $\mu_{ij} \geq 0 \quad (1 \leq i < j \leq n)$ are interpreted as unnormalized probabilities of node pairs. We allow the same pair is sampled several times. Given independent observations $e_1, e_2, \ldots, e_T$, we consider the number of links generated between v_i and v_j as

$$w_{ij} = w_{ji} := \sharp\{t \in \{1,\ldots,T\} \mid e_t = (v_i, v_j)\}.$$

The conditional probability $\mathbb{P}(\{w_{ij}\}_{1 \leq i < j \leq n} \mid T)$ follows a multinomial distribution. Assuming that T obeys Poisson distribution with mean $\sum_{1 \leq i < j \leq n} \mu_{ij}$, the probability of $\{w_{ij}\}_{1 \leq i < j \leq n}$ follows

$$\mathbb{P}(\{w_{ij}\}_{1 \leq i < j \leq n}) = \prod_{1 \leq i < j \leq n} p(w_{ij}; \mu_{ij})$$

where $p(w; \mu)$ is the probability function of Poisson distribution with mean μ. Thus w_{ij} follows Poisson distribution

independently as

$$w_{ij} \overset{\text{indep.}}{\sim} \text{Po}(\mu_{ij}) \tag{1}$$

for all $1 \leq i < j \leq n$. Although w_{ij} should be a nonnegative integer as an outcome of Poisson distribution, our likelihood computation allows w_{ij} to take any nonnegative real value.

Our probabilistic model (1) is nothing but Stochastic Block Model (Holland et al., 1983, SBM) by assuming that node v_i belongs to a cluster $c_i \in \{1, 2, \ldots, C\}$. The model is specified as

$$\mu_{ij} = \beta^{(c_i, c_j)}, \tag{2}$$

where $\beta^{(c_i,c_j)}$ is the parameter regulating the number of links whose end-points belong to clusters c_i and c_j. Note that SBM is interpreted as 1-view method with 1-hot vector $\boldsymbol{x}_i \in \{0, 1\}^C$ indicating the cluster membership.

3.3. Proposed model (PMvGE)

Inspired by various existing methods for 0-view and 1-view feature learning, we propose a novel model for the parameter μ_{ij} in eq. (1) by using the inner-product similarity as

$$\mu_{ij}(\boldsymbol{\alpha}, \boldsymbol{\psi}) := \alpha^{(d_i,d_j)} \exp\left(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle\right), \tag{3}$$
$$\boldsymbol{y}_i := f_{\boldsymbol{\psi}}^{(d_i)}(\boldsymbol{x}_i).$$

Here $\boldsymbol{\alpha} = (\alpha^{(d,e)}) \in \mathbb{R}_{\geq 0}^{D \times D}$ is a symmetric parameter matrix ($\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top$) for regulating the sparseness of $\boldsymbol{W} = (w_{ij})$. For $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{R}^K$, $\langle \boldsymbol{y}, \boldsymbol{y}' \rangle = \sum_{i=1}^K y_i y_i'$ is simply the inner product in Euclidean space. The functions $f_{\boldsymbol{\psi}}^{(d)} : \mathbb{R}^{p_d} \to \mathbb{R}^K$, $d = 1, 2, \ldots, D$, specify the nonlinear transformations from data vectors to feature vectors. $\boldsymbol{\psi}$ represents a collection of parameters (e.g., neural network weights).

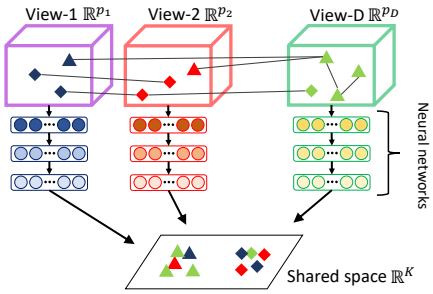


Figure 1. Data vectors $\{\boldsymbol{x}_i\}$ in each view are transformed to feature vectors $\{\boldsymbol{y}_i\}$ by neural networks $\{f_{\boldsymbol{\psi}}^{(d)}\}$.

These transformations $\{f_{\boldsymbol{\psi}}^{(d)}\}$ can be trained by maximizing the likelihood for the probabilistic model (1) as shown in Section 3.5. PMvGE computes feature vectors $\{\boldsymbol{y}_i\}_{i=1}^n$ through maximum likelihood estimation of transformations $\{f_{\boldsymbol{\psi}}^{(d)}\}$.

PMvGE associates nodes v_i and v_j with the probability specified by the similarity between their feature vectors $\boldsymbol{y}_i := f_{\boldsymbol{\psi}}^{(d_i)}(\boldsymbol{x}_i), \boldsymbol{y}_j := f_{\boldsymbol{\psi}}^{(d_j)}(\boldsymbol{x}_j) \in \mathbb{R}^K$ in the shared space. Nodes with similar feature vectors will share many links.

We consider the following neural network model for the transformation function, while any functional form can be accepted for PMvGE. Only the inner product of feature vectors is considered for measuring the similarity in PMvGE. We prove in Theorem 5.1 that the inner product with neural networks approximates a wide class of similarity measures.

Neural Network (NN) with 3-layers is defined as

$$f_{\boldsymbol{\psi}}^{(d)}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{\psi}_3^{(d)\top} \boldsymbol{\sigma}(\boldsymbol{\psi}_2^{(d)\top} \boldsymbol{\sigma}(\boldsymbol{\psi}_1^{(d)\top} \boldsymbol{x})))), \tag{4}$$

where $\boldsymbol{x} \in \mathbb{R}^{p_d}$ is data vector, $\boldsymbol{\psi}_1^{(d)} \in \mathbb{R}^{p_d \times K_1^{(d)}}$, $\boldsymbol{\psi}_2^{(d)} \in \mathbb{R}^{K_1^{(d)} \times K_2^{(d)}}$, $\boldsymbol{\psi}_3^{(d)} \in \mathbb{R}^{K_2^{(d)} \times K}$ are parameter matrices. The neural network size is specified by $p_d, K_1^{(d)}, K_2^{(d)}, K \in \mathbb{N}$ $(d = 1, 2, \ldots, D)$. Each element of $\boldsymbol{\sigma}(\boldsymbol{x})$ is user-specified activation function $\sigma(\cdot)$. Although we basically consider a multi-layer perceptron (MLP) as $f_{\boldsymbol{\psi}}^{(d)}$, it can be replaced with any deterministic NN with input and output layers, such as recurrent NN and Deep NN (DNN).

NN model reduces to linear model

$$f_{\boldsymbol{\psi}}^{(d)}(\boldsymbol{x}) := \boldsymbol{\psi}^{(d)\top} \boldsymbol{x} \tag{5}$$

$(d = 1, 2, \ldots, D)$ by applying $\sigma(x) = x$, where $\boldsymbol{\psi}^{(d)} \in \mathbb{R}^{p_d \times K}$ is parameter matrix.

3.4. Link weights across some view pairs may be missing

Link weights across all the view pairs may not be available in practice. So we consider the set of unordered view pairs

$$\mathcal{D} := \{(d, e) : \text{Link weights are observed between views } d, e\},$$

and we formally set $w_{ij} = 0$ for the missing $(d_i, d_j) \notin \mathcal{D}$. For example, $\mathcal{D} = \{(1, 2)\}$ for $D = 2$ indicates that link weights across view-1 and view-2 are observed while link weights within view-1 or view-2 are missing. We should notice the distinction between setting $w_{ij} = 0$ with missing and observing $w_{ij} = 0$ without missing, because these two cases give different likelihood functions.

3.5. Maximum Likelihood Estimator

Since PMvGE is specified by (1) and (3), the log-likelihood function is given by

$$\ell_n(\boldsymbol{\alpha},\boldsymbol{\psi}) := \sum_{(i,j)\in\mathcal{I}_n} [w_{ij}\log\mu_{ij}(\boldsymbol{\alpha},\boldsymbol{\psi}) - \mu_{ij}(\boldsymbol{\alpha},\boldsymbol{\psi})], \quad (6)$$

whose sum is over the set of index pairs $\mathcal{I}_n := \{(i,j) \mid 1 \le i < j \le n, (d_i,d_j) \in \mathcal{D}\}$. The Maximum Likelihood Estimator (MLE) of the parameter $(\boldsymbol{\alpha},\boldsymbol{\psi})$ is defined as the maximizer of (6). We estimate $(\boldsymbol{\alpha},\boldsymbol{\psi})$ by maximizing $\ell_n(\boldsymbol{\alpha},\boldsymbol{\psi})$ with constraints $\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top$ and $\alpha^{(d,e)} = 0$ for $(d,e) \notin \mathcal{D}$.

3.6. PMvGE approximately generalizes various methods for multi-view learning

SBM (2) is 1-view PMvGE with 1-hot cluster membership vector $\boldsymbol{x}_i \in \{0,1\}^C$. Consider the linear model (5) with $\boldsymbol{\psi}^{(1)\top} = (\boldsymbol{\psi}^1,\dots,\boldsymbol{\psi}^C) \in \mathbb{R}^{K\times C}$, where $\boldsymbol{\psi}^c \in \mathbb{R}^K$ is a feature vector for class $c = 1,\dots,C$. Then $\mu_{ij} = \alpha^{(1,1)}\exp(\langle\boldsymbol{\psi}^{c_i},\boldsymbol{\psi}^{c_j}\rangle)$ will specify $\beta^{(c_i,c_j)}$ for sufficiently large K, and $\{\boldsymbol{\psi}^{c_i}\}$ represent low-dimensional structure of SBM for smaller K. SBM is also interpreted as PMvGE with $D = C$ views by letting $d_i = c_i$ and $f_{\boldsymbol{\psi}}^{(d_i)}(\boldsymbol{x}) \equiv 0$. Then $\mu_{ij} = \alpha^{(c_i,c_j)}$ is equivalent to SBM.

More generally, CDMCA (Shimodaira, 2016) is approximated by D-view PMvGE with the linear transformations (5). The first half of the objective function (6) becomes

$$\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n w_{ij}\langle\boldsymbol{\psi}^{(d_i)\top}\boldsymbol{x}_i, \boldsymbol{\psi}^{(d_j)\top}\boldsymbol{x}_j\rangle \quad (7)$$

by specifying $\alpha^{(d,e)} \equiv 1$. CDMCA computes the transformation matrices $\{\boldsymbol{\psi}^{(d)}\}$ by maximizing this objective function under a quadratic constraint such as

$$\sum_{i=1}^n\sum_{j=1}^n w_{ij}\boldsymbol{\psi}^{(d_i)\top}\boldsymbol{x}_i\boldsymbol{x}_i^\top\boldsymbol{\psi}^{(d_i)} = \boldsymbol{I}. \quad (8)$$

The above observation intuitively explains why PMvGE approximates CDMCA; this is more formally discussed in Supplement C. The quadratic constraint is required for preventing the maximizer of (7) from being diverged in CDMCA. However, our likelihood approach does not require it, because the last half of (6) serves as a regularization term.

3.7. PMvGE represents neural network classifiers of multiple classes

For illustrating the generality of PMvGE, here we consider the multi-class classification problem. We show that PMvGE includes Feed-Forward Neural Network (FFNN) classifier as a special case.

Let $(\boldsymbol{x}_i, c_i) \in \mathbb{R}^p \times \{1,\dots,C\}$, $i = 1,\dots,n$ be the training data for the classification problem of C classes. FFNN classifier with softmax function (Bishop, 2006) is defined as

$$h_j(\boldsymbol{x}_i) := \frac{\exp(f_{\boldsymbol{\psi},j}(\boldsymbol{x}_i))}{\sum_{j=1}^C \exp(f_{\boldsymbol{\psi},j}(\boldsymbol{x}_i))}, \quad j = 1,\dots,C,$$

where $f_{\boldsymbol{\psi}}(\boldsymbol{x}) := (f_{\boldsymbol{\psi},1}(\boldsymbol{x}),\dots,f_{\boldsymbol{\psi},C}(\boldsymbol{x})) \in \mathbb{R}^C_{\ge 0}$ is a multi-valued neural network, and $\boldsymbol{x}_i$ is classified into the class $\underset{j\in\{1,2,\dots,C\}}{\arg\max}\ h_j(\boldsymbol{x}_i)$. This classifier is equivalent to

$$\underset{j\in\{1,2,\dots,C\}}{\arg\max}\ \exp\left(\langle f_{\boldsymbol{\psi}}(\boldsymbol{x}_i), \boldsymbol{e}_j\rangle\right), \quad (9)$$

where $\boldsymbol{e}_j \in \{0,1\}^C$ is the 1-hot vector with 1 at j-th entry and 0 otherwise.

The classifier (9) can be interpreted as PMvGE with $D = 2, \mathcal{D} = \{(1,2)\}$ as follows. For view-1, $f_{\boldsymbol{\psi}}^{(1)}(\boldsymbol{x}) := f_{\boldsymbol{\psi}}(\boldsymbol{x})$ and inputs are $\boldsymbol{x}_1,\dots,\boldsymbol{x}_n$. For view-2, $f_{\boldsymbol{\psi}}^{(2)}(\boldsymbol{x}') := \boldsymbol{x}'$ and inputs are $\boldsymbol{e}_1,\dots,\boldsymbol{e}_C$. We set $w_{ij} = 1$ between $\boldsymbol{x}_i$ and $\boldsymbol{e}_{c_i}$, and $w_{ij} = 0$ otherwise.

4. Optimization

In this section, we present an efficient way of optimizing the parameters for maximizing the objective function $\ell_n(\boldsymbol{\alpha},\boldsymbol{\psi})$ defined in eq. (6). We alternatively optimize the two parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\psi}$. Efficient update of $\boldsymbol{\psi}$ with minibatch SGD is considered in Section 4.1, and update of $\boldsymbol{\alpha}$ by solving an estimating equation is considered in Section 4.2. We iterate these two steps for maximizing $\ell_n(\boldsymbol{\alpha},\boldsymbol{\psi})$.

4.1. Update of $\boldsymbol{\psi}$

We update $\boldsymbol{\psi}$ using the gradient of $\ell_n(\bar{\boldsymbol{\alpha}},\boldsymbol{\psi})$ by fixing current parameter value $\bar{\boldsymbol{\alpha}}$. Since $\boldsymbol{W} = (w_{ij})$ may be sparse in practice, the computational cost of minibatch SGD (Goodfellow et al., 2016) for the first half of $\ell_n(\bar{\boldsymbol{\alpha}},\boldsymbol{\psi})$ is expected to be reduced by considering the sum over the set $\mathcal{W}_n := \{(i,j) \in \mathcal{I}_n \mid w_{ij} > 0\}$. On the other hand, there should be $|\mathcal{I}_n| = O(n^2)$ positive terms in the last half of $\ell_n(\bar{\boldsymbol{\alpha}},\boldsymbol{\psi})$, so we consider the sum over node pairs uniformly-resampled from $\mathcal{I}_n$.

We make two sets $\mathcal{I}'_n, \mathcal{W}'_n$ by picking (i,j) from $\mathcal{I}_n$ and $\mathcal{W}_n$, respectively, so that

$$|\mathcal{I}'_n| + |\mathcal{W}'_n| = m,\ |\mathcal{I}'_n|/|\mathcal{W}'_n| = r.$$

User-specified constants $m \in \mathbb{N}$ and $r > 0$ are usually called as "minibatch size" and "negative sampling rate".

We sequentially update minibatch $\mathcal{I}'_n, \mathcal{W}'_n$ for computing the gradient of

$$\sum_{(i,j)\in\mathcal{W}'_n} w_{ij} \log \mu_{ij}(\bar{\boldsymbol{\alpha}}, \boldsymbol{\psi}) - \tau \sum_{(i,j)\in\mathcal{I}'_n} \mu_{ij}(\bar{\boldsymbol{\alpha}}, \boldsymbol{\psi}) \qquad (10)$$

with respect to $\boldsymbol{\psi}$. By utilizing the gradient, the parameter $\boldsymbol{\psi}$ can be sequentially updated by SGD, where $\tau > 0$ is a tuning parameter. Eq. (10) approximates $\ell_n(\bar{\boldsymbol{\alpha}}, \boldsymbol{\psi})$ if (i, j) are uniformly-resampled and $\tau = |\mathcal{I}_n|/(r\,|\mathcal{W}_n|)$, however, smaller τ such as $\tau = 1$ may make this algorithm stable in some cases.

4.2. Update of $\boldsymbol{\alpha}$

Let $\bar{\boldsymbol{\psi}}$ represent current parameter value of $\boldsymbol{\psi}$. By solving the estimating equation $\frac{\partial \ell_n(\boldsymbol{\alpha}, \bar{\boldsymbol{\psi}})}{\partial \alpha^{(d,e)}} = 0$ with respect to $\alpha^{(d,e)}$ under constraints $\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top$ and $\alpha^{(d,e)} = 0$, $(d, e) \notin \mathcal{D}$, we explicitly obtain a local maximizer of $\ell_n(\boldsymbol{\alpha}, \bar{\boldsymbol{\psi}})$. However, the local maximizer requires roughly $O(n^2)$ operations for computation. To reduce the high computational cost, we efficiently update $\boldsymbol{\alpha}$ by (11), which is a minibatch-based approximation of the local maximizer.

$$\hat{\alpha}^{(d,e)} := \begin{cases} \dfrac{\sum_{(i,j)\in\mathcal{I}_n^{'(d,e)}} w_{ij}}{\sum_{(i,j)\in\mathcal{I}_n^{'(d,e)}} \exp(\bar{g}_{ij})} & \text{for } (d, e) \in \mathcal{D} \\ 0 & \text{otherwise,} \end{cases} \qquad (11)$$

where $\bar{g}_{ij} := \langle f_{\bar{\boldsymbol{\psi}}}^{(d)}(\boldsymbol{x}_i), f_{\bar{\boldsymbol{\psi}}}^{(e)}(\boldsymbol{x}_j)\rangle$, $\mathcal{I}_n^{'(d,e)} := \{(i, j) \in \mathcal{I}'_n \mid (d_i, d_j) = (d, e)\}$, and $\mathcal{I}'_n$ is the minibatch defined in Section 4.1. Note that (d, e) is unordered view pair.

4.3. Computational cost

PMvGE requires $O(m)$ operations for each minibatch iteration. It is efficiently computed even if the number of data vectors n is very large.

5. PMvGE learns arbitrary similarity measure

Two theoretical results are shown here for indicating that PMvGE with sufficiently large neural networks learns arbitrary similarity measure using sufficiently many data vectors. In Section 5.1, we prove that arbitrary similarity measure can be approximated by the inner product in the shared space with sufficiently large neural networks. In Section 5.2, we prove that MLE of PMvGE converges to the true parameter value, i.e., the consistency of MLE, in some sense as the number of data vectors increases.

5.1. Inner product of NNs approximates a wide class of similarity measures across views

Feedforward neural networks with ReLU or sigmoid function are proved to be able to approximate arbitrary continuous functions under some assumptions (Cybenko, 1989; Funahashi, 1989; Yarotsky, 2016; Telgarsky, 2017). However, these results cannot be directly applied to PMvGE, because our model is based on the inner product of two neural networks $\langle f_{\boldsymbol{\psi}}^{(d_i)}(\boldsymbol{x}_i), f_{\boldsymbol{\psi}}^{(d_j)}(\boldsymbol{x}_j)\rangle$. In Theorem 5.1, we show that the inner product can approximate $g_*\left(f_*^{(d_i)}(\boldsymbol{x}_i), f_*^{(d_j)}(\boldsymbol{x}_j)\right)$, that is, arbitrary similarity measure $g_*(\cdot,\cdot)$ in K_* dimensional shared space, where $f_*^{(d_i)}$, $f_*^{(d_j)}$ are arbitrary two continuous functions. For showing $g_*(f_*^{(d)}(\boldsymbol{x}), f_*^{(e)}(\boldsymbol{x}')) \approx \langle f_{\boldsymbol{\psi}}^{(d)}(\boldsymbol{x}), f_{\boldsymbol{\psi}}^{(e)}(\boldsymbol{x}')\rangle$, the idea is first consider a feature map $\boldsymbol{\Phi}_K : \mathbb{R}^{K^*} \to \mathbb{R}^K$ for approximating $g_*(\boldsymbol{y}_*, \boldsymbol{y}'_*) \approx \langle \boldsymbol{\Phi}_K(\boldsymbol{y}_*), \boldsymbol{\Phi}_K(\boldsymbol{y}'_*)\rangle$ with sufficiently large K, and then consider neural networks $f_{\boldsymbol{\psi}}^{(d)} : \mathbb{R}^{p_d} \to \mathbb{R}^K$ with sufficiently many hidden units for approximating $\boldsymbol{\Phi}_K(f_*^{(d)}(\boldsymbol{x})) \approx f_{\boldsymbol{\psi}}^{(d)}(\boldsymbol{x})$.

Theorem 5.1 Let $f_*^{(d)} : [-M, M]^{p_d} \to [-M', M']^{K_*}$, $d = 1, 2, \ldots, D$, be continuous functions and $g_* : [-M', M']^{K_*} \times [-M', M']^{K_*} \to \mathbb{R}$ be a symmetric, continuous, and positive-definite kernel function for some $K^*, M, M' > 0$. $\sigma(\cdot)$ is ReLU or activation function which is non-constant, continuous, bounded, and monotonically-increasing. Then, for arbitrary $\varepsilon > 0$, by specifying sufficiently large $K \in \mathbb{N}, T = T(K) \in \mathbb{N}$, there exist $\boldsymbol{A}^d \in \mathbb{R}^{K\times T}, \boldsymbol{B}^d \in \mathbb{R}^{T\times p_d}, \boldsymbol{c} \in \mathbb{R}^T$, $d \in \{1, 2, \ldots, D\}$, such that

$$\left| g_*\left(f_*^{(d)}(\boldsymbol{x}), f_*^{(e)}(\boldsymbol{x}')\right) - \langle f_{\boldsymbol{\psi}}^{(d)}(\boldsymbol{x}), f_{\boldsymbol{\psi}}^{(e)}(\boldsymbol{x}')\rangle \right| < \varepsilon \qquad (12)$$

for all $(\boldsymbol{x}, \boldsymbol{x}') \in [-M, M]^{p_d+p_e}$, $d, e \in \{1, 2, \ldots, D\}$, where $f_{\boldsymbol{\psi}}^{(d)}(\boldsymbol{x}^d) = \boldsymbol{A}^d\boldsymbol{\sigma}(\boldsymbol{B}^d\boldsymbol{x}^d + \boldsymbol{c}^d)$, $d = 1, 2, \ldots, D$, are two-layer neural networks with T hidden units and $\boldsymbol{\sigma}(\boldsymbol{x})$ is element-wise $\sigma(\cdot)$ function.

Proof of the theorem is given in Supplement A.

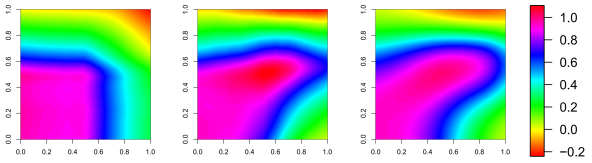
If $D = 1$, Theorem 5.1 corresponds to Mercer's theorem (Mercer, 1909; Courant and Hilbert, 1989) of Kernel methods, which states that arbitrary positive definite kernel $g_*(\cdot,\cdot)$ can be expressed as the inner product of high-dimensional feature maps. While Mercer's theorem indicates only the existence of such feature maps, Theorem 5.1 also states that the feature maps can be approximated by neural networks.

Illustrative example As an example of positive definite similarity, we consider the cosine similarity

$$g_*(f_*(\boldsymbol{x}), f_*(\boldsymbol{x}')) := \frac{\langle f_*(\boldsymbol{x}), f_*(\boldsymbol{x}')\rangle}{\|f_*(\boldsymbol{x})\|_2\|f_*(\boldsymbol{x}')\|_2}$$

with $f_*(\boldsymbol{x}) = (x_1, \cos x_2, \exp(-x_3), \sin(x_4 - x_5))$, $p = 5$. For 2-dim visualization in Fig. 2 with $(s, t) \in \mathbb{R}^2$, let us define $G_*(s, t) := g_*(f_*(s\boldsymbol{e}_1), f_*(t\boldsymbol{e}_2))$, $\boldsymbol{e}_1 :=$

$(1,1,1,0,0), \boldsymbol{e}_2 := (0,0,1,1,1)$ and its approximation by the inner product of neural networks $\hat{G}_K(s,t) := \langle f_{\boldsymbol{\psi}}(s\boldsymbol{e}_1), f_{\boldsymbol{\psi}}(t\boldsymbol{e}_2)\rangle$ with $T = 10^3$ hidden units. If K and T are sufficiently large, $\hat{G}_K(s,t)$ approximates $G_*(s,t)$ very well as suggested by Theorem 5.1.



(a) $\hat{G}_2(s,t)$ (b) $\hat{G}_{50}(s,t)$ (c) $G_*(s,t)$

Figure 2. A two-dim visualization $G_*(s,t)$ of similarity measure $g_*(\boldsymbol{y}_*, \boldsymbol{y}'_*) = \frac{\langle \boldsymbol{y}, \boldsymbol{y}'\rangle}{\|\boldsymbol{y}\|_2 \|\boldsymbol{y}'\|_2}$ with $K_* = 2$ is well approximated by the visualization $\hat{G}_K(s,t)$ of the inner product $\langle \boldsymbol{y}, \boldsymbol{y}'\rangle$ for $K = 100$, while the approximation is poor for $K = 2$. A neural network $f_{\boldsymbol{\psi}}$ with $T = 10^3$ ReLU units and $K = 2$ or $K = 50$ linear output units are trained with $n = 10^3$ data vectors and link weights across views.

5.2. MLE converges to the true parameter value

We have shown the universal approximation theorem of similarity measure in Theorem 5.1. However, it only states that the good approximation is achieved if we properly tune the parameters of neural networks. Here we argue that MLE of Section 3.5 will actually achieve the good approximation if we have sufficiently many data vectors. The technical details of the argument are given in Supplement B.

Let $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ denote the vector of free parameters in $\boldsymbol{\alpha}, \boldsymbol{\psi}$, and $\ell_n(\boldsymbol{\theta})$ be the log-likelihood function (6). We assume that the optimization algorithm in Section 4 successfully computes MLE $\hat{\boldsymbol{\theta}}_n$ that maximizes $\ell_n(\boldsymbol{\theta})$. Here we ignore the difficulty of global optimization, while we may only get a local maximum in practice. We also assume that PMvGE is correctly specified; there exists a parameter value $\boldsymbol{\theta}_*$ so that the parametric model represents the true probability distribution.

Then, we would like to claim that $\hat{\boldsymbol{\theta}}_n$ converges to the true parameter value $\boldsymbol{\theta}_*$ in the limit of $n \to \infty$, the property called the consistency of MLE. However, we have to pay careful attention to the fact that PMvGE is not a standard setting in the sense that (i) there are correlated $O(n^2)$ samples instead of n i.i.d. observations, and (ii) the model is not identifiable with infinitely many equivalent parameter values; for example there are rotational degrees of freedom in the shared space so that $\langle \boldsymbol{y}, \boldsymbol{y}'\rangle = \langle \boldsymbol{O}\boldsymbol{y}, \boldsymbol{O}\boldsymbol{y}'\rangle$ with any orthogonal matrix $\boldsymbol{O}$ in $\mathbb{R}^K$. We then consider the set of equivalent parameters $\hat{\boldsymbol{\Theta}}_n := \{\boldsymbol{\theta} \in \boldsymbol{\Theta} \mid \ell_n(\boldsymbol{\theta}) = \ell_n(\hat{\boldsymbol{\theta}}_n)\}$. Theorem B.2 states that, as $n \to \infty$, $\hat{\boldsymbol{\Theta}}_n$ converges to $\boldsymbol{\Theta}_*$, the set of $\boldsymbol{\theta}$ values equivalent to $\boldsymbol{\theta}_*$.

6. Real data analysis

6.1. Experiments on Citation dataset (1-view)

Dataset: We use Cora dataset (Sen et al., 2008) of citation network with 2,708 nodes and 5,278 ordered edges. Each node v_i represents a document, which has $1,433$-dimensional (bag-of-words) data vector $\boldsymbol{x}_i \in \{0,1\}^{1433}$ and a class label of 7 classes. Each directed edge represents citation from a document v_i to another document v_j. We set the link weight as $w_{ij} = w_{ji} = 1$ by ignoring the direction, and $w_{ij} = 0$ otherwise. There is no cross or self-citation. We divide the dataset into training set consisting of $2,166$ nodes (80%) with their edges, and test set consisting of remaining 542 nodes (20%) with their edges. We utilize 20% of the training set for validation. Hyperparameters are tuned by utilizing the validation set.

We compare PMvGE with several feature learning methods: Stochastic Block Model (Holland et al., 1983, SBM), ISOMAP (Tenenbaum et al., 2000), Locally Linear Embedding (Roweis and Saul, 2000, LLE), Spectral Graph Embedding (Belkin and Niyogi, 2001, SGE), Multi Dimensional Scaling (Kruskal, 1964, MDS), DeepWalk (Perozzi et al., 2014), and GraphSAGE (Hamilton et al., 2017).

NN for PMvGE: 2-layer fully-connected network, which consists of 3,000 tanh hidden units and 1,000 tanh output units, is used. The network is trained by Adam (Kingma and Ba, 2015) with batch normalization. The learning rate is starting from 0.0001 and attenuated by $1/10$ for every 100 iterations. Negative sampling rate r and minibatch size m are set as 1 and 512, respectively, and the number of iterations is 200.

Parameter tuning: For each method, parameters are tuned on validation sets. Especially, the dimension of feature vectors is selected from $\{50, 100, 150, 200\}$.

Label classification (Task 1): We classify the documents into 7 classes using logistic regression with the feature vector as input and the class label as output. We utilize LibLinear (Fan et al., 2008) for the implementation.

Clustering (Task 2): The k-means clustering of the feature vectors is performed for unsupervised learning of document clusters. The number of clusters is set as 7.

Results: The quality of classification is evaluated by classification accuracy in Task 1, and Normalized Mutual Information (NMI) in Task 2. Sample averages and standard deviations over 10 experiments are shown in Table 2. In experiment (A), we apply methods to both training set and test set, and evaluate them by test set. In (B), we apply methods to only the training set, and evaluate them by test set. SGE, MDS, and DeepWalk are not inductive, and they cannot be applied to unseen data vectors in (B). PMvGE

outperforms the other methods in both experiments.

6.2. Experiments on AwA dataset (2-view)

Dataset: We use Animal with Attributes (AwA) dataset (Lampert et al., 2009) with 30,475 images for view-1 and 85 attributes for view-2. We prepared $4,096$ dimensional DeCAF data vector (Donahue et al., 2014) for each image, and 300 dimensional GloVe (Pennington et al., 2014) data vector for each attribute. Each image is associated with some attributes. We set $w_{ij} = 1$ for the associated pairs between the two views, and $w_{ij} = 0$ otherwise. In addition to the attributes, each image has a class label of 50 classes. We resampled 50 images from each of 50 classes; in total, 2500 images. In each experiment, we split the 2500 images into 1500 training images and 100 test images. A validation set of 300 images is sampled from the training images.

We compare PMvGE with CCA, DCCA (Andrew et al., 2013), SGE, DeepWalk, and GraphSAGE.

NN for PMvGE: Each view uses a 2-layer fully-connected network, which consists of 2000 tanh hidden units and 100 tanh output units. The dimension of the feature vector is $K = 100$. Adam is used for optimization with Batch normalization and Dropout ($p = 0.5$). Minibatch size, learning rate, and momentum are tuned on the validation set. We monitor the score on the validation set for early stopping.

Parameter tuning: For each method, parameters are tuned on validation sets. Especially, the dimension of feature vectors is selected from $\{10, 50, 100, 150\}$.

Link prediction (Task 3): For each query image, we rank attributes according to the cosine similarity of feature vectors across views. An attribute is regarded as correct if it is associated with the query image.

Results: The quality of the ranked list of attributes is measured by Average Precision (AP) score in Task 3. Sample averages and standard deviations over 10 experiments are shown in Table 2. In experiment (A), we apply methods to both training set and test set, and evaluate them by test set. The whole training set is used for validation. In experiment (B), we apply methods to only the training set, and evaluate them by test set. 20% of training set is used for validation. PMvGE outperforms the other methods including DCCA. While DeepWalk shows good performance in experiment (A), DeepWalk and SGE cannot be applied to unseen data vectors in (B). Unlike SGE and DeepWalk which only consider the associations, 1-view feature learning methods such as GraphSAGE cannot be applied to this AwA dataset since the dimension of data vectors is different depending on the view. So we do not perform 1-view methods in Task 3.

Table 2. Task 1 and Task 2 for the experiment on Citation dataset ($D = 1$), and Task 3 for the experiment on AwA dataset ($D = 2$). The larger values are better.

		(A)	(B)
Task 1 ($D = 1$)	ISOMAP	54.5 ± 1.78	54.8 ± 2.43
	LLE	30.2 ± 1.91	31.9 ± 2.62
	SGE	47.6 ± 1.64	-
	MDS	29.8 ± 2.25	-
	DeepWalk	54.2 ± 2.04	-
	GraphSAGE	60.8 ± 1.73	57.1 ± 1.61
	PMvGE	$\mathbf{74.8 \pm 2.55}$	$\mathbf{71.1 \pm 2.10}$
Task 2 ($D = 1$)	SBM	4.37 ± 1.44	2.81 ± 0.10
	ISOMAP	13.0 ± 0.36	14.3 ± 1.98
	LLE	7.40 ± 3.40	9.47 ± 3.00
	SGE	1.41 ± 0.34	-
	MDS	2.81 ± 0.10	-
	DeepWalk	16.7 ± 1.05	-
	GraphSAGE	19.6 ± 0.93	12.4 ± 3.00
	PMvGE	$\mathbf{35.9 \pm 0.88}$	$\mathbf{30.5 \pm 3.90}$
Task 3 ($D = 2$)	CCA	45.5 ± 0.20	42.4 ± 0.30
	DCCA	41.4 ± 0.30	41.2 ± 0.35
	SGE	43.5 ± 0.39	-
	DeepWalk	71.3 ± 0.57	-
	PMvGE	$\mathbf{71.5 \pm 0.48}$	$\mathbf{70.5 \pm 0.53}$

Locality of each-view is preserved through neural networks: To see whether the locality of input is preserved through neural networks in PMvGE, we computed the Spearman's rank correlation coefficient between $\langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ and $\langle f_{\boldsymbol{\psi}}^{(d)}(\boldsymbol{x}), f_{\boldsymbol{\psi}}^{(d)}(\boldsymbol{x}') \rangle$ for view-d data vectors $\boldsymbol{x}, \boldsymbol{x}'$ in AwA dataset ($d = 1, 2$). For DeCAF (view-1) and GloVe (view-2) inputs, the values are 0.722 ± 0.058 and 0.811 ± 0.082, respectively. This result indicates that the feature vectors of PMvGE preserves the similarities of the data vectors fairly well.

7. Conclusion

We presented a simple probabilistic framework for multi-view learning with many-to-many associations. We name the framework as Probabilistic Multi-view Graph Embedding (PMvGE). Various existing methods are approximately included in PMvGE. We gave theoretical justification and practical estimation algorithm to PMvGE. Experiments on real-world datasets showed that PMvGE outperforms existing methods.

Acknowledgement

This work was partially supported by JSPS KAKENHI grant 16H02789 to HS and 17J03623 to AO.

References

Andrew, G., Arora, R., Bilmes, J., and Livescu, K. (2013). Deep Canonical Correlation Analysis. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1247–1255.

Belkin, M. and Niyogi, P. (2001). Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 585–591.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Chung, F. R. (1997). *Spectral Graph Theory*. American Mathematical Society.

Courant, R. and Hilbert, D. (1989). *Methods of Mathematical Physics*, volume 1. Wiley, New York.

Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314.

Dai, Q., Li, Q., Tang, J., and Wang, D. (2018). Adversarial Network Embedding. In *Proceedings of the conference on Artificial Intelligence (AAAI)*.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3844–3852.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 647–655.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research (JMLR)*, 9:1871–1874.

Fukui, K., Okuno, A., and Shimodaira, H. (2016). Image and tag retrieval by leveraging image-group links with multi-domain graph embedding. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 221–225.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable Feature Learning for Networks. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data mining (SIGKDD)*, pages 855–864. ACM.

Hamilton, W. L., Ying, Z., and Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems (NIPS)*, pages 1025–1035.

He, X. and Niyogi, P. (2004). Locality Preserving Projections. In *Advances in Neural Information Processing Systems (NIPS)*, pages 153–160.

Holland, P. W., Laskey, K. B., and Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Huang, Z., Shan, S., Zhang, H., Lao, S., and Chen, X. (2012). Cross-view Graph Embedding. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 770–781.

Kettenring, J. R. (1971). Canonical Analysis of Several Sets of Variables. *Biometrika*, 58(3):433–451.

Kingma, D. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27.

Lai, P. L. and Fyfe, C. (2000). Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(05):365–377.

Lampert, C. H., Nickisch, H., and Harmeling, S. (2009). Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 951–958. IEEE.

Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London. Series A.*, 209:415–446.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Nori, N., Bollegala, D., and Kashima, H. (2012). Multinomial Relation Prediction in Social Data: A Dimension Reduction Approach. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 12, pages 115–121.

Nowicki, K. and Snijders, T. A. B. (2001). Estimation and Prediction for Stochastic Blockstructures. *Journal of the American Statistical Association*, 96(455):1077–1087.

Oshikiri, T., Fukui, K., and Shimodaira, H. (2016). Cross-Lingual Word Representations via Spectral Graph Embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 493–498.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). DeepWalk: Online Learning of Social Representations. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data mining (SIGKDD)*, pages 701–710. ACM.

Rendle, S. (2010). Factorization Machines. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 995–1000. IEEE.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective Classification in Network Data. *AI magazine*, 29(3):93.

Shimodaira, H. (2016). Cross-validation of matching correlation analysis by resampling matching weights. *Neural Networks*, 75:126–140.

Sun, S. (2013). A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). LINE: Large-scale Information Network Embedding. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Telgarsky, M. (2017). Neural networks and rational functions. In Precup, D. and Teh, Y. W., editors, *Proceedings of the International Conference on Machine Learning (ICML)*.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323.

Wang, W., Yan, X., Lee, H., and Livescu, K. (2016). Deep Variational Canonical Correlation Analysis. *arXiv preprint arXiv:1610.03454*.

Yan, F. and Mikolajczyk, K. (2015). Deep Correlation for Matching Images and Text. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3441–3450. IEEE.

Yan, S., Xu, D., Zhang, B., Zhang, H.-J., Yang, Q., and Lin, S. (2007). Graph Embedding and Extensions: A General Framework for Dimensionality Reduction. *IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(1):40–51.

Yarotsky, D. (2016). Error bounds for approximations with deep ReLU networks. *arXiv preprint arXiv:1610.01145*.

Zhao, J., Xie, X., Xu, X., and Sun, S. (2017). Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54.

Zheng, W., Zhou, X., Zou, C., and Zhao, L. (2006). Facial expression recognition using kernel canonical correlation analysis (KCCA). *IEEE transactions on Neural Networks*, 17(1):233–238.