KYOTO UNIVERSITY

DOCTORAL THESIS

Studies on Neural Network-Based Graph Embedding and Its Extensions

---

Author:
Akifumi OKUNO

Supervisor:
Prof. Hidetoshi SHIMODAIRA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Informatics

in the

Department of Systems Science
Graduate School of Informatics

July, 2020

Abstract

This thesis discusses a neural network (NN)-based graph embedding and its extensions. The NN-based graph embedding computes feature vectors for nodes of a graph, by leveraging observed *data vectors* of nodes and their association strengths called *link weights*. Obtained feature vectors are, in general, (i) containing richer information than the original data vectors, as they are computed from both data vectors and link weights, and (ii) computationally tractable, as their dimension is reduced. Therefore, the graph embedding can be regarded as a kind of dimensionality reduction; the obtained feature vectors can be used for a variety of tasks in machine learning and statistics.

Although existing graph embedding methods achieve success from application perspectives, several challenges remain; this thesis provides the following three contributions towards a theoretically-guaranteed unified framework for the NN-based graph embedding.

The first contribution is applying NN-based graph embedding to different types of data vectors, i.e., users, images, and texts in social networking services, where the data type is called a *view*. Although such multi-view analysis has been developed, the existing methods in line with correlation analysis are mainly based on linear models; we extend the NN-based graph embedding to the multi-view setting, and propose *probabilistic multi-view graph embedding (PMvGE)*.

The second contribution is studying the expressive power of the NN-based graph embedding. Although inner product similarity (IPS) is often employed for the graph embedding, we may replace the IPS with other arbitrary similarities, as a study called Poincaré embedding employs negative Poincaré distance instead. For shedding light on the nature of similarity functions, we first prove that IPS is limited to approximate positive-definite (PD) similarities, and provide more expressive similarity called *shifted-IPS (SIPS)*. SIPS is capable of approximating a wider class than IPS. Furthermore, we provide a more expressive *inner-product difference similarity (IPDS)*, and prove that IPDS can approximate arbitrary similarity; graph embedding equipped with SIPS or IPDS is expected to demonstrate better performance than that with IPS.

The third contribution is extending the NN-based graph embedding to hyper-relations. A collection of U ($\in \mathbb{N}$) data vectors is called a *U-tuple*, and an association strength among the vectors of a tuple is termed as a *hyperlink weight*. We propose *Bregman hyperlink regression (BHLR)*, which learns a user-specified symmetric similarity function such that it predicts the tuple's hyperlink weight from the data vectors stored in the corresponding tuple. Whereas BHLR is a simple and general framework for hyper-relational learning including graph embedding, we prove that general BHLR possesses favorable theoretical properties, such as statistical estimation consistency.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Prof. Hidetoshi Shimodaira. In Osaka University, Kyoto University, and RIKEN center for Advanced Intelligence Project (AIP), where all the institutions I have belonged to, he has provided me with lucid and insightful advice. His continuous and enormous support was essential for completing my doctoral studies, and I have been always impressed by his unique ideas. I am also profoundly indebted to the members of my thesis committee, Prof. Toshiyuki Tanaka and Prof. Hisashi Kashima. Their helpful and suggestive comments significantly improved the quality of this thesis.

Secondly, I would like to express my appreciation to all the Shimodaira team members, especially my collaborators Kazuki Fukui, Tetsuya Hada, and Geewook Kim, and my colleagues Masaaki Inoue and Atsuto Maeda. I have discussed a variety of topics with them in the laboratory. My special thanks also go to Associate Prof. Kei Hirose, Associate Prof. Shinpei Imori, Associate Prof. Yoshikazu Terada, Dr. Tomoharu Iwata, Dr. Thong Pham, Haruhisa Nagata, Haruyoshi Maeda, Takamasa Oshikiri, and Naomi Nakagami.

Thirdly, I am grateful to the researchers from other institutions. Especially, Associate Prof. Keisuke Yano, Associate Prof. Masaaki Imaizumi, Associate Prof. Yuta Umezu, Associate Prof. Daisuke Yoneoka, Lecturer Kota Matsui, and Assistant Prof. Kosuke Morikawa for their kind support for my academic activities. Also, I appreciate all the friends of mine, especially K. Tamura, T. Takabatake, T. Kawashima, K. Uno. and H. Washizu. It was a nice change of pace, to chat with the people outside of the laboratory.

Finally, I would like to express my heartfelt gratitude to my family. Everything I have now is thanks to their strong and unconditional support.

July, 2020
Akifumi Okuno

List of Publications

Related Journal Paper

(1) Okuno, A. and Shimodaira, H. (2020). Hyperlink regression via Bregman divergence. *Neural Networks*, 126:362-383.

Related Conference Papers

(2) Okuno, A., Hada, T. and Shimodaira, H. (2018). A probabilistic framework for multi-view feature learning with many-to-many associations via neural networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, in PMLR 80:3888-3897.

(3) Okuno, A., Kim, G. and Shimodaira, H. (2019). Graph Embedding with Shifted Inner Product Similarity and Its Improved Approximation Capability. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, in PMLR 89:644-653.

(4) Okuno, A. and Shimodaira, H. (2019). Robust Graph Embedding with Noisy Link Weights. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, in PMLR 89:664-673.

Other Conference/Workshop Papers

(5) 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 2016 IEEE International Conference on Image Processing (ICIP)*, pages 221-225.

(6) Okuno, A. and Shimodaira, H. (2018). On representation power of neural network-based graph embedding and beyond. ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models. 13 pages. arXiv:1805:12332. `https://sites.google.com/view/tadgm/accepted-papers`

(7) Kim, G., Okuno, A., Fukui, K. and Shimodaira, H. (2019). Representation Learning with Weighted Inner Product for Universal Approximation of General Similarities. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5031-5038.

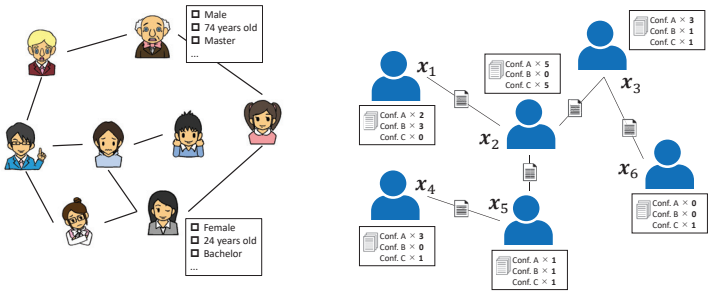
Contents

1 Introduction

In this chapter, we first briefly describe the background of this thesis in Section 1.1. Descriptions are based on a paper of ours (Okuno and Shimodaira, 2020), whereas additional descriptions and figures are also provided. Subsequently, three main challenges considered in this thesis are presented in Section 1.2, and our contributions to solving these challenges are summarized in Section 1.3. The organization of this thesis is shown in Section 1.4.

1.1 Background

Many real-world datasets are represented in the form of undirected graphs comprising nodes and their links, where nodes may have attributes called *data vectors* and the links are specified by *link weights* $w_{ij} = w_{ji} \geq 0$. Link weight represents the association strength between the corresponding data vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$ of nodes v_i, v_j. A friend network shown in Figure 1.1(A) is an example; data vectors and binary link weights represent properties of people and their friendships, respectively. Co-authorship network shown in Figure 1.1(B) is also an example, as attributes of a researcher such as the number of publications in each journal may specify the data vectors, and the number of publications written by the two researchers specifies the link weights. Furthermore, many other examples of such link weights and data vectors can be found in real-world situations.



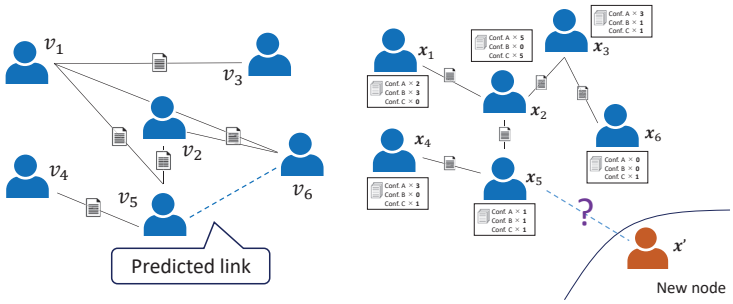
(A) Friend network

(B) Co-authorship network

FIGURE 1.1: Practical examples of link weights and data vectors

Although such a graph-structured dataset contains rich information, a large number of underlying link weights may be missing in practice (Clauset, Moore, and Newman, 2008; Lü and Zhou, 2011). Such missing link weights may be inferred by considering the observed link weights as shown in Figure 1.2(A); for instance, two nodes that are connected to the same types of

nodes in common are supposed to have high link weights (Lü and Zhou, 2011; Liben-Nowell and Kleinberg, 2007). However, such an inference deteriorates easily when no or only a few positive link weights to the target nodes are observed; link weights to unseen nodes are especially hard to predict in this way, as none of the link weights are observed, as shown in Figure 1.2(B).



(A) Link between training nodes

(B) Link to unseen node

FIGURE 1.2: Whereas (A) link weights may be inferred if some positive link weights to the target node are observed, (B) link weights to unseen nodes are more difficult to predict, as positive link weights to the nodes are not observed. For considering the situation (B), we employ observed data vectors for nodes throughout this thesis, in addition to the observed link weights; similarities of the data vectors imply the link weights.

Even in a severe situation, missing link weights can be inferred by additionally utilizing node data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$, as their similarity $\mu(\boldsymbol{x}_i, \boldsymbol{x}_j)$ implies the corresponding link weight w_{ij}, for $1 \leq i < j \leq n$. Thus, various methods inferring link weights through data vectors, which are often implemented with neural networks these days, have been developed. The method is also called *inductive*, if it can predict the link weights to unseen nodes which are not included in the training data. As w_{ij} is predicted from the corresponding data vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$, we generalize these methods as *link regression*.

A simple implementation of link regression is similarity learning, where a user-specified similarity function $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ defined for pairs of data vectors is trained to predict link weights $\{w_{ij}\}$. Although arbitrary similarity functions can be employed, many existing studies leverage the Mahalanobis distance (De Maesschalck, Jouan-Rimbaud, and Massart, 2000) $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = -\|\boldsymbol{\theta}^\top \boldsymbol{x}_j - \boldsymbol{\theta}^\top \boldsymbol{x}_j\|_2^2$ and Mahalanobis inner product (Kung, 2014) $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\theta}^\top \boldsymbol{x}_i, \boldsymbol{\theta}^\top \boldsymbol{x}_j \rangle$ where $\langle \cdot, \cdot \rangle$ represents inner product and $\boldsymbol{\theta}$ represents a $p \times K$ matrix. Using these Mahalanobis similarities is mathematically equivalent to using the Euclidean distance or inner product between low-dimensional linearly transformed data vectors $\boldsymbol{y}_i = \boldsymbol{\theta}^\top \boldsymbol{x}_i$ (Goldberger et al., 2005), implying that Mahalanobis similarity learning implicitly obtains a low-dimensional linear transformation $\mathbb{R}^p \ni \boldsymbol{x} \mapsto \boldsymbol{y} := \boldsymbol{\theta}^\top \boldsymbol{x} \in \mathbb{R}^K$. We especially call the

vector $\boldsymbol{y}$ as *feature vector*, and obtaining such a transformation is also classified into *graph embedding (GE)*.

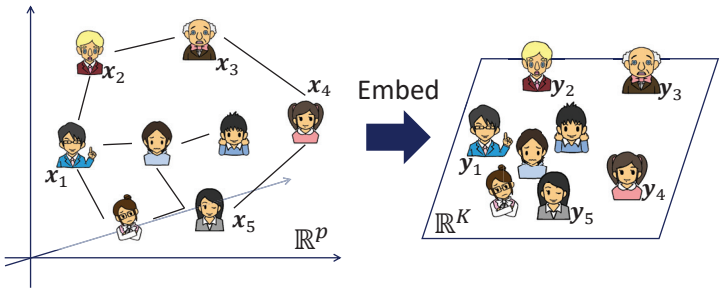


FIGURE 1.3: Graph embedding transforms p-dimensional observed data vectors into K-dimensional feature vectors. Usually, K is specified smaller than p; graph embedding is a kind of dimensionality reduction, that considers graph-structured association.

Graph embedding is a method for *representation learning*; it computes feature vectors such that their inner products predict link weights as shown in Figure 1.3. Although the above Mahalanobis similarity learning corresponds to finding linear transformation $\boldsymbol{x} \mapsto \boldsymbol{y} := \boldsymbol{\theta}^\top \boldsymbol{x}$ of data vectors, we may employ highly non-linear neural networks $\boldsymbol{x} \mapsto \boldsymbol{y} := \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x})$ instead, for enhancing its expressive power; we mainly consider the neural network (NN)-based graph embedding throughout this thesis. More specifically, this thesis considers graph embedding in line with Tang et al. (2015), that leverages a conditional probability model of the link weights given data vectors, that is

$$w_{ij} \mid \boldsymbol{x}_i, \boldsymbol{x}_j \overset{\text{indep.}}{\sim} \mathrm{Bernoulli}(\sigma(\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_j) \rangle)), \quad (1 \leq i < j \leq n)$$

where $\mathrm{Bernoulli}(\mu)$ represents Bernoulli distribution whose expectation is specified by $\mu \in [0, 1]$, and $\sigma(z) := (1 + \exp(-z))^{-1}$ represents sigmoid function. Once the NN is trained by maximizing the conditional likelihood of the link weights by stochastic optimization algorithms (Goodfellow, Bengio, and Courville, 2016), we may obtain the feature vectors

$$\boldsymbol{y}_i = \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i) \quad (i = 1, 2, \ldots, n).$$

The obtained feature vectors are,

(i) containing richer information than the original data vectors, as they are computed from not only the data vectors but also observed link weights, and

(ii) computationally tractable, as the user-specified dimension K of the feature vectors is usually smaller than the dimension p of the original data vectors.

Graph embedding can be regarded as a kind of dimensionality reduction; the obtained feature vectors can be used for a variety of downstream tasks in

machine learning and statistics, such as clustering and discriminant analysis. See Section 2.2 in Chapter 2 for more details on graph embedding. Although the existing graph embeddings achieved success from application perspectives, several challenges remain; we study three different challenges in this thesis, towards a theoretically-guaranteed unified framework for NN-based graph embedding.

The three challenges are summarized in Section 1.2, and subsequently, our contributions to these challenges are shown in Section 1.3.

1.2 Challenges

Problems considered in this thesis can be classified into three independent challenges, which are summarized in the following Section 1.2.1–1.2.3.

1.2.1 Multi-view Graph Embedding

Graph embedding considers a single type of data vectors. For instance, co-author network and friend network shown in Figure 1.1(A) and 1.1(B) employ properties of people and researchers as their data vectors, respectively; the dimensionality of all the data vectors is the same for each example. However, real-world graphs may contain various types of data vectors. A dataset consisting of users, images, and text tags obtained from social networking services, is a typical example as shown in Figure 1.4. These different types of data vectors are especially called *multi-view* data vectors, where the data type is called *domain* or *view*.

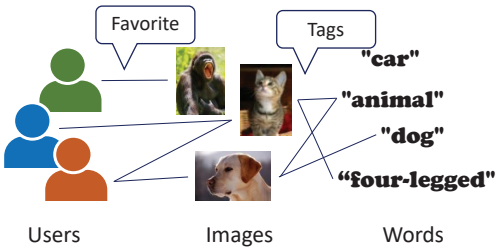


FIGURE 1.4: Multi-view data vectors with the graph-structured association. The dimensionality of the data vector depends on the view to which the data vector belongs.

Denoting the number of views and the view index by $D \in \mathbb{N}$ and $d \in \{1, 2, \ldots, D\}$, the dimension of each data vector depends on the view d to which the data vector belongs.

We here consider the case $D = 2$ that we have observed images ($d = 1$), text tags ($d = 2$) and their graph-structured associations. Then, data vectors for images and their text tags may be obtained from different external sources; dimensionalities of the vectors may be different depending on the view. Thus standard graph embedding cannot be applied to the multi-view data vectors. Although canonical correlation analysis (CCA; Hotelling, 1936)

may be employed for dealing with such a $(D =)$2-view data vectors, CCA assumes that the 2-view data vectors possess one-to-one relationship; graph-structured associations across views cannot be considered. For solving the issue, Shimodaira (2016), which is mathematically equivalent to Nori, Bollegala, and Kashima (2012), extends CCA so that the graph-structured associations can be considered. However, transformations used in these approaches are limited to linear setting; the lack of expressive power in multi-view graph embedding is the first challenge we consider in this thesis.

1.2.2 Expressive Power of NN-Based Graph Embedding

NN-based graph embedding considers applying a neural network to the data vectors, i.e., $\boldsymbol{y}_i = \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i)$, and their inner product $\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle$ is leveraged for predicting the link weight $w_{ij} \geq 0$. Whereas many of existing studies consider the inner-product for measuring the similarity (Tang, Sussman, and Priebe, 2013; Perozzi, Al-Rfou, and Skiena, 2014; Mikolov et al., 2013), a recent study called Poincaré embedding employs negative Poincaré distance instead. Poincaré embedding is known to efficiently embed the tree-structured associations in low-dimensions. Obviously, such a similarity is straightforwardly generalized to arbitrary kernel g, where the function in the form of $g(\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_j))$ shown in Figure 1.5 is also called *siamese neural network* (Bromley et al., 1994).

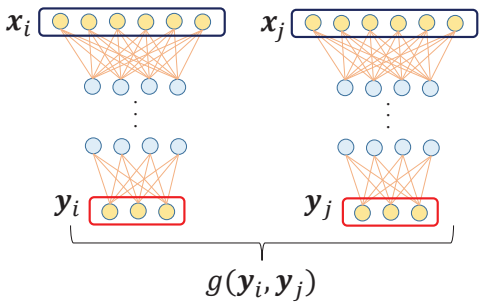


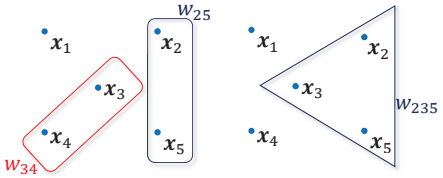
FIGURE 1.5: Siamese NN is computed via feature vectors $\boldsymbol{y}_i = \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{y}_j = \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_j)$; theories for the standard NN cannot be directly applied to the siamese NN.

Whereas the expressive power of the standard NN has attracted much attention for long decades (Cybenko, 1989; Funahashi, 1989; Yarotsky, 2017), the theorems cannot be used for the siamese neural network, that finally applies a kernel function to measure the similarity between the NN outputs $\boldsymbol{y}_i, \boldsymbol{y}_j$; the lack of theoretical evaluation for the siamese-NN is the second challenge considered in this thesis.

1.2.3 Hyper-relational Learning

Existing graph embedding methods are limited to considering the link weight defined between only two nodes, despite the fact that link weights can be similarly defined for a set of three or more nodes. The set of $U \in \mathbb{N}$ vectors

is called U-tuple, and the weight defined for the tuple is called a *hyperlink weight*.



(A) $U = 2$ (B) $U = 3$

FIGURE 1.6: Link regression including graph embedding considers (A) link weights defined between only two nodes. However, it cannot be applied to (B) hyperlink weights defined among more than 2 nodes.

A hyperlink weight appears in many practical situations; in a friend network, the existence of a group to which all the selected $U(\geq 2)$ people belong should be expressed as a binary hyperlink weight. Similarly, the number of co-authored papers written by all the selected $U(\geq 2)$ people in a co-authorship network should be represented as hyperlink weights assuming values in non-negative integers. The third challenge is that the existing link regression, including graph embedding, cannot address such complicated hyperlink weights.

Although one of the main concerns here is hyper-relational learning, objective function plays an indispensable role in hyperlink regression (including graph embedding). As well as the graph embedding methods, logistic loss and Kullback-Leibler (KL) divergence, where minimizing KL-divergence corresponds to maximizing the likelihood based on Poisson distribution, may be employed. However, there is no need to limit the choice of loss functions, as it may result in a missed opportunity to improve the hyperlink regression performance. Clarifying a class of functions, that has desirable theoretical properties in hyperlink regression setting, is also a challenge considered in this thesis.

1.3 Contributions

For solving the above challenges described in Sections 1.2.1–1.2.3, we provide the following contributions in this thesis.

(i) **Multi-view extension** of NN-based graph embedding, that is called probabilistic multi-view graph embedding (PMvGE), is proposed. PMvGE stands on a simple idea: considering multi-view data vectors, where $\boldsymbol{x}_i$ belongs to view d_i, PMvGE simply extends the conditional probability model of link weights as

$$w_{ij} \mid \boldsymbol{x}_i, \boldsymbol{x}_j \overset{\text{indep.}}{\sim} \mathrm{Po}\left(\alpha^{(d_i,d_j)} \exp(\langle \boldsymbol{f}_{\mathrm{NN}}^{(d_i)}(\boldsymbol{x}_i), \boldsymbol{f}_{\mathrm{NN}}^{(d_j)}(\boldsymbol{x}_j)\rangle)\right),$$

where $\mathrm{Po}(\mu)$ here denotes Poisson distribution whose expectation is $\mu \in \mathbb{R}_{\geq 0}$. Whereas existing graph embedding applies only one NN f_{NN} to all the data vectors, the multi-view data vector $x_i \in \mathbb{R}^{p_{d_i}}$ is transformed by a NN $f_{\mathrm{NN}}^{(d_i)}$ depending on the view d_i. Furthermore, we also prove that the proposed PMvGE at least approximately generalizes the linear CDMCA (Shimodaira, 2016), by considering the quadratic approximation. These contributions are described in Chapter 3 of this thesis.

(ii) **Expressive power** of NN-based graph embedding using inner product similarity (IPS)

$$\langle f_{\mathrm{NN}}(x_i), f_{\mathrm{NN}}(x_j) \rangle$$

(where $x \mapsto y = f_{\mathrm{NN}}(x)$) is first studied. More specifically, we prove that IPS is limited to approximate positive-definite (PD) similarities; more expressive similarity called shifted-IPS (SIPS)

$$\langle f_{\mathrm{NN}}(x_i), f_{\mathrm{NN}}(x_j) \rangle + u_{\mathrm{NN}}(x_i) + u_{\mathrm{NN}}(x_j)$$

(where $x \mapsto y = (f_{\mathrm{NN}}(x), u_{\mathrm{NN}}(x))$) is proposed for graph embedding, and we prove that SIPS can approximate a wider class of functions, called conditionally-PD similarities. A more expressive inner-product difference similarity (IPDS) and its approximation theorem are also provided; IPDS is in fact capable of approximating arbitrary similarities, i.e., continuous and symmetric functions. These contributions are described in Chapter 4 of this thesis.

(iii) **Hyperlink extension** of link regression is considered. We propose hyperlink regression using Bregman divergence, that is especially called *Bregman Hyperlink Regression (BHLR)*. Similarly to the existing link regression methods, that predict link weight w_{ij} by a function $\mu_\theta(x_i, x_j)$ of the corresponding data vectors, BHLR predicts hyperlink weight $w_{i_1,i_2,\ldots,i_U}$ by a function $\mu_\theta(x_{i_1}, x_{i_2}, \ldots, x_{i_U})$ trained via Bregman divergence.

Whereas BHLR encompasses a variety of existing methods including graph embedding, and tuples $X_i, X_{i'}$ may be dependent as they may share some data vectors therein, meaning that standard asymptotic theories cannot be applied to, we prove that general BHLR possesses the following two desirable properties: (P-1) statistical estimation consistency, and (P-2) computational tractability, for general $U \in \mathbb{N}$ and general Bregman divergence. Especially for (P-2), we propose a hyperlink extension of negative sampling used in skip-gram (a.k.a. word2vec; Mikolov et al., 2013) with its theoretical guarantee. These contributions are described in Chapter 5 of this thesis.

1.4 Organization

Organization of this thesis is summarized as follows: In Chapter 2, we describe preliminaries, especially background knowledge about neural network,

graph embedding, multi-view methods, and their related studies. As described in the previous Section 1.3, we consider a multi-view extension of NN-based graph embedding in Chapter 3, expressive power of NN-based graph embedding and the proposed expressive similarities in Chapter 4, and Bregman hyperlink regression and its theoretical properties in Chapter 5. This thesis is concluded in Chapter 6.

2 Preliminaries

In this chapter, we describe background knowledge and the preliminaries of this thesis. Particularly, we describe neural network and its expressive power in Section 2.1, and graph embedding in Section 2.2. For comparison purposes, graph neural network (GNN) is also explained in Section 2.3, though GNN is different from the graph embedding considered in this thesis. Finally, multi-view representation learning is summarized in Section 2.4.

Throughout this thesis, $\mathbb{1}(\cdot)$ represents the indicator function, $[n]$ denotes the set $\{1, 2, \ldots, n\}$, $\mathbf{1}_n := (1, 1, \ldots, 1) \in \mathbb{R}^n$ and $\boldsymbol{I}_n$ represents the $n \times n$ identity matrix, for $n \in \mathbb{N}$.

2.1 Neural Networks

In this section, we first define neural networks in Section 2.1.1, and subsequently, we describe universal approximation theorems in Section 2.1.2.

2.1.1 Definition

Given $L, K, p \in \mathbb{N}$ and p-dimensional data vector $\boldsymbol{x} \in \mathbb{R}^p$, vector-valued L-hidden layer neural network $\boldsymbol{f}_{\mathrm{NN}} : \mathbb{R}^p \to \mathbb{R}^K$, whose number of hidden units in ℓth layer is specified by $d_\ell \in \mathbb{N}$ $(\ell = 1, 2, \ldots, L-1)$, is defined as

$$\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}) := \boldsymbol{A}^{(L)} \tilde{\boldsymbol{f}}^{(L)}(\boldsymbol{x}). \tag{2.1}$$

$\tilde{\boldsymbol{f}}^{(\ell)}$ $(\ell = 0, 1, 2, \ldots, L)$ is recursively defined as

$$\tilde{\boldsymbol{f}}^{(\ell)}(\boldsymbol{x}) := \begin{cases} \boldsymbol{\sigma}(\boldsymbol{A}^{(\ell-1)} \tilde{\boldsymbol{f}}^{(\ell-1)}(\boldsymbol{x}) + \boldsymbol{b}^{(\ell-1)}) & (\ell = 1, 2, \ldots, L) \\ \boldsymbol{x} & (\ell = 0) \end{cases},$$

where $\boldsymbol{A}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ $(\ell = 0, 1, 2, \ldots, L;\ d_0 := p, d_{L+1} := K)$ are parameter matrices and $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$ $(\ell = 0, 1, \ldots, L-1)$ are parameter vectors to be estimated. $\boldsymbol{\sigma}(\cdot)$ applies a user-specified activation function $\sigma(\cdot)$ element-wise; the sigmoid activation function $\sigma(z) := (1 + \exp(-z))^{-1}$ or the Rectified Linear Unit (ReLU) $\sigma(z) := \max\{0, z\}$ is employed typically. Whereas the numbers of hidden untis $d_1, d_2, \ldots, d_L$ regulate the width of the neural network $\boldsymbol{f}_{\mathrm{NN}}$, its depth is parametrized by the user-specified parameter $L \in \mathbb{N}$; neural network with sufficiently large L is especially called *deep neural network (DNN)*, and analysis leveraging DNN is called *deep learning* (Goodfellow, Bengio, and Courville, 2016).

Rigorously speaking, (2.1) is a special case of neural network, and it is especially called multilayer perceptron (MLP). There exist a variety of other possible forms for neural networks, such as convolutional NN (Goodfellow, Bengio, and Courville, 2016). However, we mainly consider only the MLP in this thesis as its theories have been well developed, as shown in the following Section 2.1.2; throughout this thesis, the word "neural network" indicates the MLP (2.1) unless otherwise noted.

Considering the linear activation $\sigma(z) := z$, neural network obviously includes the linear transformation $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ as a special case, with the matrix $\boldsymbol{A} := \prod_{\ell=0}^{L} \boldsymbol{A}^{(\ell)} \in \mathbb{R}^{K \times p}$ and the vector $\boldsymbol{b} := \sum_{i=0}^{L-1} \{\boldsymbol{A}^{(L)} \boldsymbol{A}^{(L-1)} \cdots \boldsymbol{A}^{(i+1)} \boldsymbol{b}^{(i)}\} \in \mathbb{R}^{K}$. Note that the neural network with a small number of hidden units becomes rank-reduced linear transformation, as the rank of the matrix $\boldsymbol{A}$ can be restricted to be smaller than $\min\{d_1, d_2, \ldots, d_L\}$.

2.1.2 Universal Approximation Theorem

Unlike the neural network with linear activation, which reduces to the linear transformation, it is widely known that the neural network equipped with non-linear activation function and the sufficiently large number of hidden units has high expressive power, in the sense that any of continuous function can be approximated arbitrary well. Considering the simplest case $K = 1$, to the best of author's knowledge, Cybenko (1989) and Funahashi (1989) are the earliest works proving that $(L =)1$-hidden layer neural network can approximate any continuous function over a compact set $\mathcal{X} := [-M, M]^p$ $(M > 0, p \in \mathbb{N})$. This theorem is especially called universal approximation theorem (UAT). Although the UAT is proved for $(L =)1$-hidden layer NN, its generalization to arbitrary $L \in \mathbb{N}$, i.e., the adaptation to DNN, has attracted much attention these days, by virtue of the remarkable success and recent rise of deep learning.

Considering a function class

$$\begin{aligned} \mathfrak{S}_\alpha(W, K) := \{ f_{\mathrm{NN}} : \mathcal{X} \to \mathbb{R}^K \mid \text{neural network } f_{\mathrm{NN}} \text{ has} \\ W \text{ weights with depth } L = O((W/K)^\alpha) \} \end{aligned} \tag{2.2}$$

for some set $\mathcal{X} \subset \mathbb{R}^p$ and a user-specified parameter $\alpha \in [0, 1]$, where the activation function is specified by sigmoid or ReLU, Yarotsky (2017) and Yarotsky (2018) proved the UAT for real-valued DNN ($K = 1$). The following Theorem 2.1 summarizes the assertion. Note that the set $\mathfrak{S}_\alpha(W, 1)$ with $\alpha = 0$ corresponds to the set of constant-depth wide neural network, whereas $\alpha = 1$ represents the set of constant-width deep neural network; Theorem 2.1 simultaneously proves UATs for both shallow and deep neural networks.

Theorem 2.1: UAT for real-valued DNN ($K = 1$; Yarotsky, 2018)

For $\mathcal{X} = [-M, M]^p$, $M > 0$, $p \in \mathbb{N}$ and $\alpha \in [0, 1]$, we consider the set of real-valued neural networks $\mathfrak{S}_\alpha(W, 1)$ for $W \in \mathbb{N}$, that is defined in (2.2). Let $\omega(f; r) := \max\{|f(\boldsymbol{x}) - f(\boldsymbol{x}')| : \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \leq r\}$ be

the modulus of continuity. Then, there exist $a, c \in \mathbb{R}$ such that

$$\inf_{f_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W,1)} \|f_* - f_{\mathrm{NN}}\|_\infty \leq a\omega(f_*; cW^{-\frac{1+\alpha}{p}})$$

holds for any real-valued continuous function $f_* : \mathcal{X} \to \mathbb{R}$. Here, $\|f\|_\infty := \sup_{\boldsymbol{x} \in \mathcal{X}} |f(\boldsymbol{x})|$ represents infinity norm.

The above Theorem 2.1 evaluates the error rate of the real-valued neural network f_{NN} for approximating the continuous function f_*; the obtained upper-bound depends on the modulus of continuity. If f_* is also assumed to be continuously differentiable, more intuitive Corollary 2.1 is obtained as an immediate consequence of Theorem 2.1.

Corollary 2.1

Symbols are the same as those of Theorem 2.1. Assuming that a fixed function f_* is continuously differentiable over $\mathcal{X}$, we have

$$\inf_{f_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W,1)} \|f_* - f_{\mathrm{NN}}\|_\infty = O(W^{-\frac{1+\alpha}{p}})$$

as $W \to \infty$.

Proof. Proof is based on the intermediate value theorem. For $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ satisfying $\|\boldsymbol{x} - \boldsymbol{x}'\| \leq r$, there exists $\boldsymbol{x}_0 \in \mathcal{X}$ such that $f_*(\boldsymbol{x}) - f_*(\boldsymbol{x}') = \frac{\partial f_*(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\boldsymbol{x}_0}(\boldsymbol{x} - \boldsymbol{x}')$. Since $b := \sup_{\boldsymbol{x} \in \mathcal{X}} \|\partial f_*(\boldsymbol{x})/\partial \boldsymbol{x}\|$ is bounded because of the continuity of the first-order derivative $\partial f_*(\boldsymbol{x})/\partial \boldsymbol{x}$, Cauchy-Schwarz inequality indicates

$$|f_*(\boldsymbol{x}) - f_*(\boldsymbol{x}')| \leq \left\| \frac{\partial f_*(\boldsymbol{x}_0)}{\partial \boldsymbol{x}} \right\|_2 \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \leq br.$$

Thus we have $\omega(f_*; r) \leq br$, indicating

$$a\omega(f_*; cW^{-\frac{1+\alpha}{p}}) \leq abcW^{-\frac{1+\alpha}{p}}. \tag{2.3}$$

Substituting (2.3) into Theorem 2.1 proves the corollary. □

Corollary 2.1 proves that the approximation error rate of neural network is upper-bounded by $O(W^{-\frac{1+\alpha}{p}})$; the error converges to 0 as the number of weights increases, i.e., $W \to \infty$. Therefore, sufficiently large neural network f_{NN} is in general capable of approximating any continuously differentiable function f_*, meaning that neural network has high approximation capability.

Above Corollary 2.1 for real-valued NN ($K = 1$) is straightforwardly generalized to vector-valued NN ($K \in \mathbb{N}$) as shown in the following Lemma 2.1.

Lemma 2.1: UAT for vector-valued DNN ($K \in \mathbb{N}$)

Symbols and assumptions are the same as those of Theorem 2.1. Let $f_* : \mathcal{X} \to \mathbb{R}^K$ be a vector-valued continuously differentiable function over $\mathcal{X}$ such that $\sup_{k\in\{1,\dots,K\},x\in\mathcal{X}} \|\partial f_{*k}(x)/\partial x\|_2 \le b$ for some b which does not depend on K. Then, as $W/K \to \infty$, we have

$$\inf_{f_{\mathrm{NN}}\in\mathfrak{S}_\alpha(W,K)} \|f_* - f_{\mathrm{NN}}\|_{2,\infty} = O(K^{\frac{1}{2}+\frac{1+\alpha}{p}} W^{-\frac{1+\alpha}{p}}),$$

where $\|f\|_{2,\infty} := \sup_{x\in\mathcal{X}} \|f(x)\|_2$.

Proof. Proof is based on applying Corollary 2.1 to each of K output units of f_*. We consider K real-valued neural networks of depth $L = O((W/K)^\alpha)$

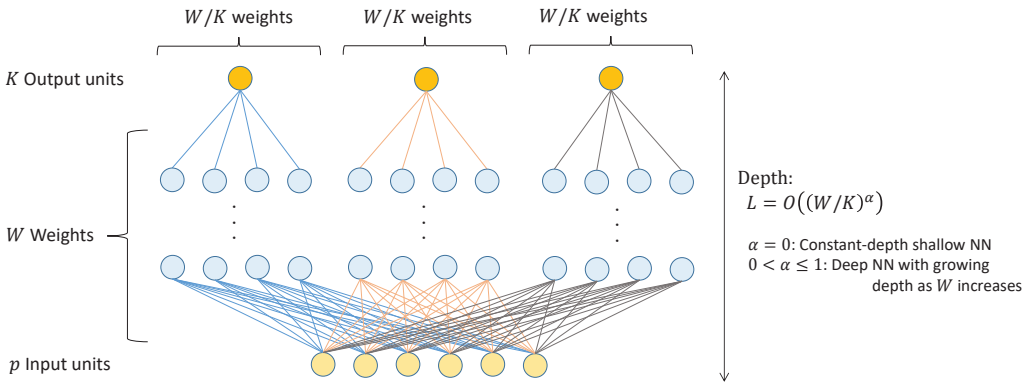


FIGURE 2.1: A structure of vector-valued neural network $f_{\mathrm{NN}} : \mathbb{R}^p \to \mathbb{R}^K$ having W weights. We allocate W/K weights to each output unit, so that weights are not shared by the K output units. In practice, internal units are often shared by the output units, but we consider the above structure for showing the upper bound of the approximation error.

with W/K weights as shown in Fig. 2.1. Since such NNs are included in $\mathfrak{S}_\alpha(W,K)$, we have

$$\begin{aligned}\inf_{f_{\mathrm{NN}}\in\mathfrak{S}_\alpha(W,K)} \|f_* - f_{\mathrm{NN}}\|_{2,\infty} &= \inf_{f_{\mathrm{NN}}\in\mathfrak{S}_\alpha(W,K)} \sup_{x\in\mathcal{X}} \|f_* - f_{\mathrm{NN}}\|_2 \\ &\le \Big(\sum_{k=1}^K \inf_{f_k\in\mathfrak{S}_\alpha(W/K,1)} |f_{*k} - f_k|^2\Big)^{1/2},\end{aligned}$$

where $f_*(x) = (f_{*1}(x), f_{*2}(x), \dots, f_{*K}(x))$, $f_{\mathrm{NN}}(x) = (f_1(x), f_2(x), \dots, f_K(x))$. We apply Corollary 2.1 with W/K weights to each f_{*k}, where the same bound b is used in (2.3). Then the error is bounded by $\sqrt{K} \times abc(W/K)^{-\frac{1+\alpha}{p}} = O(K^{\frac{1}{2}+\frac{1+\alpha}{p}} W^{-\frac{1+\alpha}{p}})$. □

2.2 Graph Embedding

We here first summarize the data structure considered throughout this thesis and the purpose of graph embedding. Subsequently, we describe existing graph embedding methods in Sections 2.2.1 and 2.2.2.

Data structure

Assuming that $v_1, v_2, \ldots, v_n$ are nodes of a graph, observed *link weight*

$$w_{ij} (= w_{ji}) \geq 0$$

represents the strength of association between two nodes v_i, v_j. Typically the weight is specified by a weighted adjacency matrix $\boldsymbol{W} = (w_{ij})$ of the graph. As introduced in Section 1.1, we may also consider observed *data vectors*

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$$

representing individual properties of nodes $v_1, v_2, \ldots, v_n$, in addition to the link weights. Friend network shown in Section 1.1 is a typical example of such an attributed graph; therein, individual properties of people such as gender, age, and birthplace specify each of the data vectors, and their friendship specifies the link weights.

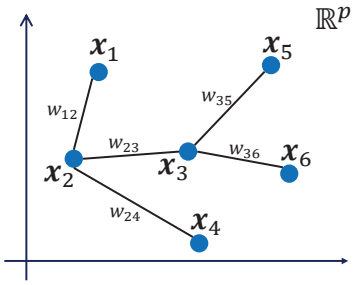


FIGURE 2.2: Data structure considered in this thesis; dataset consists of data vectors $\{\boldsymbol{x}_i\}$ and link weights $\{w_{ij}\}$. $\boldsymbol{x}_i \in \mathbb{R}^p$ represents individual properties of node v_i, and $w_{ij} (= w_{ji}) \geq 0$ represents the association strength between nodes v_i, v_j.

Although both data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ and their link weights $\{w_{ij}\}$ are assumed to be separately observed, either of them may not be obtained in practical cases; even in this case, we may consider alternatives for the unobserved data vectors (or link weights), that are defined as follows.

- **If link weights $\{w_{ij}\}$ are not observed:** We here consider computing alternatives of unobserved link weights $\boldsymbol{W} = (w_{ij})$ from data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$. Chung (1997) and Belkin and Niyogi (2001) consider

$$w_{ij} := \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 / s^2) \mathbf{1}(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \leq \varepsilon) \tag{2.4}$$

  for all $1 \leq i, j \leq n$, where $s > 0, \varepsilon \geq 0$ are user-specified parameters. As

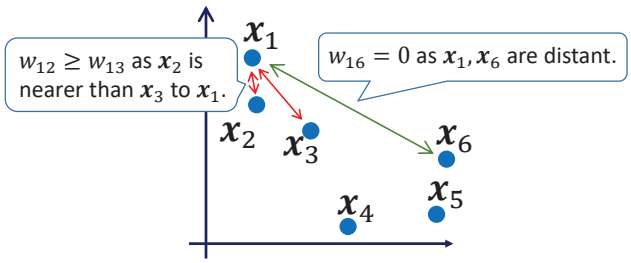


FIGURE 2.3: $w_{ij} \geq 0$ computed from data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_6 \in \mathbb{R}^6$. The weight between nearer vectors is greater than that between distant vectors, and the weight is specified by 0 if the distance between two vectors is $> \varepsilon$.

(2.4) is large/small if $\boldsymbol{x}_i, \boldsymbol{x}_j$ are near/distant each other, (2.4) is expected to represent the association strength as shown in Figure 2.3. Thus (2.4) can be employed as an alternative of the link weights. Although (2.4) is mainly employed in existing studies, that are in line with Laplacian eigenmaps (Belkin and Niyogi, 2001) described later, the function may be replaced with arbitrary kernel $\mathcal{K} : \mathcal{X}^2 \to \mathbb{R}$, i.e., continuous function satisfying symmetry $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x}', \boldsymbol{x})$; Guo, Gao, and Kwan (2006) employs

$$w_{ij} := \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{2.5}$$

- **If data vectors $\{\boldsymbol{x}_i\}$ are not observed:** Conversely, we here consider computing alternatives of unobserved data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ from link weights $\boldsymbol{W} = (w_{ij})$. One of the simplest approaches for computing alternative data vectors is called 1-hot encoding, that specifies the data vectors by

$$\boldsymbol{x}_i := (\underbrace{0, \ldots, 0}_{i-1}, \underset{i\text{th}}{1}, \underbrace{0, \ldots, 0}_{n-i}) \in \{0, 1\}^n \quad (p := n) \tag{2.6}$$

whose ith entry is 1 and the remaining entries are 0, for $i = 1, 2, \ldots, n$. It is also known as dummy coding or dummy variable (Hastie, Tibshirani, and Friedman, 2009), especially in statistics literature.

Another possible alternative is

$$\boldsymbol{x}_i = (w_{i1}, w_{i2}, \ldots, w_{in}) \in \mathbb{R}_{\geq 0}^n \quad (p := n),$$

representing the association structure of the node v_i to all the other nodes, as illustrated in Figure 2.4.

Furthermore, data vector $\boldsymbol{x}_i$ may also be specified by a vector aligning several real-valued topological properties of the graph, such as node degree, clustering coefficient, and between centrality shown in Prado et al. (2013) Section 2.

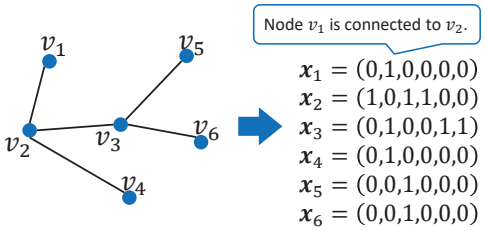


FIGURE 2.4: Data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_6 \in \mathbb{R}^6$ are computed from binary link weights $w_{ij} \in \{0,1\}$.

Therefore, the data structure we consider in this thesis encompasses a broad range of real-world datasets.

Purpose of graph embedding

Graph embedding aims at computing new feature vector $\boldsymbol{y}_i \in \mathbb{R}^K$ for node v_i $(i = 1,2,\ldots,n)$, by leveraging the attributed graph consisting of the data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ and the link weights $\{w_{ij}\}$. Typically, a function $\boldsymbol{f} : \mathbb{R}^p \to \mathbb{R}^K$ such as neural network trained with link weights $\{w_{ij}\}$ is applied to the data vectors. Then, feature vectors are obtained as

$$\mathbb{R}^p \ni \boldsymbol{x}_i \mapsto \boldsymbol{y}_i := \boldsymbol{f}(\boldsymbol{x}_i) \in \mathbb{R}^K, \quad (i = 1,2,\ldots,n). \tag{2.7}$$

The obtained feature vectors $\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathbb{R}^K$ are expected to be

(i) containing richer information than the original data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$, as the function $\boldsymbol{f}$ is trained with not only the data vectors but also observed link weights, and

(ii) computationally tractable, as the user-specified dimension K of the feature vectors is usually smaller than the dimension p of the original data vectors.

These feature vectors $\{\boldsymbol{y}_i\}_{i=1}^n$ can be used for a variety of downstream tasks in machine learning and statistics, such as clustering and discriminant analysis.

This thesis mainly considers graph embedding methods in line with large-scale information network embedding (LINE; Tang et al., 2015), which is shown in the following Section 2.2.2. However, graph neural network (GNN), that gathers attention these days, is different from the graph embedding methods considered in this thesis; we further summarize the GNN and the difference from LINE in Section 2.3.

2.2.1 Spectral Methods

Laplacian eigenmaps

Considering the observed data vectors and link weights, one of the most classical and widely-applicable approaches for graph embedding is Laplacian

eigenmaps (Belkin and Niyogi, 2001; Belkin and Niyogi, 2003), that is also known as spectral graph embedding (Chung, 1997).

By leveraging the link weights $\{w_{ij}\}$, Laplacian eigenmaps computes feature vectors $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n \in \mathbb{R}^K$ by minimizing an objective function

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2, \tag{2.8}$$

under a quadratic constraint $\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\boldsymbol{y}_i\boldsymbol{y}_i^\top = \boldsymbol{I}_K$. The constraint prevents the obtained feature vectors from being trivial solution, and the constrained optimization problem can be easily solved by the following way.

By leveraging the degree matrix $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{W}\boldsymbol{1}_n)$, the objective function becomes (2.8) $= \mathrm{tr}\boldsymbol{Y}^\top(\boldsymbol{M} - \boldsymbol{W})\boldsymbol{Y}$ and the quadratic constraint is $\boldsymbol{Y}^\top\boldsymbol{M}\boldsymbol{Y} = \boldsymbol{I}$; the above Laplacian eigenmaps is one of the simplest cases of quadratically constrained quadratic programming (QCQP; Boyd and Vandenberghe, 2004), thus it can be simply solved via eigen-decomposition. More specifically, the solution is obtained by following Lemma 2.2 by specifying $\boldsymbol{A} := \boldsymbol{M} - \boldsymbol{W}, \boldsymbol{B} := \boldsymbol{M}, \boldsymbol{C} := \boldsymbol{Y}$.

Lemma 2.2: Solution of a simple QCQP

Let $n, m \in \mathbb{N}, m \leq n$, and let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times n}$ be symmetric matrices, where $\boldsymbol{B}$ is also positive-definite, i.e., $\boldsymbol{x}^\top\boldsymbol{B}\boldsymbol{x} > 0$ for any non-zero vector $\boldsymbol{x} \in \mathbb{R}^n$. Then, a solution of

$$\text{minimize: } \mathrm{tr}\boldsymbol{C}^\top\boldsymbol{A}\boldsymbol{C} \quad \text{w.r.t. } \boldsymbol{C} \in \mathbb{R}^{n\times m} \quad \text{such that } \boldsymbol{C}^\top\boldsymbol{B}\boldsymbol{C} = \boldsymbol{I}_m \tag{2.9}$$

is $\hat{\boldsymbol{C}} = \boldsymbol{B}^{-1/2}(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m)$, where $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$ are eigenvectors of $\boldsymbol{B}^{-1/2}\boldsymbol{A}\boldsymbol{B}^{-1/2}$ whose corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ are in ascending order. Furthermore, the minimum value is

$$\mathrm{tr}\hat{\boldsymbol{C}}^\top\boldsymbol{A}\hat{\boldsymbol{C}} = \sum_{k=1}^{m}\lambda_k. \tag{2.10}$$

Proof. See e.g., Belkin and Niyogi (2001) and Yan et al. (2007); we here provide a brief proof of Lemma 2.2. We consider a set $\mathcal{O}(n, m) := \{\boldsymbol{U} \in \mathbb{R}^{n\times m} \mid \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_m\}$. Due to the constraint $\boldsymbol{C}^\top\boldsymbol{B}\boldsymbol{C} = \boldsymbol{I}_m$, it holds that $\boldsymbol{U} := \boldsymbol{B}^{1/2}\boldsymbol{C} \in \mathcal{O}(n, m)$. Substituting $\boldsymbol{C} = \boldsymbol{B}^{-1/2}\boldsymbol{U}$ into the objective function leads to

$$\mathrm{tr}\boldsymbol{C}^\top\boldsymbol{A}\boldsymbol{C} = \mathrm{tr}\boldsymbol{U}^\top\{\boldsymbol{B}^{-1/2}\boldsymbol{A}\boldsymbol{B}^{-1/2}\}\boldsymbol{U}.$$

It is obviously minimized if $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m)$ are eigenvectors of the matrix $\boldsymbol{B}^{-1/2}\boldsymbol{A}\boldsymbol{B}^{-1/2}$ whose eigenvalues are in ascending order; the solution of (2.9) is obtained by $\hat{\boldsymbol{C}} = \boldsymbol{B}^{-1/2}\boldsymbol{U}$. Then, (2.10) takes minimum value

$$\mathrm{tr}\hat{\boldsymbol{C}}^\top\boldsymbol{A}\hat{\boldsymbol{C}} = \mathrm{tr}(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m)^\top\{\boldsymbol{B}^{-1/2}\boldsymbol{A}\boldsymbol{B}^{-1/2}\}(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m)$$

$$= \sum_{k=1}^{m} \boldsymbol{u}_k^\top \{\boldsymbol{B}^{-1/2}\boldsymbol{A}\boldsymbol{B}^{-1/2}\}\boldsymbol{u}_k = \sum_{k=1}^{m} \lambda_k.$$

□

Considering Lemma 2.2, feature vectors obtained by Laplacian eigenmaps are based on eigenvectors of

$$\boldsymbol{M}^{-1/2}(\boldsymbol{M} - \boldsymbol{W})\boldsymbol{M}^{-1/2} = \boldsymbol{I}_n - \boldsymbol{M}^{-1/2}\boldsymbol{W}\boldsymbol{M}^{-1/2};$$

they are also eigenvectors of the matrix

$$\boldsymbol{M}^{-1/2}\boldsymbol{W}\boldsymbol{M}^{-1/2}. \tag{2.11}$$

Although link weights $\boldsymbol{W} = (w_{ij})$ in (2.11) is normalized by the degree matrix $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{W}\boldsymbol{1}_n)$, eigendecomposition of the (unnormalized) weights $\boldsymbol{W}$ can be considered, and it is also referred to as a case of matrix factorization (Cichocki et al., 2009).

Locality preserving projections

Locality preserving projections (LPP; He and Niyogi, 2004; Yan et al., 2007) further imposes an additional linear constraint

$$\boldsymbol{y}_i = \boldsymbol{A}^\top \boldsymbol{x}_i$$

to Laplacian eigenmaps, and estimate the matrix $\boldsymbol{A} \in \mathbb{R}^{p\times K}$ instead of estimating $\boldsymbol{Y} = (\boldsymbol{y}_1^\top, \boldsymbol{y}_2^\top, \ldots, \boldsymbol{y}_n^\top)^\top$ directly. Considering the relation $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{A}$ for $\boldsymbol{X} = (\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top, \ldots, \boldsymbol{x}_n^\top)^\top$, the minimization problem of Laplacian eigenmaps is then reduces to minimizing $\mathrm{tr}\boldsymbol{A}^\top\{\boldsymbol{X}^\top(\boldsymbol{M} - \boldsymbol{W})\boldsymbol{X}\}\boldsymbol{A}$ with respect to the matrix $\boldsymbol{A} \in \mathbb{R}^{p\times K}$, under a quadratic constraint $\boldsymbol{A}^\top\{\boldsymbol{X}^\top\boldsymbol{M}\boldsymbol{X}\}\boldsymbol{A} = \boldsymbol{I}_K$; the minimization problem is yet in the form of QCQP. Thus it can be solved by Lemma 2.2.

Although these graph embedding methods are widely developed so far, they rely on eigendecomposition, whose computational complexity $O(n^3)$ increases rapidly as the number of nodes n increases. Thus it is known that these graph embedding methods do not scale; **computational complexity** is an issue of the spectral graph embedding methods.

For reducing the computational cost, several attempts have been developed (Mantziou, Papadopoulos, and Kompatsiaris, 2013; Ravi and Diao, 2016), and more generally, studies on fast computation of eigendecomposition (Lanczos, 1950; Golub and Kahan, 1965; Hernandez, Roman, and Vidal, 2005) are also applicable. However, in the first place, the computational difficulty for solving the QCQP (2.9) is attributed to the quadratic constraint, that prevents feature vectors from being trivial solutions; the following structural deep network embedding incorporates autoencoder into Laplacian eigenmaps instead of imposing the quadratic constraint, and avoids eigendecomposition.

Structural deep network embedding (SDNE)

Structural deep network embedding (SDNE; Wang, Cui, and Zhu, 2016) incorporates autoencoder (Hinton and Salakhutdinov, 2006; Goodfellow, Bengio, and Courville, 2016) into Laplacian eigenmaps. SDNE trains two neural networks $f_{\mathrm{NN}} : \mathbb{R}^p \to \mathbb{R}^K$ and $g_{\mathrm{NN}} : \mathbb{R}^K \to \mathbb{R}^p$ called encoder and decoder, respectively, by minimizing an objective function

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 + \alpha \underbrace{\sum_{i=1}^{n} \|\boldsymbol{x}_i - g_{\mathrm{NN}}(\boldsymbol{y}_i)\|_2^2}_{\text{Reconstruction Loss}}$$

with a constraint $\boldsymbol{y}_i := f_{\mathrm{NN}}(\boldsymbol{x}_i)$ $(i = 1, 2, \ldots, n)$. $\alpha > 0$ is a user-specified parameter. Once the NNs $f_{\mathrm{NN}}, g_{\mathrm{NN}}$ are trained, the feature vectors are immediately obtained by applying f_{NN} to the data vectors. Gradient-based approaches (Rumelhart, Hinton, and Williams, 1987; Goodfellow, Bengio, and Courville, 2016) are typically employed for optimization therein. The first term of the objective function is obviously that of Laplacian eigenmaps defined in (2.8), and the second term represents the reconstruction loss for the autoencoder, that prevents the feature vectors from being trivial solutions. As the obtained feature vectors are computed by applying the NN f_{NN} to the data vectors, SDNE can be regarded as a non-linear NN variant of LPP.

The above methods rely on the objective function (2.8), that is expected to make feature vectors $\boldsymbol{y}_i, \boldsymbol{y}_j$ closer if the corresponding weight w_{ij} is positive. However, the function (2.8) is rather indirect to specify the relation between the link weights and the feature vectors; more direct specification may be considered, as we are already not in line with the spectral approaches once the neural network is incorporated. For specifying the relation more directly, large-scale information network embedding (LINE) simply considers a conditional probabilistic model of the link weights defined via feature vectors, and optimize the likelihood function. The similar model is considered in skip-gram (a.k.a. word2vec; Mikolov et al., 2013) and several existing studies (Perozzi, Al-Rfou, and Skiena, 2014; Dai et al., 2018); we mainly consider the LINE-based approaches in this thesis, but not the spectral approaches. We hereby summarize the LINE in the following Section 2.2.2.

2.2.2 Large-scale Information Network Embedding (LINE)

Assuming that link weights take binary values $\{0, 1\}$, large-scale information network embedding (LINE; Tang et al., 2015) considers a probabilistic model

$$w_{ij} \mid \boldsymbol{y}_i, \boldsymbol{y}_j \overset{\text{indep.}}{\sim} \text{Bernoulli}\left(\sigma(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle)\right) \quad (1 \leq i < j \leq n). \tag{2.12}$$

where $\sigma(z) := (1 + \exp(-z))^{-1}$ represents sigmoid function and $\text{Bernoulli}(\mu)$ denotes Bernoulli distribution whose expectation is $\mu \in [0, 1]$. The probabilistic model (2.12) is also known as latent position random graph (LPRG)

model considered in Tang, Sussman, and Priebe (2013) and Athreya et al. (2018) Definition 6, if $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ are latent variables. LPRG model is originated from the random dot product graph model (Young and Scheinerman, 2007) $w_{ij} \mid \boldsymbol{y}_i, \boldsymbol{y}_j \overset{\text{indep.}}{\sim} \text{Bernoulli}\left(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle\right)$.

LINE computes feature vectors $\boldsymbol{y}_i$ by maximizing the log-likelihood of link weights $\{w_{ij}\}$, that is

$$\sum_{1 \le i < j \le n} \{ w_{ij} \log \sigma(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle) + (1 - w_{ij}) \log \sigma(-\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle) \}. \tag{2.13}$$

Although LINE originally computes feature vectors $\{\boldsymbol{y}_i\}$ without leveraging data vectors $\{\boldsymbol{x}_i\}$, finding such feature vectors corresponds to training neural networks applied to 1-hot vectors (2.6). Thus data vectors are naturally incorporated to the above LINE formulation, by considering $\boldsymbol{y}_i = \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}_i)$; the NN $\boldsymbol{f}_{\text{NN}}$ is trained by maximizing (2.13). Once the NN is trained, the feature vector is obtained by

$$\boldsymbol{y}_i = \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}_i) \quad (i = 1, 2, \ldots, n).$$

As we may apply stochastic optimization methods, such as stochastic gradient descent (Goodfellow, Bengio, and Courville, 2016) for minimizing the objective function (2.13), LINE is not only intuitive but also computationally efficient graph embedding method. Furthermore, by virtue of the probabilistic formulation, the statistical estimation consistency can be proved, as shown in Chapter 5 for more general settings.

Although the above formulation leverages only the binary link weights, we may consider more general weights by replacing the distribution from Bernoulli to others. For instance, Chapters 3 and 4 in this thesis consider Poisson distribution instead, and Chapter 5 considers more general settings.

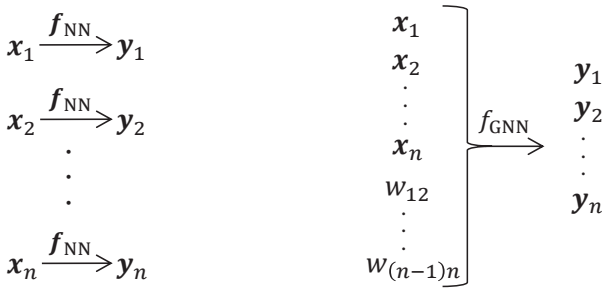
Although the above graph embedding methods employ standard NNs taking only one data vector $\boldsymbol{x}_i$ as its input, graph neural network (GNN), that takes attention these days, outputs n feature vectors $\{\boldsymbol{y}_i\}_{i=1}^n$ by taking all the data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ and link weights $\{w_{ij}\}_{i,j=1}^n$ as inputs simultaneously. In the following Section 2.3, we summarize the GNN for comparison purposes.

2.3 Graph Neural Network

Given a graph, whose nodes $v_1, v_2, \ldots, v_n$ are equipped with data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$ and the weighted adjacency matrix is denoted by $\boldsymbol{W} = (w_{ij})$, we here first define an index set

$$\mathcal{N}_i := \{ j \in \{1, 2, \ldots, n\} \setminus \{i\} \mid w_{ij} > 0 \}, \quad (i = 1, 2, \ldots, n).$$

Entries in the set $\mathcal{N}_i$ represents the neighbourhood of the node v_i, i.e., indices of all the nodes associated to the node v_i. Then, (real-valued) graph neural network, that is originated from Gori, Monfardini, and Scarselli (2005) and



(A) Standard NN (B) GNN

FIGURE 2.5: (A) the standard NN used in graph embedding methods considered in this thesis takes only one data vector $\boldsymbol{x}_i$ as its input. (B) GNN takes all the data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ and link weights $\{w_{ij}\}_{i,j=1}^n$ as inputs simultaneously.

Scarselli et al. (2009), is defined by

$$f_{\mathrm{GNN},i}(\boldsymbol{X}, \boldsymbol{W}) = g(h_i^{(L)}),$$

where $g : \mathbb{R}^{d_L} \to \mathbb{R}$ is a user-specified function, $h_i^{(\ell)} : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}$ $(i = 1, 2, \ldots, n; \ell = 0, 1, 2, \ldots, L; d_0 := p)$ are recursively defined as

$$h_i^{(\ell)} := \begin{cases} g^{(\ell)}(h_i^{(\ell-1)}, \{h_j^{(\ell-1)} \mid j \in \mathcal{N}_i\}) & (\ell = 1, 2, \ldots, L) \\ \boldsymbol{x}_i & (\ell = 0) \end{cases},$$

and $g^{(\ell)}(\cdot, \cdot) : \mathbb{R}^{d_{\ell-1}} \times \prod_{m=0}^{n-1} (\mathbb{R}^{d_{\ell-1}})^m \to \mathbb{R}^{d_\ell}$ is a user-specified aggregation function to be trained.

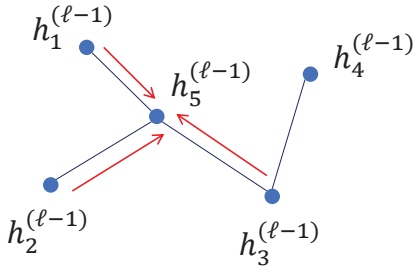


FIGURE 2.6: GNN considers the neighbour data vectors; $h_5^{(\ell)} = g^{(\ell)}(h_5^{(\ell-1)}, \{h_1^{(\ell-1)}, h_2^{(\ell-1)}, h_3^{(\ell-1)}\})$, as $\mathcal{N}_5 = \{1, 2, 3\}$.

Particularly, a special case of GNN called graph convolutional network (GCN; Kipf and Welling, 2017) trains the function

$$g^{(\ell)}(\boldsymbol{h}_i^{(\ell-1)}, \{\boldsymbol{h}_j^{(\ell-1)} \mid j \in \mathcal{N}_i\}) = \boldsymbol{\sigma}\left(\boldsymbol{A}^{(\ell)} \boldsymbol{h}_i^{(\ell-1)} + |\mathcal{N}_i|^{-1} \boldsymbol{B}^{(\ell)} \sum_{j \in \mathcal{N}_i} \boldsymbol{h}_j^{(\ell-1)}\right)$$

parametrized by $\boldsymbol{A}^{(\ell)}, \boldsymbol{B}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ for $\ell = 2, \ldots, L$. $\sigma(\cdot)$ applies the user-specified activation function $\sigma(\cdot)$ entry-wise. Once the GNN is trained, feature vectors are then obtained by

$$\boldsymbol{y}_i := h_i^{(L)} \quad (i = 1, 2, \ldots, n).$$

By virtue of the aggregation function $g^{(\ell)}$, GNN is expected to fully leverage the graph-structured associations therein.

Unlike standard NN defined in Section 2.1.1, for aggregation, GNN simultaneously takes all the data vectors $\boldsymbol{X} = (\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top, \ldots, \boldsymbol{x}_n^\top)^\top$ and the weighted adjacency matrix $\boldsymbol{W} = (w_{ij})$ as its input; GNN cannot compute the feature vector for any unseen data vector $\boldsymbol{x}$, which is not included in the data vectors $\boldsymbol{X}$ used for training.

For solving the issue, GraphSAGE (Hamilton, Ying, and Leskovec, 2017) considers the setting that associations of unseen data vector $\boldsymbol{x}$ to other data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_n$ are observed; then, feature vector $\boldsymbol{y}$ can be computed for the preliminary unseen data vector $\boldsymbol{x}$. Although GraphSAGE can deal with the unseen data vector $\boldsymbol{x}$, associations to the training data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ are yet required for computation; GNNs including GCN and GraphSAGE are different from the standard NN, that can compute feature vector for the unseen vector $\boldsymbol{x}$ without leveraging any association to any other node. Thus the GNNs are not in line with the graph embedding equipped with the standard NN, which is considered in this thesis.

Regarding the expressive power of the GNN, popular approaches in recent studies are originated from Xu et al. (2019), which considers graph isomorphism problem. This problem asks whether two graphs are topologically identical if their neighbourhoods are specified by two different weighted adjacency matrices $\boldsymbol{W}, \boldsymbol{W}'$; Xu et al. (2019) proves that the GCN and the GraphSAGE cannot distinguish some simple graphs, and even the most powerful GNN is as powerful as a test of graph isomorphism proposed by Weisfeiler and Lehman (1968). Then, Xu et al. (2019) develops a more expressive variant of the GNN called graph isomorphism network (GIN). See, e.g., Sato (2020) for a more recent line of works on the expressive power of the GNN. Note that, theories for the expressive power of the GNN rather focus on the graph isomorphism problem; it is different from those for graph embedding considered in Section 4 in this thesis, as ours is in line with the expressive power of the standard NN described in Section 2.1.2.

2.4 Multi-view Representation Learning (RL)

Whereas graph embedding explained in previous Section 2.2 considers only a single type of data vectors such as friend network, we here consider the representation learning (RL) for several different types of data vectors. It is called *multi-view RL*; we hereby summarize the multi-view RL.

Data structure

We consider D different types of data vectors. The data type called *domain* or *view* is represented by an indicator $d = 1, 2, \ldots, D$; for $d = 1, 2, \ldots, D$, data vectors $\boldsymbol{x}_1^{(d)}, \boldsymbol{x}_2^{(d)}, \ldots, \boldsymbol{x}_{n_d}^{(d)} \in \mathbb{R}^{p_d}$ of the type d are observed. The dimension p_d can depend on the view d. In addition, the following relations are considered for the link weights. Note that, the notation of multi-view data here is different from those in Chapter 3, for simply referring to existing studies. These different notations are easily translated each other.

- **One-to-one relation:** $n_1 = n_2 = \cdots = n_d$, and data vectors $\boldsymbol{x}_i^{(d)}, \boldsymbol{x}_i^{(e)}$ are associated for all $1 \leq d, e \leq D$ and $i = 1, 2, \ldots, n$.
- **Many-to-many relation:** $n_1, n_2, \ldots, n_d$ can be different, and additionally link weights within and across views $w_{ij}^{(d,e)}$ are observed. $w_{ij}^{(d,e)}$ represents the association strength between two data vectors $\boldsymbol{x}_i^{(d)}, \boldsymbol{x}_j^{(e)}$; $\boldsymbol{x}_i^{(d)}$ and $\boldsymbol{x}_j^{(e)}$ are associated if $w_{ij}^{(d,e)} > 0$.

Many-to-many relation obviously includes one-to-one relation as a special case, by specifying $w_{ij}^{(d,e)} = 1$ for $i = j$ and 0 otherwise.

Purpose of multi-view RL

Similarly to the purpose of graph embedding, that computes feature vectors $\boldsymbol{y}_i$ by transforming data vectors via (2.7), multi-view RL aims at computing new feature vectors from data vectors, i.e.,

$$\mathbb{R}^{p_d} \ni \boldsymbol{x}_i^{(d)} \mapsto \boldsymbol{y}_i^{(d)} \in \mathbb{R}^K, \quad (i = 1, 2, \ldots, n_d) \tag{2.14}$$

for each view $d = 1, 2, \ldots, D$. The feature vectors are embedded in the same dimensional common space, by leveraging the across-views graph $w_{ij}^{(1,2)}$.

2.4.1 Multi-view RL (I): One-to-One Relation

In this section, we consider multi-view RL methods leveraging the datasets consisting of one-to-one relations. Although the limited variety of relation is here considered, multi-view RL with many-to-many relation is further discussed in subsequent Section 2.4.2

Canonical correlation analysis (CCA)

Canonical correlation analysis (CCA; Hotelling, 1936; Anderson, 2003) estimates linear transformation matrices $\boldsymbol{A}^{(1)} \in \mathbb{R}^{p_1 \times K}, \boldsymbol{A}^{(2)} \in \mathbb{R}^{p_2 \times K}$ by minimizing

$$\sum_{i=1}^{n} \| \boldsymbol{A}^{(1)\top} \boldsymbol{x}_i^{(1)} - \boldsymbol{A}^{(2)\top} \boldsymbol{x}_i^{(2)} \|_2^2, \tag{2.15}$$

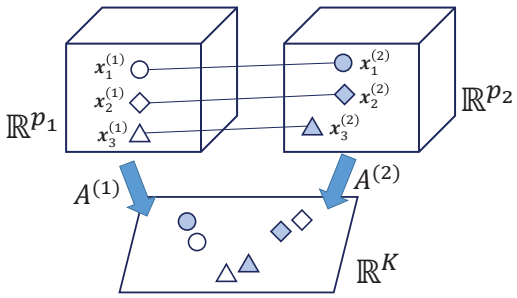


FIGURE 2.7: CCA considers one-to-one correspondence between two views; CCA finds transformation matrices $A^{(d)} \in \mathbb{R}^{p_d \times K}$ $(d = 1, 2)$ so that $y_i^{(1)} = A^{(1)\top} x_i^{(1)} \approx y_i^{(2)} = A^{(2)\top} x_i^{(2)}$.

or equivalently, maximizing $\sum_{i=1}^n \langle A^{(1)\top} x_i^{(1)}, A^{(2)\top} x_i^{(2)} \rangle$, under a quadratic constraint $\sum_{d=1}^2 \sum_{i=1}^n (A^{(d)\top} x_i^{(d)})(A^{(d)\top} x_i^{(d)})^\top = I_K$. The constraint is for preventing matrices $A^{(1)}, A^{(2)}$ from being trivial solutions. By leveraging matrices

$$\tilde{X} := \begin{pmatrix} X^{(1)} & O \\ O & X^{(2)} \end{pmatrix}, \quad \tilde{A} := \begin{pmatrix} A^{(1)} \\ A^{(2)} \end{pmatrix}, \quad T := \begin{pmatrix} I_n & -I_n \\ -I_n & I_n \end{pmatrix},$$

the objective function becomes (2.15) $= \mathrm{tr} \tilde{A}^\top \{\tilde{X}^\top T \tilde{X}\} \tilde{A}$ and the quadratic constraint is $\tilde{A}^\top \{\tilde{X}^\top \tilde{X}\} \tilde{A} = I_K$; the minimization problem

$$\text{minimize: } \mathrm{tr} \tilde{A}^\top \{\tilde{X}^\top T \tilde{X}\} \tilde{A} \quad \text{such that } \tilde{A}^\top \{\tilde{X}^\top \tilde{X}\} \tilde{A} = I_K \tag{2.16}$$

is in the form of QCQP, thus it can be solved by Lemma 2.2, similarly to spectral methods for graph embedding. Using the solution $\hat{A} = (\hat{A}^{(1)\top}, \hat{A}^{(2)\top})^\top$, we obtain feature vectors

$$y_i^{(d)} = \hat{A}^{(d)\top} x_i^{(d)} \in \mathbb{R}^K, \quad (i = 1, 2, \ldots, n)$$

for each of views $d = 1, 2$, and Lemma 2.2 yields the minimum value

$$\mathrm{tr} \hat{A}^\top \{\tilde{X}^\top T \tilde{X}\} \hat{A} = \sum_{k=1}^K \lambda_k, \tag{2.17}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_{p_1+p_2}$ are eigenvalues of $(\tilde{X}^\top \tilde{X})^{-1/2} \tilde{X}^\top T \tilde{X} (\tilde{X}^\top \tilde{X})^{-1/2}$ in ascending order.

Whereas CCA considers 2-view data vectors, i.e., $\{(x_i^{(1)}, x_i^{(2)})\}_{i=1}^n$, it is straightforwardly generalized to the setting that more than 2-view data vectors $\{(x_i^{(1)}, x_i^{(2)}, x_i^{(3)}, \ldots)\}_{i=1}^n$ are observed. This generalization is called generalized CCA (Horst, 1961) or multiset CCA (Kettenring, 1971).

Kernel CCA (KCCA)

Kernel function $\mathcal{K} : \mathcal{X}^2 \to \mathbb{R}$ defined with a set $\mathcal{X} \subset \mathbb{R}^p$ measures the similarity between two data vectors $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$. A typical example is linear kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') := \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ and Gaussian kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') := \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2/\sigma^2)$ for a user-specified parameter $\sigma > 0$. As CCA is based on the linear kernel, namely the inner product, we here consider employing non-linear kernel instead, for enhancing its expressive power.

Here, we first reformulate the CCA for defining its kernel variant. CCA estimates the projection matrix $\tilde{\boldsymbol{A}} = (\boldsymbol{A}^{(1)\top}, \boldsymbol{A}^{(2)\top})^\top \in \mathbb{R}^{(p_1+p_2)\times K}$, and the matrix $\boldsymbol{A}^{(d)}$ can be written as $\boldsymbol{A}^{(d)} = \boldsymbol{X}^{(d)\top}\boldsymbol{B}^{(d)}$ for some matrix $\boldsymbol{B}^{(d)} \in \mathbb{R}^{n\times K}$, as long as the rank of $\boldsymbol{X}^{(d)}$ is greater than or equal to p_d, for $d = 1, 2$. Namely,

$$\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{X}}^\top \boldsymbol{B}$$

for some $\boldsymbol{B} = (\boldsymbol{B}^{(1)\top}, \boldsymbol{B}^{(2)\top})^\top$. Then, the objective function of CCA (2.15) reduces to

$$\mathrm{tr}\tilde{\boldsymbol{A}}^\top\{\tilde{\boldsymbol{X}}^\top \boldsymbol{T}\tilde{\boldsymbol{X}}\}\tilde{\boldsymbol{A}} = \mathrm{tr}\boldsymbol{B}^\top\{(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top)\boldsymbol{T}(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top)\}\boldsymbol{B} = \mathrm{tr}\boldsymbol{B}^\top\{\tilde{\boldsymbol{K}}\tilde{\boldsymbol{T}}\tilde{\boldsymbol{K}}\}\boldsymbol{B} \tag{2.18}$$

for a matrix

$$\tilde{\boldsymbol{K}} = \begin{pmatrix} \boldsymbol{K}^{(1)} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{K}^{(2)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}^{(1)}\boldsymbol{X}^{(1)\top} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{X}^{(2)}\boldsymbol{X}^{(2)\top} \end{pmatrix}. \tag{2.19}$$

As well, the quadratic constraint reduces to

$$\tilde{\boldsymbol{A}}^\top\{\tilde{\boldsymbol{X}}^\top\tilde{\boldsymbol{X}}\}\tilde{\boldsymbol{A}} = \boldsymbol{B}^\top\{(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top)(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top)\}\boldsymbol{B} = \boldsymbol{B}^\top\tilde{\boldsymbol{K}}^2\boldsymbol{B}; \tag{2.20}$$

considering (2.18) and (2.20), CCA is consequently equivalent to finding aa solution $\boldsymbol{B}$ of a QCQP

$$\text{minimize: } \mathrm{tr}\boldsymbol{B}^\top\{\tilde{\boldsymbol{K}}\tilde{\boldsymbol{T}}\tilde{\boldsymbol{K}}\}\boldsymbol{B} \quad \text{such that } \boldsymbol{B}^\top\tilde{\boldsymbol{K}}^2\boldsymbol{B} = \boldsymbol{I}. \tag{2.21}$$

Lemma 2.2 solves the problem; the optimal $\hat{\boldsymbol{B}}$ yields feature vectors

$$\boldsymbol{Y} = (\boldsymbol{y}_1^{(1)\top}, \ldots, \boldsymbol{y}_n^{(1)\top}, \boldsymbol{y}_1^{(2)\top}, \ldots, \boldsymbol{y}_n^{(2)\top})^\top = \tilde{\boldsymbol{X}}\hat{\boldsymbol{A}} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top\hat{\boldsymbol{B}} = \tilde{\boldsymbol{K}}\hat{\boldsymbol{B}}.$$

The above procedure is a reformulation of CCA.

The reformulated CCA is essentially based on the matrix (2.19), where $\boldsymbol{K}^{(d)} = (k_{ij}^{(d)})$ is obtained via the linear kernel $k_{ij}^{(d)} = \langle \boldsymbol{x}_i^{(d)}, \boldsymbol{x}_j^{(d)} \rangle$ for $d = 1, 2$. We may replace the linear kernel with other non-linear kernels

$$k_{ij}^{(d)} = \mathcal{K}^{(d)}(\boldsymbol{x}_i^{(d)}, \boldsymbol{x}_j^{(d)}) \qquad \text{for } d = 1, 2;$$

following the above procedure yields kernel CCA (Lai and Fyfe, 2000; Akaho, 2001).

Considering the QCQP (2.21) of the reformulated CCA, kernel CCA applied to pairs of data vectors $\{(\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)})\}_{i=1}^n$ is in fact equivalent to linear CCA applied to $\{(\boldsymbol{z}_i^{(1)}, \boldsymbol{z}_i^{(2)})\}_{i=1}^n$, where

$$\boldsymbol{z}_i^{(d)} = (\mathcal{K}^{(d)}(\boldsymbol{x}_i^{(d)}, \boldsymbol{x}_1^{(d)}), \mathcal{K}^{(d)}(\boldsymbol{x}_i^{(d)}, \boldsymbol{x}_2^{(d)}), \ldots, \mathcal{K}^{(d)}(\boldsymbol{x}_i^{(d)}, \boldsymbol{x}_n^{(d)})) \in \mathbb{R}^n, \quad (2.22)$$

for $d = 1, 2$ and $i = 1, 2, \ldots, n$. Thus KCCA can be regarded as (i) first computing $n(\geq p_d)$-dimensional dimensional data vectors $\boldsymbol{z}_i^{(d)}$ for each view $d = 1, 2$ by leveraging the user-specified kernels $\mathcal{K}^{(1)}, \mathcal{K}^{(2)}$, and (ii) second applying CCA to the data vectors. By virtue of the high-dimensional data vectors $\{\boldsymbol{z}_i^{(d)}\}$, KCCA is in general more expressive than linear CCA.

Deep CCA (DCCA)

Similarly to kernel CCA, that corresponds to applying CCA to $\{(\boldsymbol{z}_i^{(1)}, \boldsymbol{z}_i^{(2)})\}_{i=1}^n$ defined in (2.22), deep CCA (DCCA; Andrew et al., 2013; Wang et al., 2016) applies CCA to $\{(\boldsymbol{z}_{\mathrm{NN},i}^{(1)}, \boldsymbol{z}_{\mathrm{NN},i}^{(2)})\}_{i=1}^n$ where

$$\boldsymbol{z}_{\mathrm{NN},i}^{(d)} = \boldsymbol{f}_{\mathrm{NN}}^{(d)}(\boldsymbol{x}_i^{(d)}) \quad (i = 1, 2, \ldots, n; d = 1, 2). \quad (2.23)$$

$\boldsymbol{f}_{\mathrm{NN}}^{(1)}, \boldsymbol{f}_{\mathrm{NN}}^{(2)}$ are DNNs to be trained. As shown in (2.17), the minimum value of the objective function for CCA, that is applied to (2.23), is

$$\sum_{k=1}^{K} \lambda_{\mathrm{NN},k} \quad (2.24)$$

where $\lambda_{\mathrm{NN},1}, \lambda_{\mathrm{NN},2}, \ldots, \lambda_{\mathrm{NN},p_1+p_2}$ are eigenvalues of a matrix in ascending order, that is computed via the data vectors (2.23); DCCA trains the neural networks $\boldsymbol{f}_{\mathrm{NN}}^{(1)}, \boldsymbol{f}_{\mathrm{NN}}^{(2)}$ by minimizing the above function (2.24). Typically, gradient-based approaches, such as stochastic gradient descent (Goodfellow, Bengio, and Courville, 2016), are leveraged for the optimization. Once the NNs are trained, we have obtained the feature vectors, by

$$\boldsymbol{y}_i^{(d)} = \hat{\boldsymbol{A}}^{(d)\top} \boldsymbol{z}_{\mathrm{NN},i}^{(d)} \quad (i = 1, 2, \ldots, n)$$

for each of views $d = 1, 2$. By virtue of the NNs $\boldsymbol{f}_{\mathrm{NN}}^{(1)}, \boldsymbol{f}_{\mathrm{NN}}^{(2)}$, DCCA is expected to be highly expressive, compared to the linear CCA. Similarly to multiset CCA, DCCA is also extended to general $D(\in \mathbb{N})$ views (Benton et al., 2019).

2.4.2 Multi-view RL (II): Many-to-Many Relation

Whereas CCA, KCCA and DCCA consider pairs of data vectors $\{(\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)})\}_{i=1}^n$, that have one-to-one correspondence across views, we here consider the setting that some data vectors have many-to-many associations as shown in

Figure 2.8. The associations are specified by link weights $\{w_{ij}^{(d,e)}\}$.

Cross-domain matching correlation analysis (CDMCA)

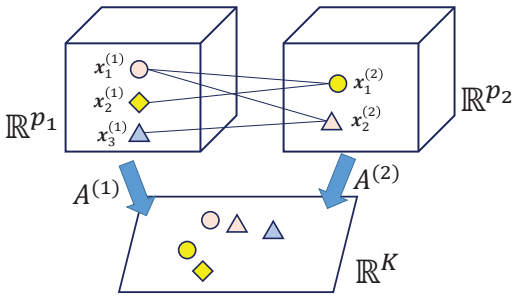


FIGURE 2.8: CDMCA considers many-to-many associations among D views; CDMCA finds transformation matrices $\boldsymbol{A}^{(d)} \in \mathbb{R}^{p_d \times K}$ $(d = 1, 2, \ldots, D)$ so that $\boldsymbol{y}_i^{(d)} = \boldsymbol{A}^{(d)\top} \boldsymbol{x}_i^{(d)} \approx \boldsymbol{y}_j^{(e)} = \boldsymbol{A}^{(e)\top} \boldsymbol{x}_j^{(e)}$ if $w_{ij}^{(d,e)} > 0$.

Considering D-view data vectors having many-to-many associations, where $w_{ij}^{(d,e)} \geq 0$ represents the association strength between the corresponding data vectors $\boldsymbol{x}_i^{(d)} \in \mathbb{R}^{p_d}$ and $\boldsymbol{x}_j^{(e)} \in \mathbb{R}^{p_e}$, cross-domain matching correlation analysis (CDMCA; Shimodaira, 2016) estimates linear transformation matrices $\boldsymbol{A}^{(d)} \in \mathbb{R}^{p_d \times K}$ $(d = 1, 2, \ldots, D)$ by minimizing

$$\sum_{d=1}^{D} \sum_{e=1}^{D} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}^{(d,e)} \| \boldsymbol{A}^{(d)\top} \boldsymbol{x}_i^{(d)} - \boldsymbol{A}^{(e)\top} \boldsymbol{x}_j^{(e)} \|_2^2, \tag{2.25}$$

or equivalently, maximizing

$$\sum_{d=1}^{D} \sum_{e=1}^{D} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}^{(d,e)} \langle \boldsymbol{A}^{(d)\top} \boldsymbol{x}_i^{(d)}, \boldsymbol{A}^{(e)\top} \boldsymbol{x}_j^{(e)} \rangle \tag{2.26}$$

under a quadratic constraint $\sum_{d=1}^{D} \sum_{e=1}^{D} \sum_{i=1}^{n_d} \sum_{j=1}^{n_e} (\boldsymbol{A}^{(d)\top} \boldsymbol{x}_i^{(d)})(\boldsymbol{A}^{(d)\top} \boldsymbol{x}_i^{(d)})^\top = \boldsymbol{I}_K$. The constraint is for preventing the matrices $\{\boldsymbol{A}^{(d)}\}_{d=1}^{D}$ from being trivial solutions. By leveraging matrices $\boldsymbol{W}^{(d,e)} = (w_{ij}^{(d,e)})$,

$$\tilde{\boldsymbol{W}} := \begin{pmatrix} \boldsymbol{W}^{(1,1)} & \boldsymbol{W}^{(1,2)} & \ldots & \boldsymbol{W}^{(1,D)} \\ \boldsymbol{W}^{(2,1)} & \boldsymbol{W}^{(2,2)} & \ldots & \boldsymbol{W}^{(2,D)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{W}^{(D,1)} & \boldsymbol{W}^{(D,2)} & \ldots & \boldsymbol{W}^{(D,D)} \end{pmatrix},$$

$$\tilde{\boldsymbol{X}} := \begin{pmatrix} \boldsymbol{X}^{(1)} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{X}^{(2)} & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{X}^{(D)} \end{pmatrix}, \tilde{\boldsymbol{A}} := \begin{pmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \\ \vdots \\ \boldsymbol{A}^{(D)} \end{pmatrix}, \boldsymbol{T} := \mathrm{diag}(\tilde{\boldsymbol{W}}\boldsymbol{1}_n) - \tilde{\boldsymbol{W}},$$

where $\mathrm{diag}(\boldsymbol{a})$ represents the diagonal matrix whose diagonal entries are specified by $\boldsymbol{a} = (a_1, a_2, \ldots, a_d) \in \mathbb{R}^d$ and $n := \sum_{d=1}^{D} n_d$, the minimization problem of CDMCA reduces to the QCQP

$$\text{minimize: } \mathrm{tr}\tilde{\boldsymbol{A}}^\top \{\tilde{\boldsymbol{X}}^\top \boldsymbol{T}\tilde{\boldsymbol{X}}\}\tilde{\boldsymbol{A}} \quad \text{such that } \tilde{\boldsymbol{A}}^\top \{\tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{X}}\}\tilde{\boldsymbol{A}} = \boldsymbol{I}_K.$$

The minimization problem is the same as the QCQP for CCA (2.16); it can be solved by Lemma 2.2. Then, the feature vectors are obtained by

$$\boldsymbol{y}_i^{(d)} = \hat{\boldsymbol{A}}^{(d)\top} \boldsymbol{x}_i^{(d)}, \quad (i = 1, 2, \ldots, n_d)$$

for each of views $d = 1, 2, \ldots, D$. A special case $D = 2$ of CDMCA is also called cross-view graph embedding (CvGE; Huang et al., 2012). CvGE also reduces to CCA, by considering the case that $n_1 = n_2$, $\boldsymbol{W}^{(1,1)} = \boldsymbol{W}^{(2,2)} = \boldsymbol{O}$ and $\boldsymbol{W}^{(1,2)} = \boldsymbol{W}^{(2,1)} = \boldsymbol{I}_{n_1}$.

Hyper-graph incidence matrix factorization (HIMFAC)

Here, we last consider the case that we have observed data vectors $\{\boldsymbol{x}_i^{(d)}\}_{i=1}^{n_d} \subset \mathbb{R}^{p_d}$ for views $d = 1, 2, \ldots, D$ and their hyper-relations $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_M$. The hyper-relation

$$\mathcal{R}_m = \{(d_{m1}, i_{m1}), (d_{m2}, i_{m2}), \ldots, (d_{mo_m}, i_{mo_m})\}$$

represents that data vectors $\boldsymbol{x}_{i_{m1}}^{(d_{m1})}, \boldsymbol{x}_{i_{m2}}^{(d_{m2})}, \ldots, \boldsymbol{x}_{i_{mo_m}}^{(d_{mo_m})}$ are associated, for $m = 1, 2, \ldots, M$.

Using the hyper-relations, we may compute link weights

$$w_{ij}^{(d,e)} = \sum_{m=1}^{M} \mathbb{1}(\text{both } (d, i), (e, j) \text{ belong to } \mathcal{R}_m),$$

indicating the number of observed hyper-relations associating data vectors $\boldsymbol{x}_i^{(d)}, \boldsymbol{x}_j^{(e)}$, for $i = 1, 2, \ldots, n_d; j = 1, 2, \ldots, n_e; d, e = 1, 2, \ldots, D$. The link weight $w_{ij}^{(d,e)}$ represents the association strength, as the strongly associated data vectors may have more relations than other pairs. Then, applying CDMCA to the data vectors $\{\boldsymbol{x}_i^{(d)}\}_{i=1}^{n_d}$ $(d = 1, 2, \ldots, D)$ and the obtained link weights $\{w_{ij}^{(d,e)}\}$ yields feature vectors $\boldsymbol{y}_i^{(d)}$ $(i = 1, 2, \ldots, n_d; d = 1, 2, \ldots, D)$. Above procedure is mathematically equivalent to hyperlink incidence matrix factorization (HIMFAC; Nori, Bollegala, and Kashima, 2012), that is originally

formulated in a slightly different way; HIMFAC computes feature vectors of data vectors, by leveraging hyper-relations among multi-view data vectors.

3 Multi-view Graph Embedding

This chapter discusses the multi-view extension of NN-based graph embedding, that is named probabilistic multi-view graph embedding (PMvGE). Our contributions, descriptions, and materials including figures in this chapter are mostly attributed to a published conference paper of ours (Okuno, Hada, and Shimodaira, 2018).

TABLE 3.1: Existing multi-view / graph-embedding methods are compared to the proposed PMvGE (grey); only PMvGE possesses all the following properties simultaneously.

(Nv): Number of views, (MM): Many-to-many, (NL): Non-linear,
(Ind): Inductive, (Lik): Likelihood-based.

	(Nv)	(MM)	(NL)	(Ind)	(Lik)
CCA (Hotelling, 1936)	2			✓	
DCCA (Andrew et al., 2013)	2		✓	✓	
MCCA (Kettenring, 1971)	*D*			✓	
SGE (Belkin and Niyogi, 2001)	0	✓			
LINE (Tang et al., 2015)	0	✓			✓
LPP (He and Niyogi, 2004)	1	✓		✓	
CvGE (Huang et al., 2012)	2	✓		✓	
HIMFAC (Nori, Bollegala, and Kashima, 2012)	*D*	✓		✓	
CDMCA (Shimodaira, 2016)	*D*	✓		✓	
DeepWalk (Perozzi, Al-Rfou, and Skiena, 2014)	0	✓			✓
SBM (Holland, Laskey, and Leinhardt, 1983)	1	✓		✓	✓
GCN (Kipf and Welling, 2017)	1	✓	✓		✓
GraphSAGE (Hamilton, Ying, and Leskovec, 2017)	1	✓	✓	✓	✓
IDW (Dai et al., 2018)	1	✓	✓	✓	✓
PMvGE (Okuno, Hada, and Shimodaira, 2018)	*D*	✓	✓	✓	✓

3.1 Introduction

With the rapid development of Internet communication tools in the 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 chapter, 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.

One of the best known approaches to multi-view feature learning is Canonical Correlation Analysis (CCA; Hotelling, 1936) for two-views, and Multiset CCA (MCCA; Kettenring, 1971) 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 $\boldsymbol{y}_i^{(1)} := \boldsymbol{A}^{(1)\top} \boldsymbol{x}_i^{(1)}, \boldsymbol{y}_i^{(2)} := \boldsymbol{A}^{(2)\top} \boldsymbol{x}_i^{(2)} \in \mathbb{R}^K$ 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 (DCCA; Andrew et al., 2013; Wang et al., 2016) 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, where $w_{ij}^{(d,e)} \geq 0$ represents the association strength between $\boldsymbol{x}_i^{(d)}$ and $\boldsymbol{x}_j^{(e)}$, Cross-view Graph Embedding (CvGE; Huang et al., 2012) and its extension to more than three views called Cross-Domain Matching Correlation Analysis (CDMCA; Shimodaira, 2016) 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 weighted sum of inner products

$$\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} w_{ij}^{(1,2)} \langle \boldsymbol{A}^{(1)\top}\boldsymbol{x}_i^{(1)}, \boldsymbol{A}^{(2)\top}\boldsymbol{x}_j^{(2)}\rangle$$

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 $w_{ij}^{(1,2)} = \delta_{ij}$, where δ_{ij} is Kronecker's delta and $w_{ij}^{(1,1)} = w_{ij}^{(2,2)} = 0$. 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 (LPP; He and Niyogi, 2004; Yan et al., 2007). LPP also reduces to Spectral Graph Embedding (SGE; Chung, 1997; Belkin and Niyogi, 2001), 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, Fukui, and Shimodaira, 2016; Fukui, Okuno, and Shimodaira, 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 chapter, 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 the 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, Ying, and Leskovec, 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 $w_{ij} \geq 0$.

For D-view feature learning ($D \geq 1$), PMvGE transforms multi-view data vectors to the corresponding feature vectors by applying neural networks. Inspired by the existing graph embedding methods, the neural networks in PMvGE are trained so that the inner product similarity between the obtained feature vectors predicts the corresponding link weight. We introduce a parametric model of the conditional probability of link weights given data vectors, and thus PMvGE is a non-linear and probabilistic extension of CDMCA. This leads to an efficient computation of the Maximum Likelihood Estimator (MLE) with minibatch SGD.

Our contribution in this chapter is summarized as follows:

(1) In Section 3.3, we propose PMvGE for multi-view feature learning with many-to-many associations, which is non-linear, efficient to compute,

and inductive; a method is called *inductive* if it computes feature vectors for newly obtained vectors which are not included in the training set. A comparison with existing methods is shown in Table 3.1.

(2) In Section 3.4, we define maximum likelihood estimator for PMvGE, and propose an efficient minibatch sampling procedure for conducting stochastic optimization, that leverages graph-structured associations characterized by link weights.

(3) In Section 3.5, we show that PMvGE generalizes various existing multi-view methods, at least approximately, by considering the Maximum Likelihood Estimator (MLE) with a novel probabilistic model. Especially, we prove that PMvGE at least approximately generalizes linear CDMCA, by considering the quadratic approximation.

(4) In Section 3.6, the proposed PMvGE and other baselines are performed on real-world datasets.

3.2 Preliminaries

In this section, we first define symbols for D-view data vectors and their associations. In Introduction of this chapter and preliminaries in Chapter 2 in this thesis, data vectors in view d were denoted by $\{\boldsymbol{x}_i^{(d)}\}_{i=1}^{n_d}$ by referring to existing studies. However, in this chapter, we employ a simpler notation; D-view data vectors are denoted by

$$\boldsymbol{x}_1(\in \mathbb{R}^{p_{d_1}}),\ \boldsymbol{x}_2(\in \mathbb{R}^{p_{d_2}}),\ \ldots,\boldsymbol{x}_n(\in \mathbb{R}^{p_{d_n}})$$

altogether, and $d_i \in \{1,2,\ldots,D\}$ denotes the view to which the data vector $\boldsymbol{x}_i$ belongs. The dimension p_{d_i} of data vector $\boldsymbol{x}_i$ depends on the view d_i.

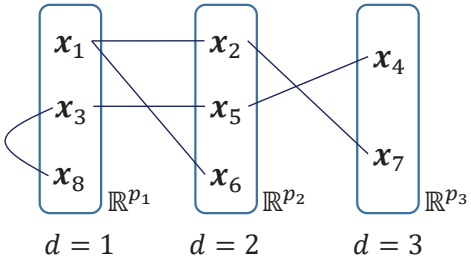


FIGURE 3.1: $\{\boldsymbol{x}_i\}_{i=1}^n$ are multi-view data vectors, and the association strength between $\boldsymbol{x}_i, \boldsymbol{x}_j$ is denoted by link weight $w_{ij} \geq 0$. d_i denotes the view to which data vector $\boldsymbol{x}_i$ belongs.

See Figure 3.1 for example. Therein, data vectors $\boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_8$ belong to view-1, $\boldsymbol{x}_2, \boldsymbol{x}_5, \boldsymbol{x}_6$ belong to view-2, and $\boldsymbol{x}_4, \boldsymbol{x}_7$ belong to view-3. By virtue of the simple notation, we may simply denote the link weights within/across views in a unified manner; the link weight

$$w_{ij} = w_{ji} \geq 0$$

is defined to represent the association strength between data vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$ for all i, j, and $w_{ii} = 0$.

Therefore, $\{w_{ij}\}_{i,j=1}^n, \{(\boldsymbol{x}_i, d_i)\}_{i=1}^n$ are observations hereinafter. By taking w_{ij} as a random variable, we consider a parametric model of the conditional probability of w_{ij} given the data vectors. In Section 3.3.1, we consider the probability model of w_{ij} with the conditional expected value

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

where $\{\boldsymbol{x}_i, d_i\}_{i=1}^n$ is given, for all $1 \leq i < j \leq n$. In Section 3.3.2, we then define PMvGE by specifying the functional form of μ_{ij} via feature vectors.

3.3 PMvGE

3.3.1 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

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

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

$$w_{ij} \overset{\text{indep.}}{\sim} \mathrm{Po}(\mu_{ij}) \tag{3.1}$$

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

Our probabilistic model (3.1) coincides with Stochastic Block Model (Holland, Laskey, and Leinhardt, 1983, SBM) by assuming that node v_i belongs

to a cluster $c_i \in \{1, 2, \ldots, C\}$. The model is specified as

$$\mu_{ij} = \beta^{(c_i, c_j)}, \tag{3.2}$$

where $\beta^{(c_i,c_j)}$ is the parameter regulating the number of links whose endpoints 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.

3.3.2 PMvGE

Inspired by existing methods for feature learning such as LINE (Tang et al., 2015), that is described in Section 2.2.2 in Chapter 2, we propose a novel model for the parameter μ_{ij} in eq. (3.1) by using the inner-product similarity as

$$\mu_{\boldsymbol{\alpha},\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j; d_i, d_j) := \alpha^{(d_i,d_j)} \exp\left(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle\right), \tag{3.3}$$
$$\boldsymbol{y}_i := \boldsymbol{f}_{\mathrm{NN}}^{(d_i)}(\boldsymbol{x}_i).$$

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 $\boldsymbol{y} = (y_1, \ldots, y_n), \boldsymbol{y}' = (y'_1, \ldots, y'_n) \in \mathbb{R}^K$, $\langle \boldsymbol{y}, \boldsymbol{y}' \rangle = \sum_{k=1}^K y_k y'_k$ is simply the inner product in Euclidean space. $\boldsymbol{f}_{\mathrm{NN}} : \mathbb{R}^{p_d} \to \mathbb{R}^K \quad (d = 1, 2, \ldots, D)$ represent vector-valued neural networks (see, Section 2.1 in Chapter 2 for neural networks), whose parameters are altogether specified by $\boldsymbol{\theta}$.

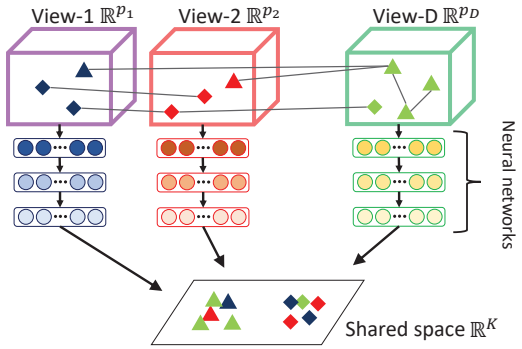


FIGURE 3.2: Data vectors $\{\boldsymbol{x}_i\}$ in each view are transformed to feature vectors $\{\boldsymbol{y}_i\}$ by neural networks $\{\boldsymbol{f}_{\mathrm{NN}}^{(d)}\}$.

As shown in Lemma 2.1 in Chapter 2, universal approximation theorem proves that sufficiently large neural networks can approximate arbitrary continuous function, meaning that neural networks are highly expressive. These neural networks $\{\boldsymbol{f}_{\mathrm{NN}}^{(d)}\}$ can be trained by maximizing the likelihood for the probabilistic model (3.1) as later shown in Section 3.4.1. Then, feature vectors are obtained by simply applying these trained NNs to the data vectors $\boldsymbol{x}_i$.

Link weights across some view pairs may be missing

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

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

Although link weights between the view pair $(d, e) \notin \mathcal{D}$ are not leveraged in the proposed PMvGE, we formally set $w_{ij} = 0$ for the missing $(d_i, d_j) \notin \mathcal{D}$ when needed.

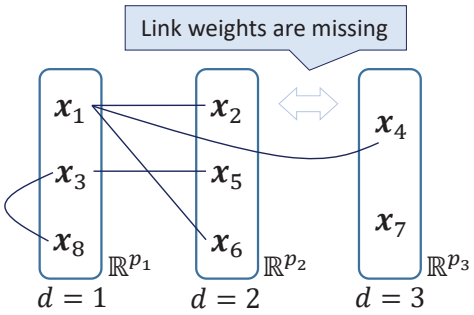


FIGURE 3.3: $\mathcal{D} = \{(1,1), (1,2), (1,3)\}$

For example in Figure 3.3, $\mathcal{D} = \{(1,1), (1,2), (1,3)\}$ indicates that link weights (i) within view 1, (ii) across views 1 and 2, (iii) across views 1 and 3, are observed while link weights between other view pairs 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.

3.4 Parameter Estimation

In this section, we first define the maximum likelihood estimator for PMvGE in Section 3.4.1. We then present an efficient way of optimizing the likelihood function in Section 3.4.2; we iteratively optimize the two parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$. In the proposed algorithm, $\boldsymbol{\theta}$ is updated by minibatch SGD, and $\boldsymbol{\alpha}$ is updated by directly solving an estimating equation.

3.4.1 Maximum Likelihood Estimator

Since PMvGE is specified by (3.1) and (3.3), the log-likelihood function is given by

$$\ell_n(\boldsymbol{\alpha}, \boldsymbol{\theta}) := \sum_{(i,j) \in \mathcal{I}_n} [w_{ij} \log \mu_{\boldsymbol{\alpha},\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j; d_i, d_j) - \mu_{\boldsymbol{\alpha},\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j; d_i, d_j)], \tag{3.4}$$

whose sum is over the set of index pairs $\mathcal{I}_n := \{(i,j) \mid 1 \leq i < j \leq n, (d_i, d_j) \in \mathcal{D}\}$. The Maximum Likelihood Estimator (MLE) of the parameter $(\boldsymbol{\alpha}, \boldsymbol{\theta})$ is defined as the maximizer of (3.4). We estimate $(\boldsymbol{\alpha}, \boldsymbol{\theta})$ by maximizing $\ell_n(\boldsymbol{\alpha}, \boldsymbol{\theta})$

with constraints $\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top$. Although not leveraged, we formally set $\alpha^{(d,e)} = 0$ for $(d,e) \notin \mathcal{D}$.

3.4.2 Stochastic Optimization

We propose an algorithm for optimizing the NNs $f_{\mathrm{NN}}^{(1)}, f_{\mathrm{NN}}^{(2)}, \ldots, f_{\mathrm{NN}}^{(D)}$. The algorithm iteratively updates the parameters $\boldsymbol{\theta}, \boldsymbol{\alpha}$ therein.

Update of $\boldsymbol{\theta}$

We update $\boldsymbol{\theta}$ using the gradient of $\ell_n(\bar{\boldsymbol{\alpha}}, \boldsymbol{\theta})$ 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, Bengio, and Courville, 2016) for the former half of $\ell_n(\bar{\boldsymbol{\alpha}}, \boldsymbol{\theta})$ is expected to be reduced by considering the sum over the set $\mathcal{P}_n := \{(i,j) \in \mathcal{I}_n \mid w_{ij} > 0\}$. On the other hand, there should be $|\mathcal{I}_n| = O(n^2)$ positive terms in the latter half of $\ell_n(\bar{\boldsymbol{\alpha}}, \boldsymbol{\theta})$, so we consider the sum over node pairs uniformly-resampled from $\mathcal{I}_n$.

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

$$|\tilde{\mathcal{I}}_n| + |\tilde{\mathcal{P}}_n| = m,\ |\tilde{\mathcal{I}}_n| / |\tilde{\mathcal{P}}_n| = r.$$

User-specified constants $m \in \mathbb{N}$ and $r > 0$ are usually called as "minibatch size" and "negative sampling rate". We sequentially update minibatches $\tilde{\mathcal{I}}_n, \tilde{\mathcal{P}}_n$ for computing the gradient of

$$\sum_{(i,j) \in \tilde{\mathcal{P}}_n} w_{ij} \log \mu_{\bar{\boldsymbol{\alpha}}, \boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j; d_i, d_j) - \eta \sum_{(i,j) \in \tilde{\mathcal{I}}_n} \mu_{\bar{\boldsymbol{\alpha}}, \boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j; d_i, d_j) \tag{3.5}$$

with respect to $\boldsymbol{\theta}$. By utilizing the gradient, the parameter $\boldsymbol{\theta}$ can be sequentially updated by SGD, where $\eta > 0$ is a tuning parameter. Eq. (3.5) approximates $\ell_n(\bar{\boldsymbol{\alpha}}, \boldsymbol{\theta})$ if (i,j) are uniformly-resampled and $\eta = |\mathcal{I}_n| / (r\,|\mathcal{P}_n|)$, however, smaller η such as $\eta = 1$ may make this algorithm stable in some cases.

This optimization procedure is strongly inspired by negative sampling used in skip-gram (a.k.a. word2vec; Mikolov et al., 2013), and it requires $O(m)$ operations for each minibatch iteration. Thus $\boldsymbol{\theta}$ is efficiently updated even if the number of data vectors n is very large. The proposed stochastic optimization and negative sampling are further generalized in Chapter 5, and they are theoretically justified in a unified manner.

Update of $\boldsymbol{\alpha}$

Let $\bar{\boldsymbol{\theta}}$ represent current parameter value of $\boldsymbol{\theta}$, and let $\bar{f}_{\mathrm{NN}}^{(d)}$ represent a neural network $f_{\mathrm{NN}}^{(d)}$ whose parameter is fixed to $\bar{\boldsymbol{\theta}}$, for $d = 1, 2, \ldots, D$. By solving the estimating equation $\frac{\partial \ell_n(\boldsymbol{\alpha}, \bar{\boldsymbol{\theta}})}{\partial \alpha^{(d,e)}} = 0$ 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 $\ell_n(\boldsymbol{\alpha}, \bar{\boldsymbol{\theta}})$. However, the local maximizer requires roughly $O(n^2)$ operations for

computation. To reduce the high computational cost, we efficiently update $\boldsymbol{\alpha}$ by (3.6), which is a minibatch-based approximation of the local maximizer.

$$\hat{\alpha}^{(d,e)} := \begin{cases} \dfrac{\sum_{(i,j)\in\mathcal{I}_n^{'(d,e)}} w_{ij}}{\sum_{(i,j)\in\mathcal{I}_n^{'(d,e)}} \exp(\langle \bar{\boldsymbol{f}}_{\mathrm{NN}}^{(d)}(\boldsymbol{x}_i), \bar{\boldsymbol{f}}_{\mathrm{NN}}^{(e)}(\boldsymbol{x}_j)\rangle)} & (\text{for } (d,e) \in \mathcal{D}) \\ 0 & (\text{otherwise}) \end{cases}, \tag{3.6}$$

where $\mathcal{I}_n^{'(d,e)} := \{(i,j) \in \mathcal{I}_n' \mid (d_i, d_j) = (d,e)\}$ and $\mathcal{I}_n'$ is the minibatch defined for updating $\boldsymbol{\theta}$. Note that (d,e) is unordered view pair.

3.5 Relation to Other Multi-view Methods

Assuming that the activation function in the neural networks $\boldsymbol{f}_{\mathrm{NN}}^{(d)}$ is identity function, i.e., $\sigma(z) = z$, the neural networks reduce to linear models

$$\boldsymbol{f}_{\mathrm{NN}}^{(d)}(\boldsymbol{x}_i) = \boldsymbol{\theta}^{(d)\top}\boldsymbol{x}_i \quad (d = 1, 2, \ldots, D). \tag{3.7}$$

Consider the linear model (3.7) with $\boldsymbol{\theta}^{(1)\top} = (\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^C) \in \mathbb{R}^{K\times C}$, where $\boldsymbol{\theta}^c \in \mathbb{R}^K$ is a feature vector for class $c = 1, \ldots, C$. Then,

$$\mu_{\boldsymbol{\alpha},\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j; d_i, d_j) = \alpha^{(1,1)} \exp(\langle \boldsymbol{\theta}^{c_i}, \boldsymbol{\theta}^{c_j}\rangle)$$

will specify $\beta^{(c_i,c_j)}$ for sufficiently large K, and $\{\boldsymbol{\theta}^{c_i}\}$ 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 $f_{\boldsymbol{\theta}}^{(d_i)}(\boldsymbol{x}) \equiv 0$. Then $\mu_{\boldsymbol{\alpha},\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j; d_i, d_j) = \alpha^{(c_i,c_j)}$ is equivalent to SBM.

More generally, CDMCA (Shimodaira, 2016) is approximated by D-view PMvGE with the linear transformations (3.7). The first half of the objective function (3.4) becomes

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\langle \boldsymbol{\theta}^{(d_i)\top}\boldsymbol{x}_i, \boldsymbol{\theta}^{(d_j)\top}\boldsymbol{x}_j\rangle \tag{3.8}$$

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

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\boldsymbol{\theta}^{(d_i)\top}\boldsymbol{x}_i\boldsymbol{x}_i^\top\boldsymbol{\theta}^{(d_i)} = \boldsymbol{I}. \tag{3.9}$$

The above observation intuitively explains why PMvGE approximates CDMCA; this is more formally discussed in Appendix A.1. The quadratic constraint is required for preventing the maximizer of (3.8) from being diverged in CDMCA. However, our likelihood approach does not require it, because the last half of (3.4) serves as a regularization term.

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

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

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

$$\underset{j \in \{1,2,\ldots,C\}}{\arg\max}\ \exp\left(\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{e}_j \rangle\right), \tag{3.10}$$

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

The classifier (3.10) can be interpreted as PMvGE with $D = 2, \mathcal{D} = \{(1,2)\}$ as follows: $\boldsymbol{f}^{(1)}_{\mathrm{NN}}(\boldsymbol{x}) := \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x})$ and inputs are $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ for view-1, $\boldsymbol{f}^{(2)}_{\mathrm{NN}}(\boldsymbol{x}') := \boldsymbol{x}'$ and inputs are $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_C$ for view-2, and $w_{ij} = 1$ between $\boldsymbol{x}_i$ and $\boldsymbol{e}_{c_i}$, and $w_{ij} = 0$ otherwise.

3.6 Numerical Experiments

3.6.1 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. Hyper-parameters are tuned by utilizing the validation set.

- **Baselines:** We compare PMvGE with several feature learning methods: Stochastic Block Model (Holland, Laskey, and Leinhardt, 1983, SBM), ISOMAP (Tenenbaum, De Silva, and Langford, 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, Al-Rfou, and Skiena, 2014), and GraphSAGE (Hamilton, Ying, and Leskovec, 2017).

- **NN for PMvGE:** 1-hidden-layer fully-connected network, which consists of 3000 tanh hidden units and (K =)1000 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 $\{50, 100, 150, 200\}$.

We conduct the following two experiments.

- **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 3.2. In experiment (A), we apply methods to both training set and test set, and evaluate them by the test set. In (B), we apply methods to only the training set, and evaluate them by the 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.

3.6.2 Experiments on AwA dataset (2-view)

- **Dataset:** We use Animal with Attributes (AwA) dataset (Lampert, Nickisch, and Harmeling, 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, Socher, and Manning, 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 1000 test images. A validation set of 300 images is sampled from the training images.

TABLE 3.2: Task 1 and Task 2 for the experiment on Citation dataset ($D = 1$). 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	$\mathbf{35.9 \pm 0.88}$	$\mathbf{30.5 \pm 3.90}$

- **Baselines:** We compare PMvGE with CCA, DCCA (Andrew et al., 2013), SGE, DeepWalk, and GraphSAGE.
- **NN for PMvGE:** Each view uses a 1-hidden-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 ($p = 0.5$). Mini-batch 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 $\{10, 50, 100, 150\}$.

Then, we conduct the following experiment.

- **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 3.3. In experiment (A), we apply methods to both training set and test set, and evaluate them by the 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 the test set. 20% of the 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 3.3: Task 3 for the experiment on AwA dataset ($D = 2$). The larger values are better.

		(A)	(B)
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 $\langle \boldsymbol{f}^{(d)}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}^{(d)}_{\mathrm{NN}}(\boldsymbol{x}') \rangle$ 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 preserve the similarities of the data vectors fairly well.

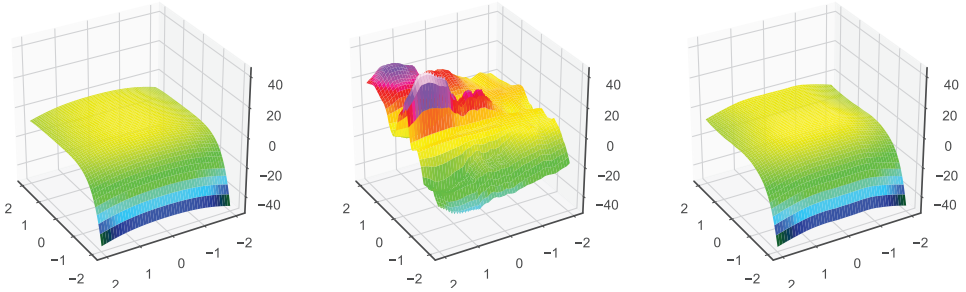
3.7 Conclusion

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 a practical estimation algorithm. Experiments on real-world datasets showed that PMvGE outperforms existing methods.

4 Expressive Power of NN-Based Graph Embedding

This chapter discusses the expressive power of the NN-based graph embedding. Firstly, we discuss the expressive power of inner-product similarity (IPS) that is often used in graph embedding; we prove that IPS is limited to approximate positive-definite (PD) similarities even if highly expressive NN is employed for computing feature vectors. We then propose shifted-IPS (SIPS), and we theoretically prove that SIPS is more expressive than IPS. The expressive power is also illustrated in the following Figure 4.1. At last, we also propose more expressive inner-product difference similarity (IPDS), and prove that the IPDS can approximate arbitrary similarities; graph embeddings using the SIPS and IPDS are expected to demonstrate higher performance.

Our contributions, descriptions, and materials including figures in this chapter are mostly attributed to a published conference paper of ours (Okuno, Kim, and Shimodaira, 2019), and experiments are partially attributed to another workshop paper of ours (Okuno and Shimodaira, 2018).



(A) True (B) Existing (IPS) (C) Proposed (SIPS)

FIGURE 4.1: (A) For $f_*(x) = (x_1, \cos x_2, \exp(-x_3), \sin(x_4 - x_5)) \in \mathbb{R}^4$ with respect to $x \in \mathbb{R}^5$, the negative squared distance (NSD) similarity $-\|f_*(se_1) - f_*(te_2)\|_2^2$ is plotted on (s,t)-plane along with two orthogonal directions $e_1, e_2 \in \mathbb{R}^5$. This NSD is approximated by the two similarity models: (B) Existing IPS $\langle f_{\mathrm{NN}}(se_1), f_{\mathrm{NN}}(te_2)\rangle$, and (C) Proposed SIPS $\langle f_{\mathrm{NN}}(se_1), f_{\mathrm{NN}}(te_2)\rangle + u_{\mathrm{NN}}(se_1) + u_{\mathrm{NN}}(te_2)$, where $f_{\mathrm{NN}} : \mathbb{R}^5 \to \mathbb{R}^{10}$ and $u_{\mathrm{NN}} : \mathbb{R}^5 \to \mathbb{R}$ are two-layer neural networks with 1,000 hidden units and ReLU activations. The proposed SIPS approximates the NSD better than the existing IPS.

4.1 Introduction

As discussed in previous Chapters 2 and 3, various graph embedding methods measure the similarities between feature vectors, for predicting the corresponding link weights. Especially, inner-product is often employed for measuring the similarity between two feature vectors therein; the inner-product applied to the feature vectors, that are computed via a neural network to be trained, is called *inner-product similarity (IPS)*.

IPS is considered to be highly expressive for representing the association between data vectors due to the universal approximation theorem (UAT; Funahashi, 1989; Cybenko, 1989; Yarotsky, 2017; Telgarsky, 2017; Yarotsky, 2018) for NN. As described in section 2.1.2 in chapter 2, the UAT proves that NNs having many hidden units approximate arbitrary continuous functions within any given accuracy. However, since IPS considers the inner product of two vector-valued NNs, the UAT is not directly applicable to the whole network with the constraints at the final layer. Thus the approximation capability of IPS is yet to be clarified.

Meanwhile, similarities based on specific kernels other than the inner product have received considerable attention in recent years. One example is Poincaré embedding (Nickel and Kiela, 2017) which is an NN-based GE using Poincaré distance for embedding vectors in hyperbolic space instead of Euclidean space. Hyperbolic space is especially compatible with computing feature vectors of tree-structured relational data (Sarkar, 2011). These methods efficiently compute reasonable low-dimensional feature vectors by virtue of specific kernels. However, their theoretical differences from IPS, in terms of the expressive power in line with the standard NN summarized in Section 2.1.2 in Chapter 2, is not well understood.

For that reason, in this chapter, we first incorporate UAT into Mercer's theorem, and prove that IPS approximates any similarity based on Positive Definite (PD) kernels arbitrarily well. For example, IPS can learn cosine similarity, because it is a PD kernel. Although this result shows the validity, we also prove the fundamental limitation of IPS, meaning that the IPS cannot approximate non-PD similarities.

Furthermore, in order to provide theoretical insights, we will point out that some specific kernels are not PD by referring to existing studies. To deal with such non-PD kernels, we consider Conditionally PD (CPD) kernels (Berg, Christensen, and Ressel, 1984; Schölkopf, 2001) which include PD kernels as special cases. We then propose a novel model named *Shifted IPS (SIPS)* that approximates similarities based on CPD kernels within any given accuracy. Interestingly, negative Poincaré distance is already proved to be CPD (Faraut and Harzallah, 1974) and it is not PD. So, similarities based on this kernel can be approximated by SIPS but not by IPS.

At last, we further consider more generalization beyond CPD. We then propose *inner-product difference similarity (IPDS)*, which can approximate a wider class than CPD. Especially, we prove that IPDS can approximate arbitrary similarities, i.e., arbitrary symmetric and continuous functions.

Our contribution in this chapter is summarized as follows:

(1) In Section 4.3, we first show that IPS can approximate arbitrary PD similarities, whereas it cannot approximate non-PD similarities. In subsequent sections, we propose SIPS and C-SIPS to go beyond the limitation, and prove that they can approximate any CPD similarities arbitrarily well. Approximation error rates are also evaluated.

(2) In Section 4.6, we conduct numerical experiments on approximation capability of IPS, SIPS, and C-SIPS. Subsequently, experiments on two real-world datasets are also performed to show that graph embedding using SIPS outperforms existing methods.

(3) In Section 4.7, we prove that even the proposed SIPS cannot approximate non-CPD similarities. Inner-product difference similarity (IPDS) is also provided to approximate such non-CPD similarities.

4.2 Preliminaries

We consider 1-view data vectors and graph-structured associations specified by link weights. This is a special case ($D = 1$) of the problem setting considered in Chapter 3, which is again briefly summarized as follows.

We work on an undirected graph consisting of n nodes $\{v_i\}_{i=1}^n$ and link weights $\{w_{ij}\}_{i,j=1}^n \subset \mathbb{R}_{\geq 0}$ satisfying $w_{ij} = w_{ji}$ and $w_{ii} = 0$, where w_{ij} represents the strength of association between v_i and v_j. The data vector representing the attributes (or side-information) at v_i is denoted as $\boldsymbol{x}_i \in \mathbb{R}^p$. If we have no attributes, we use 1-hot vectors in $\mathbb{R}^n$ instead. We assume that the observed dataset consists of $\{w_{ij}\}_{i,j=1}^n$ and $\{\boldsymbol{x}_i\}_{i=1}^n$.

By specifying $D = 1$ and $\alpha^{(1,1)} = 1$, meaning that only a single type of data vectors is considered, the Poisson model (3.1) of PMvGE equipped with the similarity function (3.3), where they are defined in the previous Chapter 3, reduces to

$$w_{ij} \overset{\text{indep.}}{\sim} \mathrm{Po}(\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)), \qquad \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) := \exp(h_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)). \tag{4.1}$$

The similarity function $h_{\boldsymbol{\theta}}$ is specified by the inner-product similarity

$$h_{\mathrm{IPS}}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}) = \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_j) \rangle \tag{4.2}$$

using a neural network $\boldsymbol{f}_{\mathrm{NN}} : \mathbb{R}^p \to \mathbb{R}^K$ whose parameter is $\boldsymbol{\theta}$. IPS commonly appears in a broad range of methods, such as DeepWalk (Perozzi, Al-Rfou, and Skiena, 2014), LINE (Tang et al., 2015), node2vec (Grover and Leskovec, 2016), Variational Graph AutoEncoder (Kipf and Welling, 2016), and GraphSAGE (Hamilton, Ying, and Leskovec, 2017), and IPS is straightforwardly generalized to a model

$$g(\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_j)),$$

that is especially called siamese neural network (Bromley et al., 1994) in neural network literature. Although IPS is often considered, a study called

Poincaré embedding considers employing negative Poincaré distance for the kernel g. In order to clarify which of the kernels is better to be employed, in the remaining of this chapter, we discuss the expressive power of the similarities in the form of siamese NN.

Throughout this chapter, similarity between the two feature vectors $\boldsymbol{y} = \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{y}' = \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')$ is measured by the kernel g, that will be formally defined as a symmetric and continuous function in Definition 4.1. In addition to the symmetry and continuity, some additional conditions, such as the triangle inequality, may be imposed on the kernel g. Related to such additional conditions, the negative-squared distance (NSD) between two feature vectors

$$h_{\mathrm{NSD}}(\boldsymbol{x}, \boldsymbol{x}') := -\|\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}) - \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\|_2^2, \tag{4.3}$$

will be considered later in Section 4.4.4, and the corresponding NSD kernel $g(\boldsymbol{y}, \boldsymbol{y}') := -\|\boldsymbol{y} - \boldsymbol{y}'\|_2^2$ satisfies the axioms of distance, such as the (sign-reversed) triangle inequality $(-g(\boldsymbol{y}, \boldsymbol{y}'))^{1/2} \leq (-g(\boldsymbol{y}, \boldsymbol{y}''))^{1/2} + (-g(\boldsymbol{y}'', \boldsymbol{y}'))^{1/2}$ and the property that $g(\boldsymbol{y}, \boldsymbol{y}') = 0$ if and only if $\boldsymbol{y} = \boldsymbol{y}'$.

For comparison purposes, the similarity (4.3) is evaluated in Section 4.6.2 numerically. In the experiments conducted on two real-world datasets, our proposed shifted IPS model, which is free from the additional conditions, outperforms both existing IPS and the similarity (4.3). Therefore, we only assume the symmetry and the continuity of the kernel g, for developing more expressive similarities.

4.3 Inner Product Similarity (IPS)

In this section, we consider the expressive power of IPS given in eq. (4.2).

4.3.1 PD Similarities

In order to prove the approximation capability of IPS, we first incorporate the UAT for NN (Funahashi, 1989; Cybenko, 1989; Yarotsky, 2017; Telgarsky, 2017) into Mercer's theorem (Minh, Niyogi, and Yao, 2006). Subsequently, we describe the assertion that shows uniform convergence of IPS to any PD similarity. To show the result in Theorem 4.2, we here first define a kernel and its positive-definiteness.

Definition 4.1: Kernel

Let $K_* \in \mathbb{N}$ and $\mathcal{Y} \subset \mathbb{R}^{K_*}$ be a non-empty set. Then, a symmetric continuous function $g : \mathcal{Y}^2 \to \mathbb{R}$ is called a *kernel* on $\mathcal{Y}^2$.

Definition 4.2: Positive definiteness

A kernel g on $\mathcal{Y}^2$ is said to be *Positive Definite (PD)* if satisfying $\sum_{i=1}^n \sum_{j=1}^n c_i c_j g(\boldsymbol{y}_i, \boldsymbol{y}_j) \geq 0$ for arbitrary $c_1, c_2, \ldots, c_n \in \mathbb{R}$ and

$\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n \in \mathcal{Y}$.

For instance, cosine similarity $g(\boldsymbol{y}, \boldsymbol{y}') := \langle \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_2}, \frac{\boldsymbol{y}'}{\|\boldsymbol{y}'\|_2} \rangle$ is a PD kernel on $(\mathbb{R}^p \setminus \{\boldsymbol{0}\})^2$. Its PD-ness immediately follows from $\sum_{i=1}^n \sum_{j=c}^n c_i c_j g(\boldsymbol{y}_i, \boldsymbol{y}_j) = \| \sum_{i=1}^n c_i \frac{\boldsymbol{y}_i}{\|\boldsymbol{y}_i\|_2} \|_2^2 \geq 0$ for arbitrary $\{c_i\}_{i=1}^n \subset \mathbb{R}$ and $\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathcal{Y}$. Also polynomial kernel, Gaussian kernel, and Laplacian kernel are PD (Berg, Christensen, and Ressel, 1984).

Definition 4.3: Similarity

Let g be a kernel on $\mathcal{Y}^2$ and let $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ be a continuous function. Then, a function $h(\boldsymbol{x}, \boldsymbol{x}') := g(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}'))$ is called a *similarity* on $\mathcal{X}^2$.

Rigorously speaking, the similarity h can be regarded as a kernel on $\mathcal{X}^2$. However, our study focuses on graph embedding, which aims at computing feature vectors; we especially call h as a "similarity" to distinguish it from the kernel, in the sense that the similarity h is used for measuring how similar two data vectors $\boldsymbol{x}, \boldsymbol{x}'$ are, while the kernel g is used to compare feature vectors $\boldsymbol{y}, \boldsymbol{y}'$.

The similarity h equipped with a PD kernel g is also PD on $\mathcal{X}^2$, since

$$\sum_{i=1}^n \sum_{j=1}^n c_i c_j h(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^n \sum_{j=1}^n c_i c_j g(\boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j)) \geq 0.$$

Assuming that a kernel is PD, the following Mercer's theorem (Minh, Niyogi, and Yao, 2006) proves that the kernel has a series expansion.

Theorem 4.1: Mercer's theorem

For a compact set $\mathcal{Y} \subset \mathbb{R}^{K^*}$, $K^* \in \mathbb{N}$, we consider a positive definite kernel $g_*^{(\mathrm{PD})} : \mathcal{Y}^2 \to \mathbb{R}$. Then, there exist nonnegative eigenvalues $\{\lambda_k\}_{k=1}^\infty$, $\lambda_1 \geq \lambda_2 \geq \cdots$, and continuous eigenfunctions $\{\varphi_k\}_{k=1}^\infty$ such that

$$g_*^{(\mathrm{PD})}(\boldsymbol{y}, \boldsymbol{y}') = \sum_{k=1}^\infty \lambda_k \varphi_k(\boldsymbol{y}) \varphi_k(\boldsymbol{y}'), \tag{4.4}$$

for all $\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}$, where the series converges absolutely for each $(\boldsymbol{y}, \boldsymbol{y}')$ and uniformly for $\mathcal{Y}$.

Note that the condition (2) in Minh, Niyogi, and Yao (2006), i.e.,

$$\int_{\mathcal{Y}} \int_{\mathcal{Y}} g_*^{(\mathrm{PD})}(\boldsymbol{y}, \boldsymbol{y}')^2 \, d\boldsymbol{y} \, d\boldsymbol{y}' < \infty,$$

holds since $g_*^{(\mathrm{PD})}$ is continuous and $\mathcal{Y}$ is compact. This theorem can be extended to closed set $\mathcal{Y}$, but we assume compactness for simplifying our argument.

It is obvious that IPS is always PD, because

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_j)\rangle = \| \sum_{i=1}^{n} c_i \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i)\|_2^2 \geq 0.$$

We would like to show the converse: IPS approximates any PD similarities. This is given by the approximation theorem (AT) for IPS shown in Section 4.3.2.

Mercer's theorem asserts that all the PD kernels contain only the positive eigenvalues. By additionally considering negative eigenvalues, it is further generalized to the spectral decomposition, that will be described in Section 4.7.3.

4.3.2 AT for IPS

The AT for IPS is built on a simple idea, that incorporates the UAT for NN into Mercer's theorem (Theorem 4.1). Considering an equivalent form of Mercer's theorem

$$g_*^{(\mathrm{PD})}(\boldsymbol{y}, \boldsymbol{y}') = \lim_{K\to\infty} \langle \tilde{\boldsymbol{\varphi}}_K(\boldsymbol{y}), \tilde{\boldsymbol{\varphi}}_K(\boldsymbol{y}')\rangle, \tag{4.5}$$

where $\tilde{\boldsymbol{\varphi}}_K(\boldsymbol{y}) := (\sqrt{\lambda_1}\varphi_1(\boldsymbol{y}), \sqrt{\lambda_2}\varphi_2(\boldsymbol{y}), \ldots, \sqrt{\lambda_K}\varphi_K(\boldsymbol{y})) \in \mathbb{R}^K$, approximating $\tilde{\boldsymbol{\varphi}}_K(\boldsymbol{f}_*(\cdot))$ by a vector-valued neural network $\boldsymbol{f}_{\mathrm{NN}} : \mathcal{X} \to \mathbb{R}^K$ with a fixed $K \in \mathbb{N}$ immediately leads to the assertion

$$\begin{aligned} g_*^{(\mathrm{PD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) &\overset{\text{Mercer}}{=} \lim_{K'\to\infty} \langle \tilde{\boldsymbol{\varphi}}_{K'}(\boldsymbol{f}_*(\boldsymbol{x})), \tilde{\boldsymbol{\varphi}}_{K'}(\boldsymbol{f}_*(\boldsymbol{x}'))\rangle \\ &\overset{\text{Finite approximation}}{\approx} \langle \tilde{\boldsymbol{\varphi}}_K(\boldsymbol{f}_*(\boldsymbol{x})), \tilde{\boldsymbol{\varphi}}_K(\boldsymbol{f}_*(\boldsymbol{x}'))\rangle \\ &\overset{\text{UAT for NN}}{\approx} \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\rangle. \end{aligned}$$

The assertion is formally given in the following Theorem 4.2, and is more rigorously proved in Appendix B.1.1.

Theorem 4.2: AT for IPS

For $\mathcal{X} = [-M, M]^p$, $M > 0$, and some compact set $\mathcal{Y} \subset \mathbb{R}^{K^*}$, $K^* \in \mathbb{N}$, we consider a continuous function $\boldsymbol{f}_* : \mathcal{X} \to \mathcal{Y}$ and a PD kernel $g_*^{(\mathrm{PD})} : \mathcal{Y}^2 \to \mathbb{R}$. We also consider a 1-hidden layer NN $\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}) = \boldsymbol{A}\sigma(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{c})$ with m hidden units and K outputs, whose parameter $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{c}) \in \boldsymbol{\Theta} := \mathbb{R}^{K\times m} \times \mathbb{R}^{m\times p} \times \mathbb{R}^m$ is to be estimated. $\sigma(\boldsymbol{x})$ applies element-wise $\sigma(\cdot)$ function, which is ReLU or a non-constant, continuous, bounded, and monotonically-increasing activation function. Then, for any $\varepsilon > 0$, there exists $K_0 = K_0(\varepsilon) \in \mathbb{N}$ such that

$$\inf_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\| g_*^{(\mathrm{PD})}\left(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')\right) - \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\rangle \right\|_\infty < \varepsilon$$

with sufficiently large $m = m(K, \varepsilon) \in \mathbb{N}$, for all $K \geq K_0$, where $\|h(\boldsymbol{x}, \boldsymbol{x}')\|_\infty := \sup_{\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}} |h(\boldsymbol{x}, \boldsymbol{x}')|$.

Unlike Mercer's theorem which indicates only the existence of the feature map $\tilde{\boldsymbol{\varphi}}_K$, Theorem 4.2 shows that a neural network $\boldsymbol{f}_{\mathrm{NN}} : \mathcal{X} \to \mathbb{R}^K$ can be implemented so that the IPS $\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle$ eventually approximates the PD similarity $g_*^{(\mathrm{PD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}'))$ arbitrarily well.

4.3.3 Fundamental Limitation of IPS

In the previous Section 4.3.2, IPS is proved to approximate arbitrary PD similarities arbitrarily well. However, there remains a fundamental limitation of IPS; IPS cannot approximate any non-PD similarities, as we formally proved in the following Proposition 4.1.

Proposition 4.1: Limitation of IPS

Let $p, K^*, K \in \mathbb{N}$, $\mathcal{X} \subset \mathbb{R}^p$ be a non-empty set, $\boldsymbol{f}_* : \mathcal{X} \to \mathbb{R}^{K^*}$ be a continuous function whose image is $\mathcal{Y} = \boldsymbol{f}_*(\mathcal{X}) := \{\boldsymbol{f}_*(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{X}\}$, and $g_*^{(\text{non-PD})} : \mathcal{Y}^2 \to \mathbb{R}$ be a non-PD kernel on $\mathcal{Y}^2$. Then, there exists $\xi > 0$ such that

$$\inf_{K \in \mathbb{N}} \inf_{\boldsymbol{f} \in \mathfrak{S}(K)} \left\| g_*^{(\text{non-PD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) - \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}') \rangle \right\|_\infty \geq \xi,$$

where $\|h(\boldsymbol{x}, \boldsymbol{x}')\|_\infty := \sup_{\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}} |h(\boldsymbol{x}, \boldsymbol{x}')|$, and $\mathfrak{S}(K)$ represents the set of functions $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^K$.

Proof. Since $g_*^{(\text{non-PD})}$ is non-PD, there exist $n \in \mathbb{N}, \boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$ and $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathcal{X}$ such that $\langle \boldsymbol{c}, \boldsymbol{H}\boldsymbol{c} \rangle < 0$, where the (i, j)-th entry of the matrix $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is $g_*^{(\text{non-PD})}(\boldsymbol{f}_*(\boldsymbol{x}_i), \boldsymbol{f}_*(\boldsymbol{x}_j))$. Therefore, the matrix $\boldsymbol{H}$ is not PD; considering a set $\mathfrak{P}(n) := \{\tilde{\boldsymbol{F}} \in \mathbb{R}^{n \times n} \mid \tilde{\boldsymbol{F}} \text{ is PD.}\}$ and $\|\boldsymbol{A}\|_\infty := \sup_{1 \leq i, j \leq n} |a_{ij}|$ for $\boldsymbol{A} = (a_{ij}) \in \mathbb{R}^{n \times n}$, there is a positive gap between the non-PD matrix $\boldsymbol{H}$ and any PD matrix $\boldsymbol{F}$ as $\|\boldsymbol{H} - \boldsymbol{F}\|_\infty \geq \inf_{\tilde{\boldsymbol{F}} \in \mathfrak{P}(n)} \|\boldsymbol{H} - \tilde{\boldsymbol{F}}\|_\infty =: \xi > 0$.

Here, we fix $K \in \mathbb{N}$ and $\boldsymbol{f} \in \mathfrak{S}(K)$, and define a matrix $\boldsymbol{F}$ whose (i, j)-th entry is $\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j) \rangle$. Then, $\boldsymbol{F}$ is PD; we have

$$\begin{aligned}
&\sup_{\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}} \left| g_*^{(\text{non-PD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) - \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}') \rangle \right| \\
&\qquad \geq \max_{\boldsymbol{x}, \boldsymbol{x}' \in \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}} \left| g_*^{(\text{non-PD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) - \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}') \rangle \right| \\
&\qquad = \|\boldsymbol{H} - \boldsymbol{F}\|_\infty \geq \inf_{\tilde{\boldsymbol{F}} \in \mathfrak{P}(n)} \|\boldsymbol{H} - \tilde{\boldsymbol{F}}\|_\infty = \xi. \qquad (4.6)
\end{aligned}$$

Since the inequality (4.6) holds for any $K \in \mathbb{N}$ and $f \in \mathfrak{S}(K)$, the assertion is proved by taking $\inf_{K\in\mathbb{N}} \inf_{f\in\mathfrak{S}(K)}$. □

Since any neural network belongs to the set $\mathfrak{S}(K)$, Proposition 4.1 asserts that IPS cannot approximate any of non-PD similarities arbitrarily well, even if NN has a huge amount of hidden units with sufficiently large output dimension K. As there exists a fundamental limitation of IPS, we consider more expressive similarities, that can approximate even non-PD similarities.

4.4 Shifted IPS (SIPS)

Theorem 4.2 shows that IPS can approximate any PD similarities arbitrarily well. However, similarities in general are not always PD; there is a fundamental limitation of IPS to approximate non-PD similarities, as described in Proposition 4.1. To deal with such non-PD similarities, in Section 4.4.1, we first define Conditionally PD (CPD) kernels (Berg, Christensen, and Ressel, 1984; Schölkopf, 2001) which include PD kernels as special cases. In Section 4.4.2, we then propose shifted IPS (SIPS) to approximate CPD similarities. We also propose a simpler modification of SIPS as constantly-SIPS (C-SIPS), and provide intuitive interpretations of SIPS and C-SIPS. In Section 4.4.3, we prove that SIPS and C-SIPS approximate CPD similarities arbitrarily well. In Section 4.4.4, we consider the negative squared distance between two vector-valued NNs, that can be regarded as a special case of SIPS; we also discuss its approximation capability.

4.4.1 CPD Similarities

Here, we introduce similarities based on Conditionally PD (CPD) kernels (Berg, Christensen, and Ressel, 1984; Schölkopf, 2001), in order to consider non-PD similarities that cannot be approximated by the existing IPS.

Definition 4.4: Conditionally PD (CPD)

A kernel g on $\mathcal{Y}^2$ is called *Conditionally PD (CPD)* if it satisfies the inequality $\sum_{i=1}^n \sum_{j=1}^n c_i c_j g(\boldsymbol{y}_i, \boldsymbol{y}_j) \geq 0$ for arbitrary $c_1, c_2, \ldots, c_n \in \mathbb{R}, \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n \in \mathcal{Y}, n \in \mathbb{N}$ satisfying $\sum_{i=1}^n c_i = 0$.

The difference between the definitions of CPD and PD kernels is whether it imposes the constraint $\sum_{i=1}^n c_i = 0$ or not. According to these definitions, CPD kernels include PD kernels as special cases. For a CPD kernel g, the similarity h is also a CPD kernel on $\mathcal{X}^2$.

Later in Section 4.7.4, CPD kernel will be discussed from the perspective of the spectral decomposition; CPD kernel has at most one negative eigenvalue, whereas PD kernel contains only the non-negative eigenvalues.

A simple example of CPD kernel is $g(\boldsymbol{y}, \boldsymbol{y}') = -\|\boldsymbol{y} - \boldsymbol{y}'\|_2^\alpha$ for $0 < \alpha \leq 2$ defined on $\mathbb{R}^K \times \mathbb{R}^K$. Other examples are $-(\sin(y - y'))^2$ and $-\mathbf{1}_{(0,\infty)}(y + y')$

on $\mathbb{R} \times \mathbb{R}$. CPD-ness is a well-established concept with interesting properties (Berg, Christensen, and Ressel, 1984): For any function $u(\cdot)$, $g(\boldsymbol{y}, \boldsymbol{y}') = u(\boldsymbol{y}) + u(\boldsymbol{y}')$ is CPD. Constants are CPD. The sum of two CPD kernels is also CPD. For CPD kernels g with $g(\boldsymbol{y}, \boldsymbol{y}') \leq 0$, CPD-ness holds for $-(-g)^\alpha$ ($\alpha \in (0, 1]$) and $-\log(1-g)$. Furthermore, negative Poincaré distance defined in the following Example 4.1 is also CPD.

Example 4.1: Negative Poincaré distance is CPD

For open unit ball $B^K := \{\boldsymbol{y} \in \mathbb{R}^K \mid \|\boldsymbol{y}\|_2 < 1\}$, negative Poincaré distance between $\boldsymbol{y}, \boldsymbol{y}' \in B^K$ is defined as

$$d_{\text{Poincaré}}(\boldsymbol{y}, \boldsymbol{y}') := \cosh^{-1}\left(1 + 2\frac{\|\boldsymbol{y} - \boldsymbol{y}'\|_2^2}{(1 - \|\boldsymbol{y}\|_2^2)(1 - \|\boldsymbol{y}'\|_2^2)}\right), \tag{4.7}$$

where $\cosh^{-1}(z) = \log(z + \sqrt{z^2 - 1})$. Then, negative Poincaré distance $-d_{\text{Poincaré}}$ is proved to be CPD on $B^K \times B^K$, in Faraut and Harzallah (1974, Corollary 7.4).

Considering the generative model of Section 4.2 with 1-hot data vectors, Poincaré embedding (Nickel and Kiela, 2017) learns parameters $\boldsymbol{y}_i, i = 1, \ldots, n$, by fitting $\sigma(-d_{\text{Poincaré}}(\boldsymbol{y}_i, \boldsymbol{y}_j))$ to the observed $w_{ij} \in \{0, 1\}$. Lorentz embedding (Nickel and Kiela, 2018) reformulates Poincaré embedding with a specific variable transformation, that enables more efficient computation.

Note that the negative Poincaré distance $-d_{\text{Poincaré}}$ is strictly CPD, in the sense that $-d_{\text{Poincaré}}$ is not PD. A counter-example of PD-ness is, for example, $n = 2, K = 2, c_1 = c_2 = 1, \boldsymbol{y}_1 = (1/2, 1/2), \boldsymbol{y}_2 = (0, 0) \in B^2$.

4.4.2 SIPS and Constantly-SIPS (C-SIPS)

For approximating not only PD but also CPD similarities, we propose a novel shifted IPS (SIPS) model

$$h_{\text{SIPS}}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}, \boldsymbol{\psi}) = \langle \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}_j) \rangle + u_{\text{NN}}(\boldsymbol{x}_i) + u_{\text{NN}}(\boldsymbol{x}_j), \tag{4.8}$$

where $\boldsymbol{f}_{\text{NN}} : \mathbb{R}^p \to \mathbb{R}^K$ is a vector-valued NN, and $u_{\text{NN}} : \mathbb{R}^p \to \mathbb{R}$ is a real-valued NN representing a bias term, whose parameters are $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$, respectively.

For illustrating how SIPS expresses CPD similarities, let us consider a very simple CPD similarity called negative squared distance (NSD) $g(\boldsymbol{y}, \boldsymbol{y}') = -\|\boldsymbol{y} - \boldsymbol{y}'\|_2^2$ with the identity map $\boldsymbol{f}(\boldsymbol{x}) := \boldsymbol{x}$. Then, the NSD

$$g(\boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j)) = -\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = \underbrace{\langle \sqrt{2}\boldsymbol{x}_i, \sqrt{2}\boldsymbol{x}_j \rangle}_{\Leftrightarrow \langle \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}_j) \rangle} \underbrace{-\|\boldsymbol{x}_i\|_2^2}_{\Leftrightarrow u_{\text{NN}}(\boldsymbol{x}_i)} \underbrace{-\|\boldsymbol{x}_j\|_2^2}_{\Leftrightarrow u_{\text{NN}}(\boldsymbol{x}_j)}$$

is expressed by SIPS with $f_{\mathrm{NN}}(\boldsymbol{x}) = \sqrt{2}\boldsymbol{x}$ and $u_{\mathrm{NN}}(\boldsymbol{x}) = -\|\boldsymbol{x}\|_2^2$. Later, we show in Theorem 4.3 that SIPS approximates any CPD similarities arbitrarily well.

Here, we also consider a simplified version of SIPS. By assuming $u_{\mathrm{NN}}(\boldsymbol{x}) = -\gamma/2$ for all $\boldsymbol{x}$, SIPS reduces to

$$h_{\text{C-SIPS}}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}, \gamma) = \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_j) \rangle - \gamma, \tag{4.9}$$

where $\boldsymbol{f}_{\mathrm{NN}} : \mathbb{R}^p \to \mathbb{R}^K$ is a vector-valued NN to be trained and $\gamma \gg 0$ is a user-specified parameter. We call (4.9) as constantly-shifted IPS (C-SIPS) model. As will be explained later in Theorem 4.4, C-SIPS also approximates arbitrary CPD similarities in a weaker sense.

If we have no attributes, we use 1-hot vectors (2.6) for $\boldsymbol{x}_i$ in $\mathbb{R}^n$ instead, and $\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i) = \boldsymbol{y}_i \in \mathbb{R}^K$, $u_{\mathrm{NN}}(\boldsymbol{x}_i) = u_i \in \mathbb{R}$ are model parameters. Then SIPS reduces to the matrix decomposition model with biases $h_{\mathrm{SIPS}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle + u_i + u_j$. This model is widely used for recommender systems (Koren, Bell, and Volinsky, 2009) and word embedding such as GloVe (Pennington, Socher, and Manning, 2014), and SIPS is considered as its generalization.

Interpretation of SIPS and C-SIPS

Here, we illustrate the interpretation of the proposed SIPS and C-SIPS. As explained in Section 4.2, exponential function is often applied to the similarity function in practice, such that the function range is restricted to be non-negative. Then, exponential function with SIPS leads to

$$\begin{aligned}
\exp\left(h_{\mathrm{SIPS}}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}, \boldsymbol{\psi})\right) &= \exp\left(\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_j) \rangle + u_{\mathrm{NN}}(\boldsymbol{x}_i) + u_{\mathrm{NN}}(\boldsymbol{x}_j)\right) \\
&= \beta(\boldsymbol{x}_i)\beta(\boldsymbol{x}_j) \exp\left(\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_i), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}_j) \rangle\right) \\
&= \beta(\boldsymbol{x}_i)\beta(\boldsymbol{x}_j) \exp\left(h_{\mathrm{IPS}}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta})\right).
\end{aligned} \tag{4.10}$$

Since $\beta(\boldsymbol{x}) := \exp(u_{\mathrm{NN}}(\boldsymbol{x})) > 0$ can be regarded as the "importance weight" of data vector $\boldsymbol{x}$, the exponential function with SIPS naturally incorporates the weight function $\beta(\boldsymbol{x})$ into the IPS-based model, that is used in a broad range of existing methods. Similarly, applying exponential function to C-SIPS leads to

$$\exp\left(h_{\text{C-SIPS}}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}, \gamma)\right) = \alpha \exp\left(h_{\mathrm{IPS}}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})\right), \tag{4.11}$$

where $\alpha := \exp(-\gamma) > 0$.

As explained in Section 4.2, in graph embedding, link weight w_{ij} is often assumed to follow a distribution $\mathrm{Po}(\exp(h(\boldsymbol{x}_i, \boldsymbol{x}_j)))$ where $\mathrm{Po}(\mu)$ represents Poisson distribution whose expectation is μ. Considering the C-SIPS model, the parameter α in

$$\mathrm{Po}(\exp(h_{\text{C-SIPS}}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}, \gamma))) = \mathrm{Po}(\alpha \exp(h_{\mathrm{IPS}}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta})))$$

regulates the sparseness of $\{w_{ij}\}$; the same model is already defined as PMvGE (3.3) in previous Chapter 3. Therefore, PMvGE is in fact capable of approximating CPD similarities, by virtue of the sparseness parameter $\alpha > 0$.

4.4.3 ATs for SIPS and C-SIPS

In this section, we show in the following Theorem 4.3 that SIPS approximates any CPD similarity $g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}'))$ where $\boldsymbol{f}_* : \mathcal{X} \to \mathcal{Y}$ is a vector-valued continuous function. Thus it overcomes the fundamental limitation of IPS.

Theorem 4.3: AT for SIPS

Symbols and assumptions are the same as those of Theorem 4.2 but $g_*^{(\mathrm{CPD})} : \mathcal{Y}^2 \to \mathbb{R}$ is a CPD kernel. We consider 1-hidden layer NNs $\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{\sigma}(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}) \in \mathbb{R}^K$ and $u_{\mathrm{NN}}(\boldsymbol{x}) = \langle \boldsymbol{e}, \boldsymbol{\sigma}(\boldsymbol{F}\boldsymbol{x} + \boldsymbol{o})\rangle \in \mathbb{R}$ with m_f and m_u hidden units, respectively. Their parameters $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{c}) \in \boldsymbol{\Theta} := \mathbb{R}^{K\times m_f} \times \mathbb{R}^{m_f \times p} \times \mathbb{R}^{m_f}$ and $\boldsymbol{\psi} = (\boldsymbol{e}, \boldsymbol{F}, \boldsymbol{o}) \in \boldsymbol{\Psi} := \mathbb{R}^{K\times m_u} \times \mathbb{R}^{m_u \times p} \times \mathbb{R}^{m_u}$ are to be estimated. Then, for any $\varepsilon > 0$, there exists $K_0 = K_0(\varepsilon) \in \mathbb{N}$ such that

$$\inf_{\substack{\boldsymbol{\theta}\in\boldsymbol{\Theta}\\ \boldsymbol{\psi}\in\boldsymbol{\Psi}}} \left\| g_*^{(\mathrm{CPD})}\left(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')\right) - \left(\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\rangle + u_{\mathrm{NN}}(\boldsymbol{x}) + u_{\mathrm{NN}}(\boldsymbol{x}')\right) \right\|_\infty < \varepsilon$$

with sufficiently large $m_f = m_f(K, \varepsilon), m_u = m_u(\varepsilon) \in \mathbb{N}$, for all $K \geq K_0$.

Proof. Basic idea for the proof is shown as follows. Berg, Christensen, and Ressel (1984) Lemma 2.1 shows the equivalence between CPD-ness of a kernel $g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}')$ and PD-ness of

$$g_0(\boldsymbol{y}, \boldsymbol{y}') := g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}') + g_*^{(\mathrm{CPD})}(\boldsymbol{y}_0, \boldsymbol{y}_0) - g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}_0) - g_*^{(\mathrm{CPD})}(\boldsymbol{y}', \boldsymbol{y}_0) \quad (4.12)$$

for any fixed $\boldsymbol{y}_0 \in \mathcal{Y}$. Therefore, any CPD similarity has an expression

$$g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) = g_0(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) + u_*(\boldsymbol{x}) + u_*(\boldsymbol{x}') \quad (4.13)$$

where $u_*(\boldsymbol{x}) := g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{y}_0) - \frac{1}{2} g_*^{(\mathrm{CPD})}(\boldsymbol{y}_0, \boldsymbol{y}_0)$. Approximating the PD-similarity $g_0(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}'))$ and the continuous function $u_*(\boldsymbol{x})$ by IPS and the NN $u_{\mathrm{NN}}(\boldsymbol{x})$, respectively, immediately proves the assertion. See Appendix B.1.2 for the proof. □

Furthermore, the following Theorem 4.4 proves that C-SIPS given in eq. (4.9) approximates CPD similarities in a weaker sense.

Theorem 4.4: AT for C-SIPS

Symbols and assumptions are the same as those of Theorem 4.2 but $g_*^{(\mathrm{CPD})}$ is a CPD kernel. Here, we specify a sufficiently large $r \gg 0$ and

$\gamma = O(r^2)$. We consider a 1-hidden layer NN $\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}) = \boldsymbol{A}\sigma(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}) \in \mathbb{R}^K$ with m hidden units, and $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{c}) \in \boldsymbol{\Theta} := \mathbb{R}^{K\times m} \times \mathbb{R}^{m\times p} \times \mathbb{R}^m$ is a parameter to be estimated. Then, for any $\varepsilon > 0$, there exists $K_0 = K_0(\varepsilon) \in \mathbb{N}$ such that

$$\inf_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\| g_*^{(\mathrm{CPD})}\left(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')\right) - \left(\left\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\right\rangle - \gamma\right) \right\|_\infty < \varepsilon + O(r^{-2})$$

with sufficiently large $m = m(K, \varepsilon) \in \mathbb{N}$, for all $K \geq K_0$.

See Appendix B.1.3 for more rigorous proof.

There is an additional error term of $O(\gamma^{-1})$ in Theorem 4.4. A large γ will reduce the error, but then it may lead to unstable computation for finding an optimal NN. Conversely, a small γ increases the upper bound of the approximation error. Thus, if available, we prefer SIPS in terms of both computational stability and small approximation error.

In the following section, we last consider the approximation capability of negative squared distance (NSD) between two feature vectors $\boldsymbol{y} = \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{y}' = \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')$ that can be regarded as a special case of SIPS.

4.4.4 NSD and Its Approximation Capability

Assuming that

$$u_{\mathrm{NN}}(\boldsymbol{x}) = -\frac{1}{2}\|\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x})\|_2^2, \tag{4.14}$$

SIPS reduces to the negative squared distance (NSD) between $\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')$, i.e.,

$$-\frac{1}{2}\|\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}) - \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\|_2^2. \tag{4.15}$$

(4.15) is expected to inherit the high approximation capability of SIPS. However, AT for SIPS, that is shown as Theorem 4.3, is not applicable to the NSD (4.15) due to the constraint (4.14). The NSD (4.15) is in fact not capable of approximating arbitrary CPD similarities, as $(4.15) = 0$ for all $\boldsymbol{x} = \boldsymbol{x}'$ whereas CPD similarities between two same vectors are not restricted to be 0.

For showing the approximation capability of the similarity (4.15), we here consider an additional condition

$$g(\boldsymbol{y}, \boldsymbol{y}) = 0, \quad \forall \boldsymbol{y} \in \mathcal{Y} \tag{4.16}$$

for kernel $g : \mathcal{Y}^2 \to \mathbb{R}$. The similarity (4.15) can approximate any similarities, that are equipped with any CPD kernel $g^{(\mathrm{CPD})}$ satisfying the condition (4.16).

The proof is straightforward. Similarly to Mercer's theorem asserting that any PD kernel $g^{(\mathrm{PD})}$ can be decomposed as in eq. (4.5), i.e., $g^{(\mathrm{PD})}(\boldsymbol{y}, \boldsymbol{y}') = \lim_{K\to\infty}\langle\tilde{\boldsymbol{\varphi}}_K(\boldsymbol{y}), \tilde{\boldsymbol{\varphi}}_K(\boldsymbol{y}')\rangle$, Schölkopf (2001) Proposition 5 asserts that any CPD kernel $g^{(\mathrm{CPD})}$ satisfying $g^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}) = 0$ can be decomposed as

$$g^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}') = \lim_{K\to\infty}\left(-\|\tilde{\boldsymbol{\phi}}_K(\boldsymbol{y}) - \tilde{\boldsymbol{\phi}}_K(\boldsymbol{y}')\|_2^2\right)$$

for some $\tilde{\boldsymbol{\phi}}_K : \mathcal{X} \to \mathbb{R}^K$. Therefore, considering a CPD kernel g_* and a continuous $\boldsymbol{f}_*$, approximating $\tilde{\boldsymbol{\phi}}_K(\boldsymbol{f}(\boldsymbol{x}))$ by the NN $\boldsymbol{f}_{\mathrm{NN}}$ yields

$$\begin{aligned} g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) &= \lim_{K'\to\infty}\left(-\|\tilde{\boldsymbol{\phi}}_{K'}(\boldsymbol{f}_*(\boldsymbol{x})) - \tilde{\boldsymbol{\phi}}_{K'}(\boldsymbol{f}_*(\boldsymbol{x}'))\|_2^2\right) \\ &\overset{\text{(Finite approximation)}}{\approx} -\|\tilde{\boldsymbol{\phi}}_K(\boldsymbol{f}_*(\boldsymbol{x})) - \tilde{\boldsymbol{\phi}}_K(\boldsymbol{f}_*(\boldsymbol{x}'))\|_2^2 \\ &\overset{\text{(UAT for NN)}}{\approx} -\|\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}) - \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\|_2^2. \end{aligned}$$

The coefficient $1/2$ in (4.15) can be ignored, as any CPD kernel remains CPD if it is multiplied by any positive constant; the approximation capability of (4.15) is thus proved.

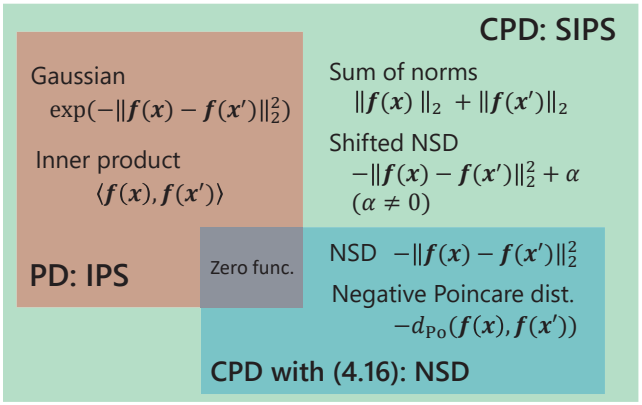


FIGURE 4.2: Classes of similarities. IPS and SIPS can approximate any PD and CPD similarities, respectively. NSD (4.15) can approximate the similarities, that are equipped with any CPD kernel satisfying the condition (4.16).

We last show that, only one PD kernel satisfying the condition (4.16) is the zero function, which is $g(\boldsymbol{y}, \boldsymbol{y}') = 0$ for all $\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}$. The proof is obtained as follows.

Proof. If a PD kernel g satisfies the condition (4.16), it holds that

$$\sum_{i,j=1}^{2} c_i c_j g(\boldsymbol{y}_i, \boldsymbol{y}_j) = 2c_1 c_2 g(\boldsymbol{y}_1, \boldsymbol{y}_2) \geq 0$$

for all $c_1, c_2 \in \mathbb{R}$ and $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}$. Considering the two cases $(c_1, c_2) = (1, 1)$ and $(c_1, c_2) = (1, -1)$, two inequalities $g(\boldsymbol{y}_1, \boldsymbol{y}_2) \geq 0$ and $g(\boldsymbol{y}_1, \boldsymbol{y}_2) \leq 0$

should be satisfied simultaneously for all $\boldsymbol{y}_1, \boldsymbol{y}_2$. These inequalities hold simultaneously if and only if $g(\boldsymbol{y}_1, \boldsymbol{y}_2) = 0$. Thus the assertion is proved. □

The classes of similarities, that are approximated by IPS, SIPS, and NSD, are summarized in Figure 4.2. CPD similarity encompasses all the similarities in line with both IPS and NSD. Furthermore, SIPS outperforms both IPS and NSD numerically, as will be demonstrated in the experiments shown in Section 4.6.2.

4.5 Approximation Error Rates

Thus far, we showed universal approximation capabilities of IPS, SIPS, and C-SIPS in Theorems 4.2-4.4. In this section, we evaluate error rates for these approximation, by assuming some additional conditions. These conditions are used for leveraging eigenvalue decay rate of PD kernels (Cobos and Kühn, 1990, Theorem 4), that is shown in Theorem B.1 in Appendix B.2.1, and a slight modification of approximation error rate for NNs (Yarotsky, 2018), that is shown in Theorem 2.1 in Chapter 2.

Regularity conditions

We consider the following conditions on the function $\boldsymbol{f}_*$ and the kernel g_* for the underlying true similarity $g_*(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}'))$.

(C-1) Eigenfunctions $\{\phi_k(\boldsymbol{y})\}_{k=1}^{\infty}$ of $g_*(\boldsymbol{y}, \boldsymbol{y}')$ defined in Theorem 4.1 are continuously differentiable, i.e., C^1, and uniformly bounded in the sense of $\sup_{k\in\mathbb{N},\boldsymbol{y}\in\mathcal{Y}} |\phi_k(\boldsymbol{y})| < \infty$ and $\sup_{k\in\mathbb{N},\boldsymbol{y}\in\mathcal{Y}} \lambda_k \|\partial\phi_k(\boldsymbol{y})/\partial\boldsymbol{y}\|_2^2 < \infty$.

(C-2) $g_*(\boldsymbol{y}, \boldsymbol{y}')$ is C^1.

(C-3) $\boldsymbol{f}_*$ is C^1.

NN architecture

We employ a set of K-dimensional vector-valued NNs for $\mathcal{X} = [-M, M]^p$. The activation function is confined to ReLU $\sigma(z) := \max\{0, z\}$. Let $L \in \mathbb{N}$ be the number of hidden layers, i.e., depth, of the NN, and let $W \in \mathbb{N}$ be the total number of weights in the NN. For example, $L = 1$ and W is the number of elements in $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{c}$ in Theorems 4.2. Instead of the fixed network architecture, here we consider a class of architectures specified by W with a specific growing rate of the depth L. For $0 \leq \alpha \leq 1$, a set of NNs are denoted as

$$\mathfrak{S}_\alpha(W, K) := \{\boldsymbol{f}_{\mathrm{NN}} : \mathcal{X} \to \mathbb{R}^K \mid \text{neural network } \boldsymbol{f}_{\mathrm{NN}} \text{ has } W \text{ weights with depth } L = O((W/K)^\alpha)\},$$

where $W/K \to \infty$. This set $\mathfrak{S}_\alpha(W,K)$ is already defined in (2.2) in Chapter 2.1; $\alpha = 0$ and $\alpha = 1$ correspond to constant-depth shallow NNs and constant-width deep NNs, respectively.

Approximation error rates

Considering the set $\mathfrak{S}_\alpha(W,K)$ and its universal approximation theorem that is already shown in Lemma 2.1 in Chapter 2, we obtain the following Theorem 4.5.

Theorem 4.5: Approximation error rate for IPS

Symbols and assumptions are the same as those of Theorem 4.2 except for the additional conditions (C-1) and (C-2) for $g_*^{(\mathrm{PD})}$ and (C-3) for $\boldsymbol{f}_*$. We consider the set of NNs $\boldsymbol{f}_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_f,K)$ for $W_f \in \mathbb{N}$. Then, the approximation error rate of IPS is given by

$$\inf_{\boldsymbol{f}_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_f,K)} \left\| g_*^{(\mathrm{PD})}(\boldsymbol{f}_*(\boldsymbol{x}),\boldsymbol{f}_*(\boldsymbol{x}')) - \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\rangle \right\|_\infty = O\Big(K^{-\frac{1}{K^*}} + K^{\frac{1}{2}+\frac{1+\alpha}{p}} W_f^{-\frac{1+\alpha}{p}}\Big). \tag{4.17}$$

See Appendix B.2.2 for the rigorous proof. Roughly speaking, the proof is obtained by combining the decay of eigenvalues for kernel g_* (Cobos and Kühn, 1990, Theorem 4) and approximation error rate for NNs (Yarotsky, 2018).

In the above result, the first term $O(K^{-1/K^*})$ in the error rate represents the finite approximation error by Mercer's theorem, and the second term $O(K^{\frac{1}{2}+\frac{1+\alpha}{p}} W_f^{-\frac{1+\alpha}{p}})$ is attributed to the approximation error of $\boldsymbol{f}_{\mathrm{NN}}$.

The error rate for SIPS is also similarly evaluated.

Theorem 4.6: Approximation error rate for SIPS

Symbols and assumptions are the same as those of Theorem 4.3 except for the additional conditions (C-1) for g_0 of (4.12), (C-2) for $g_*^{(\mathrm{CPD})}$, and (C-3) for $\boldsymbol{f}_*$. Instead of the 1-hidden layer NN, we consider the set of NNs $\boldsymbol{f}_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_f,K)$ for $W_f \in \mathbb{N}$ and $u_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_u,1)$ for $W_u \in \mathbb{N}$. Then the approximation error rate of SIPS is given by

$$\inf_{\substack{\boldsymbol{f}_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_f,K) \\ u_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_u,1)}} \left\| g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}),\boldsymbol{f}_*(\boldsymbol{x}')) - (\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\rangle + u_{\mathrm{NN}}(\boldsymbol{x}) + u_{\mathrm{NN}}(\boldsymbol{x}')) \right\|_\infty = O\Big(K^{-\frac{1}{K^*}} + K^{\frac{1}{2}+\frac{1+\alpha}{p}} W_f^{-\frac{1+\alpha}{p}} + W_u^{-\frac{1+\alpha}{p}}\Big). \tag{4.18}$$

Proof is almost similar to that of IPS; see Appendix B.2.3 for the rigorous proof.

In Theorems 4.5 and 4.6, the commonly appearing term $O(K^{-1/K^*})$ may be a bottleneck when K^* is very large. We may specify $W_f = O(K^{1+\frac{p}{1+\alpha}(\frac{1}{K^*}+\frac{1}{2})}) \approx O(K^{1+\frac{p}{2(1+\alpha)}})$ and $W_u = O(K^{\frac{p}{(1+\alpha)K^*}})$ so that the overall approximation error rate is $O(K^{-1/K^*})$.

4.6 Numerical Experiments

In this section, we conduct numerical experiments to compare existing IPS, proposed SIPS, and its simper variant C-SIPS. We first examine the approximation capability of IPS, SIPS and C-SIPS in Section 4.6.1. Subsequently, we perform these similarities on real-world datasets in Section 4.6.2.

4.6.1 Experiments on Approximation Capability of SIPS

In this section, we examine whether SIPS and C-SIPS have higher approximation capabilities than IPS, by leveraging synthetic datasets generated via CPD similarities.

- **Kernels:** Three types of kernels are considered as $g_*(\boldsymbol{y}, \boldsymbol{y}')$ for generating simulation data: (i) cosine similarity $\langle \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_2}, \frac{\boldsymbol{y}'}{\|\boldsymbol{y}'\|_2} \rangle$, (ii) negative squared distance $-\|\boldsymbol{y} - \boldsymbol{y}'\|_2^2$, (iii) negative Poincaré distance defined in Example 4.1. These kernels are PD, CPD, and CPD, in this order.

- **Synthetic data:** For kernels (i) and (ii), data vectors $\{\boldsymbol{x}_i\} \subset \mathbb{R}^p$ are generated independently by uniform distribution over $[-2, 2]^p$ with $p = 5$. Feature vectors of dimensions $K^* = 4$ are computed by a continuous map $\boldsymbol{f}_*(\boldsymbol{x}) := (x_1, \cos x_2, \exp(-x_3), \sin(x_4 - x_5)) \in \mathbb{R}^4$, and similarity values are given as

$$h_{ij}^* := g_*(\boldsymbol{f}_*(\boldsymbol{x}_i), \boldsymbol{f}_*(\boldsymbol{x}_j)) \tag{4.19}$$

  for all $1 \leq i, j \leq n$. For kernel (iii), data vectors $\{\boldsymbol{x}_i\} \subset \mathbb{R}^p$ are generated by $\boldsymbol{x}_i := r_i \tilde{\boldsymbol{x}}_i / \|\tilde{\boldsymbol{x}}_i\|_2$ with $\tilde{\boldsymbol{x}}_i \overset{\text{i.i.d.}}{\sim} N_p(\boldsymbol{0}, \boldsymbol{I})$, $r_i \overset{\text{i.i.d.}}{\sim} B(5, 1)$, so that $\|\boldsymbol{x}_i\|_2 < 1$. $B(\alpha, \beta)$ is the beta distribution with parameters $\alpha, \beta > 0$, and $\mathcal{N}_p(\boldsymbol{0}, \boldsymbol{I})$ represents the p-variate standard normal distribution. Feature vectors of dimensions $K^* = p = 5$ are computed by the identity map $\boldsymbol{f}_*(\boldsymbol{x}) = \boldsymbol{x}$, and similarity values $\{h_{ij}^*\}$ are given as (4.19). In order to simplify the experiment just for illustrating the differences of similarity models, we do not generate $\{w_{ij}\}$ and treat $\{h_{ij}^*\}$ as observed samples. For each setting (i)-(iii), we generate $n = 1{,}000$ training samples and $n = 3{,}000$ test samples.

- **NN architecture:** Three models are considered for similarity function: (i) IPS (existing) defined in eq. (4.2), (ii) SIPS (proposed) defined in eq. (4.8), (iii) C-SIPS (the simpler variant of SIPS) defined in eq. (4.9).

For each model, $f_{\mathrm{NN}} : \mathbb{R}^p \to \mathbb{R}^K, u_{\mathrm{NN}} : \mathbb{R}^p \to \mathbb{R}$ are two-layer NNs with T hidden units and ReLU activations. We denote T as "♯units".

- **Training:** For training the three similarity models, we minimize the mean squared error between h^*_{ij} and $h(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The loss functions are ℓ^2-regularized with a coefficient 0.01, and they are minimized by 10,000 iterations of full-batch gradient descent. The learning rate is starting from 0.001 and attenuated by 1/10 for every 100 iterations.

- **Evaluation:** Neural networks are trained with 1,000 samples, and they are evaluated by the Mean Squared Prediction Error (MSPE) with respect to 3,000 test samples. We compute the average and the standard deviation of 5 runs for each setting.

Results: In the following plots, black, blue, and red lines represent the MSPE of IPS (Existing), SIPS (Proposed), and its simpler variant C-SIPS, respectively.

Table 4.1 shows the MSPE for the cosine similarity. Error bar shows the standard deviation (=1σ). In accordance with the theory, all of IPS, SIPS, and C-SIPS show the good approximation performance since cosine similarity is PD. Interestingly, output dimension $K = 3$ is sufficient to approximate the function $g_*(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}'))$ regardless of the number of hidden units.

TABLE 4.1: Cosine similarity: $\langle \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_2}, \frac{\boldsymbol{y}'}{\|\boldsymbol{y}'\|_2} \rangle$

(A) ♯units= 10 (B) ♯units= 100 (C) ♯units= 1000

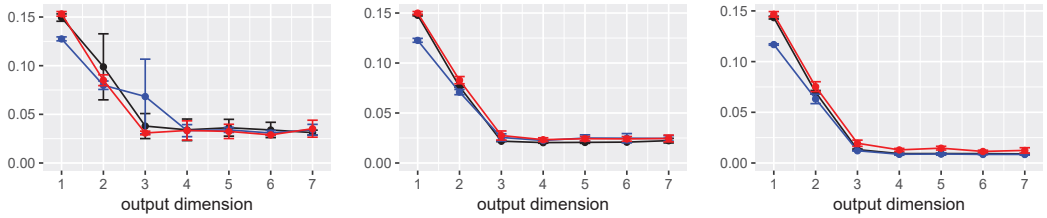


Table 4.2 shows the MSPE for the negative squared distance (NSD). In accordance with the theory, NSD is well approximated by SIPS but not by IPS due to its CPD-ness. The approximation error of SIPS with $m = 1{,}000$ becomes almost zero at $K = 4$ as expected from $K^* = 4$. In theory, C-SIPS can approximate CPD kernels, but it does not perform well in this setting. Since C-SIPS requires the parameter γ to be very large for minimizing the approximation error, its computation becomes unstable in some cases.

Table 4.3 shows the MSPE for $-d_{\text{Poincaré}}$. In accordance with the theory, the generated similarity values are well approximated by both SIPS and C-SIPS, but not by IPS due to the CPD-ness. The approximation error of SIPS and C-SIPS with $m \geq 100$ becomes almost zero at $K = 5$ as expected from $K^* = 5$. However, the standard deviation of MSPE for C-SIPS is large with $T = 10$.

TABLE 4.2: Negative squared distance: $-\|\boldsymbol{y}-\boldsymbol{y}'\|_2^2$

(A) ♯units= 10 (B) ♯units= 100 (C) ♯units= 1000

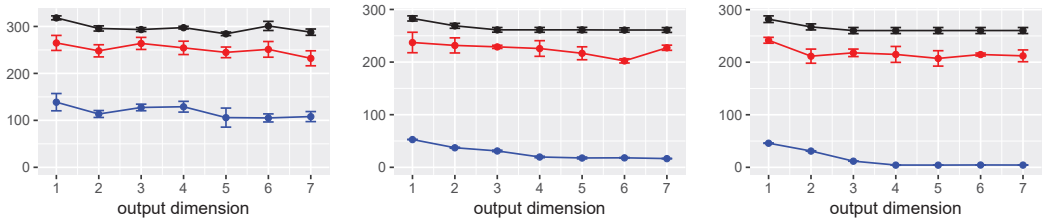
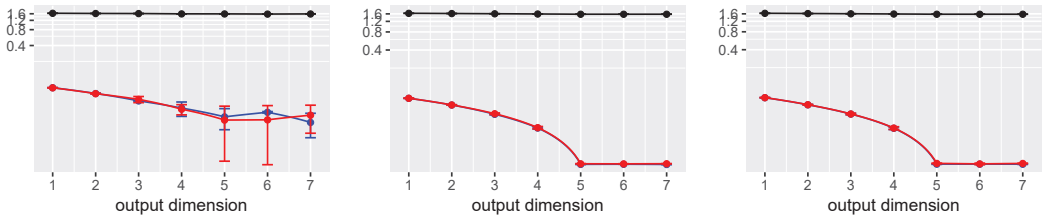


TABLE 4.3: Negative Poincaré distance: $-d_{\text{Poincaré}}(\boldsymbol{y},\boldsymbol{y}')$

(A) ♯units= 10 (B) ♯units= 100 (C) ♯units= 1000



4.6.2 Experiments on Graph Embedding with SIPS

In this section, we evaluate similarity models (NSD, Poincaré, IPS, SIPS) on two real-world datasets: Co-authorship network dataset (Prado et al., 2013) and WordNet dataset.

Experiment on co-authorship network

- **Dataset:** Co-authorship network dataset (Prado et al., 2013) consists of $n = 42,252$ nodes and $210,320$ undirected edges. Each node v_i represents an author, and data vector $\boldsymbol{x}_i \in \mathbb{R}^{33}$ ($p = 33$) represents the numbers of publications in 29 conferences/journals and 4 microscopic topological properties describing the direct neighborhood of the node. Adjacency matrix $\boldsymbol{W} = (w_{ij}) \in \{0,1\}^{n\times n}$ represents the co-authorship relations: $w_{ij} = w_{ji} = 1$ if v_i and v_j have any co-authorship relation, and $w_{ij} = w_{ji} = 0$ otherwise.

- **Preprocessing:** We split authors into training set (90%) and test set (10%). Co-authorship relations for the test set are treated as unseen. We use 10% of the training set as validation set.

- **Author feature vectors:** Using the data vectors for authors $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^p$, feature vectors $\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathbb{R}^K$ are computed via a neural network $\boldsymbol{y}_i = \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}_i)$. We employ 1-hidden layer perceptron with $10,000$ hidden units and ReLU activation function. For implementing SIPS, one of the K output units of $\boldsymbol{f}_{\text{NN}}(\boldsymbol{x}_i)$ is used for the bias term $u_i = u_{\text{NN}}(\boldsymbol{x}_i)$, so actually the feature vector is computed as $(\boldsymbol{y}_i, u_i) = \boldsymbol{f}_{\text{NN}}(\boldsymbol{x}_i) \in \mathbb{R}^K$ with

$\boldsymbol{y}_i \in \mathbb{R}^{K-1}$. Model parameters are trained by maximizing the objective

$$\sum_{1 \le i \ne j \le n} w_{ij} \log \frac{\exp(h(\boldsymbol{x}_i, \boldsymbol{x}_j))}{\sum_{k \in \mathcal{S}_r(\mathcal{N}_{ij})} \exp(h(\boldsymbol{x}_i, \boldsymbol{x}_k))}, \tag{4.20}$$

where $h : \mathcal{X}^2 \to \mathbb{R}$ is a similarity function and $\mathcal{S}_r(\mathcal{N}_{ij})$ is a subset that consists of $r = 10$ entries randomly sampled from $\mathcal{N}_{ij} := \{k | 1 \le k \le n, w_{ik} = 0\} \cup \{j\}$.

- **Similarity models:** (i) NSD uses $h(\boldsymbol{x}_i, \boldsymbol{x}_j) = -\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$. (ii) Poincaré embedding (Nickel and Kiela, 2017) uses $h(\boldsymbol{x}_i, \boldsymbol{x}_j) = -d_{\text{Poincare}}(\boldsymbol{y}_i, \boldsymbol{y}_j)$ defined in (4.7). (iii) IPS uses $h(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle$. (iv) SIPS uses $h(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle + u_i + u_j$.
- **Optimization:** All parameters are initialized as He et al. (2015) and trained by Adam (Kingma and Ba, 2015) with initial learning rate 0.01 and batch size 64. The number of iterations is 300,000. To ensure robust comparison, we save model parameters at every 5,000 iterations, and select the best performance parameters tested on the validation set.

Results: Models are evaluated by ROC-AUC (Bradley, 1997) on the task of predicting unseen co-authorship relations. ROC-AUC scores are shown on the left-hand side of Table 4.4. Although NSD demonstrates a good performance for $K = 2$, SIPS outperforms the other methods for $K = 5, 10, 20$.

TABLE 4.4: Experiments on Co-authorship network evaluated by ROC-AUC score (higher is better). Sample average and the standard deviation of 5 runs are shown.

	Co-authorship network			
	$K = 2$	$K = 5$	$K = 10$	$K = 20$
NSD	**0.8220** $\pm$ 0.010	0.8655 $\pm$ 0.014	0.8771 $\pm$ 0.012	0.8651 $\pm$ 0.033
Poincaré	0.7071 $\pm$ 0.021	0.8738 $\pm$ 0.001	0.8822 $\pm$ 0.001	0.8835 $\pm$ 0.001
IPS	0.7802 $\pm$ 0.005	0.8830 $\pm$ 0.001	0.8955 $\pm$ 0.001	0.8956 $\pm$ 0.001
SIPS	0.7811 $\pm$ 0.001	**0.8853** $\pm$ 0.001	**0.8964** $\pm$ 0.002	**0.8974** $\pm$ 0.001

Experiment on WordNet

- **Dataset:** WordNet dataset (Miller, 1995) is a lexical resource that contains a variety of nouns and their relations. For instance, a noun "mammal" represents a superordinate concept of a noun "dog", thus these two words have hypernymy relation. We preprocess WordNet dataset in the same way as Nickel and Kiela (2017). We used a subset of the graph with $n = 4027$ nouns and 53,905 hierarchical relations by extracting all the nouns subordinate to "animal". Each noun is represented by v_i, and relations are represented by adjacency matrix $\boldsymbol{W} =$

$(w_{ij}) \in \{0,1\}^{n\times n}$, where $w_{ij} = w_{ji}$ represents any hypernymy relation, including transitive closure, between v_i and v_j.

- **Word feature vectors:** Since nodes have no attributes, data vectors are formally treated as 1-hot vectors in $\mathbb{R}^n$. Instead of learning neural networks, the distributed representations $\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathbb{R}^K$ of words are learned by maximizing the objective (4.20) with $r = 20$ for NSD, Poincaré and IPS, and $\{(\boldsymbol{y}_i, u_i)\}_{i=1}^n \subset \mathbb{R}^K$ are learned for SIPS. Similarity models are the same as those of co-authorship network experiments.
- **Optimization:** the most settings are the same as the experiments for co-authorship network. All parameters are initialized as He et al. (2015) and trained by Adam with initial learning rate 0.001 and batch size 128. The number of iterations is 150,000.

Results: Models are evaluated by ROC-AUC of reconstruction error on the task of reconstructing hierarchical relations in the same way as Nickel and Kiela (2017). ROC-AUC score is listed on the right-hand side of Table 4.5. SIPS outperforms the other methods for $K = 2, 20$, and it is competitive to Poincaré embedding for $K = 5, 10$.

TABLE 4.5: Experiments on WordNet evaluated by ROC-AUC score (higher is better). Sample average and the standard deviation of 5 runs are shown.

	WordNet			
	$K = 2$	$K = 5$	$K = 10$	$K = 20$
NSD	0.7924 ± 0.0072	0.8997 ± 0.0009	0.9569 ± 0.0005	0.9836 ± 0.0001
Poincaré	0.8401 ± 0.0073	$\mathbf{0.9792} \pm 0.0006$	$\mathbf{0.9866} \pm 0.0003$	0.9851 ± 0.0002
IPS	0.7245 ± 0.0056	0.7604 ± 0.0055	0.7688 ± 0.0023	0.7918 ± 0.0018
SIPS	$\mathbf{0.9632} \pm 0.0008$	0.9766 ± 0.0006	0.9825 ± 0.0005	$\mathbf{0.9865} \pm 0.0004$

Visualization: Feature vectors computed by SIPS with $K = 5$ are visualized in Figure 4.3. Since $(\boldsymbol{y}_i, u_i) \in \mathbb{R}^5$ for SIPS, we actually used $\boldsymbol{y}_i \in \mathbb{R}^4$ for the visualization. We extracted 97 words from the $n = 4027$ nouns, and applied t-SNE to $\{\boldsymbol{y}_i\}$ for the extracted words. Words with any hypernymy relations are connected by segments. In other words, v_i and v_j are connected when $w_{ij} = 1$. For extracting the 97 words, we chose the word "animal" as the root. Then chose four subordinate words ("mammal", "fish", "reptile", "invertebrate") connected to the root, and sampled more subordinate words from these four words, so that the total number of words becomes 97. Words are grouped by the four subordinate words of the root, which are indicated by the colors.

4.7 Inner Product Difference Similarity (IPDS)

SIPS is proved to approximate arbitrary CPD kernels, which include a broad range of kernels, and SIPS also empirically demonstrates a good performance

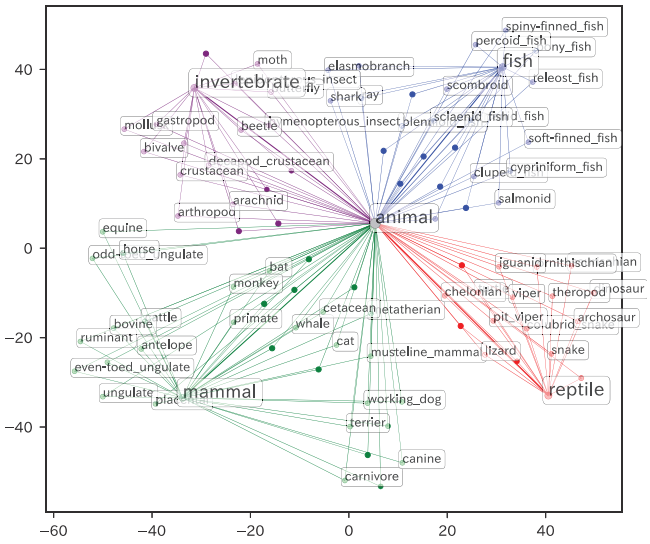


FIGURE 4.3: Visualization of word feature vectors for WordNet dataset computed by GE with our proposed SIPS model. See Section 4.6.2 for details.

in the previous Section 4.6. However, there exists a variety of non-CPD kernels, such as Epanechnikov kernel, Gaussian mixture and Jeffrey's divergence, and SIPS is in fact not capable of approximating similarities based on such non-CPD kernels as shown in the following Section 4.7.1. To approximate the similarities based on such non-CPD kernels, we propose IPDS in Section 4.7.2, yet based on inner product, with high approximation capability beyond SIPS. Its high approximation capability is provided in Section 4.7.3. The approximation theorem of IPDS is based on the spectral decomposition of the kernel; we last revisit the PD and CPD kernels in Section 4.7.4 from the perspective of the spectral decomposition.

4.7.1 Fundamental Limitation of SIPS

Although SIPS is capable of approximating wider class of functions than IPS, still there exists a fundamental limitation of SIPS, as shown in the following Proposition 4.2.

Proposition 4.2: Limitation of SIPS

Symbols and assumptions are the same as those of Proposition 4.1 but $g_*^{(\text{non-CPD})} : \mathcal{Y}^2 \to \mathbb{R}$ is a non-CPD kernel on $\mathcal{Y}^2$. Then, there exists $\xi > 0$ such that,

$$\inf_{K \in \mathbb{N}} \inf_{f \in \mathfrak{S}(K), u \in \mathfrak{S}(1)} \left\| g_*^{(\text{non-CPD})}(f_*(x), f_*(x')) \right.$$

$$-(\langle f(x), f(x')\rangle + u(x) + u(x'))\Big\|_\infty \geq \xi,$$

where $\mathfrak{S}(K)$ represents the set of functions $f : \mathcal{X} \to \mathbb{R}^K$

Proof. Proof is almost the same as that of Proposition 4.1. Since $g^{(\text{non-CPD})}$, there exist $n \in \mathbb{N}, \boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}$ and $\{x_i\}_{i=1}^n \subset \mathcal{X}$ such that $\langle \boldsymbol{c}, \boldsymbol{H}\boldsymbol{c}\rangle < 0$ and $\langle \boldsymbol{c}, \boldsymbol{1}\rangle = 0$, where the (i,j)-th entry of the matrix $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ is $g^{(\text{non-CPD})}(f_*(x_i), f_*(x_j))$. Considering a set of CPD matrices, i.e.,

$$\mathfrak{C}(n) := \{\boldsymbol{F} \in \mathbb{R}^{n\times n} \mid \langle \boldsymbol{c}, \boldsymbol{F}\boldsymbol{c}\rangle \geq 0, \quad \forall \boldsymbol{c} \in \mathbb{R}^n \text{ satisfying } \langle \boldsymbol{c}, \boldsymbol{1}\rangle = 0\},$$

obviously it holds that $\boldsymbol{H} \notin \mathfrak{C}(n)$; there is a positive gap between the non-CPD matrix $\boldsymbol{H}$ and any CPD matrix $\boldsymbol{F}$ as

$$\|\boldsymbol{H} - \boldsymbol{F}\|_\infty \geq \inf_{\tilde{\boldsymbol{F}} \in \mathfrak{C}(n)} \|\boldsymbol{H} - \tilde{\boldsymbol{F}}\|_\infty =: \xi > 0.$$

The assertion is proved similarly to Proposition 4.1. □

Therefore, the SIPS model cannot approximate the similarities, which are equipped with the non-CPD kernels. Simple examples of such non-CPD kernels are, Epanechnikov kernel

$$d_{\text{Epan}}(\boldsymbol{y}, \boldsymbol{y}') := \left(1 - \frac{\|\boldsymbol{y} - \boldsymbol{y}'\|_2^2}{\sigma}\right)^p \mathbb{1}\left(\frac{\|\boldsymbol{y} - \boldsymbol{y}'\|_2^2}{\sigma} \leq 1\right)$$

and Gaussian combination

$$d_{\text{Gauss.Comb.}}(\boldsymbol{y}, \boldsymbol{y}') := \exp\left(-\frac{\|\boldsymbol{y} - \boldsymbol{y}'\|_2^2}{\sigma_1}\right) - \exp\left(-\frac{\|\boldsymbol{y} - \boldsymbol{y}'\|_2^2}{\sigma_2}\right),$$

as shown in Ong et al. (2004). For approximating the non-CPD similarities, in Section 4.7.2, we develop more expressive inner product difference similarity (IPDS).

Before developing the IPDS, in the remaining of this section, two interesting examples of non-CPD kernels are also described by formally generalizing the set $\mathcal{Y}$ from the subset of $\mathbb{R}^{K*}$ to a set of functions.

Example 4.2: Wasserstein distance

Let $\boldsymbol{Z}$ be a metric space endowed with a metric d_Z, which we call "ground distance". For $q \geq 1$, let $\mathcal{Y}$ be the space of all measures μ on $\boldsymbol{Z}$ satisfying $\int_{\boldsymbol{Z}} d_Z(z, z_0)^q \mathrm{d}\mu(z) < \infty$ for some $z_0 \in \boldsymbol{Z}$. The q-Wasserstein

distance between $\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}$ is defined as

$$d_W^{(q)}(\boldsymbol{y}, \boldsymbol{y}') := \left(\inf_{\pi \in \Pi(\boldsymbol{y}, \boldsymbol{y}')} \iint_{\boldsymbol{Z} \times \boldsymbol{Z}} d_Z(z, z')^q \mathrm{d}\pi(z, z') \right)^{1/q}.$$

Here, $\Pi(\boldsymbol{y}, \boldsymbol{y}')$ is the set of joint probability measures on $\boldsymbol{Z} \times \boldsymbol{Z}$ having marginals $\boldsymbol{y}, \boldsymbol{y}'$. Wasserstein distance is used for a broad range of methods, such as Generative Adversarial Networks (Arjovsky, Chintala, and Bottou, 2017) and AutoEncoder (Tolstikhin et al., 2018).

Example 4.3: Jeffrey's divergence

Let $\mathcal{Y}$ be a set of distributions over a set $\boldsymbol{Z} \subset \mathbb{R}^q$. Geffrey's divergence between two distributions $\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}$ is defined via Kullback-Leibler divergence (Kullback and Leibler, 1951) by

$$d_{\text{Jeffrey's}}(\boldsymbol{y}, \boldsymbol{y}') := d_{\mathrm{KL}}(\boldsymbol{y}, \boldsymbol{y}') + d_{\mathrm{KL}}(\boldsymbol{y}', \boldsymbol{y}),$$
$$d_{\mathrm{KL}}(\boldsymbol{y}, \boldsymbol{y}') := \int_{\boldsymbol{Z}} y(z) \log \frac{y(z)}{y'(z)} \mathrm{d}z,$$

where $y(z)$ is the probability density function corresponding to the distribution $\boldsymbol{y} \in \mathcal{Y}$.
Kullback-Leibler divergence is used in Gaussian embedding (Vilnis and McCallum, 2015), that embeds data vectors into q-variate normal distribution $\mathbb{R}^p \ni \boldsymbol{x} \mapsto \boldsymbol{y} := \mathcal{N}_q(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\Sigma}(\boldsymbol{x})) \in \mathcal{Y}$, where $\boldsymbol{y}$ is also interpreted as a vector of dimension $K = q + q(q+1)/2$ by considering the number of parameters in $\boldsymbol{\mu}, \boldsymbol{\Sigma}$. Its deep NN extension is called deep Gaussian embedding (Bojchevski and Günnemann, 2018). Although these Gaussian embedding methods employ non-symmetric functions such as Kullback-Leibler divergence (Kullback and Leibler, 1951) for measuring the similarity, Jiang et al. (2019) instead considers arbitrary symmetric function. Thus KL divergence in Gaussian embedding may be replaced with the symmetric Jeffrey's divergence.

General Wasserstein distance is known not to be CPD (Gardner et al., 2017). Moreover, negative Jeffrey's divergence $-d_{\text{Jeff}}$ is not CPD on $\tilde{\mathcal{P}}_q^2$, where $\tilde{\mathcal{P}}_q$ represents the set of all q-variate normal distributions; a counterexample of CPD-ness is, $n = 3, q = 2, c_1 = -2/5, c_2 = -3/5, c_3 = 1, \boldsymbol{y}_i = \mathcal{N}_2(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \in \mathcal{Y}\ (i = 1, 2, 3), \boldsymbol{\mu}_1 = (2, 1)^\top, \boldsymbol{\mu}_2 = (-1, 1)^\top, \boldsymbol{\mu}_3 = (1, 2)^\top, \boldsymbol{\Sigma}_1 = \mathrm{diag}(1/10, 1), \boldsymbol{\Sigma}_2 = \mathrm{diag}(1/2, 1), \boldsymbol{\Sigma}_3 = \mathrm{diag}(1, 1)$.

4.7.2 IPDS

For approximating non-CPD similarities, we propose inner product difference similarity (IPDS)

$$h_{\text{IPDS}}(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}^+, \boldsymbol{\theta}^-) = \langle \boldsymbol{f}_{\text{NN}}^+(\boldsymbol{x}_i), \boldsymbol{f}_{\text{NN}}^+(\boldsymbol{x}_j) \rangle - \langle \boldsymbol{f}_{\text{NN}}^-(\boldsymbol{x}_i), \boldsymbol{f}_{\text{NN}}^-(\boldsymbol{x}_j) \rangle, \quad (4.21)$$

where $f^{\pm}_{\mathrm{NN}} : \mathcal{X} \to \mathbb{R}^{K_\pm}$ are vector-valued NNs whose parameters are $\boldsymbol{\theta}^{\pm}$. Once the IPDS model is trained, feature vectors can be obtained by concatenating these two NNs, i.e.,

$$\boldsymbol{y}_i = (f^{+}_{\mathrm{NN}}(\boldsymbol{x}_i), f^{-}_{\mathrm{NN}}(\boldsymbol{x})) \in \mathbb{R}^{K_+ + K_-} \quad (i = 1, 2, \ldots, n).$$

4.7.3 ATs for IPDS

In this section, we first prove that IPDS can approximate arbitrary similarity, i.e., symmetric and continuous function, in the sense of L^2-norm

$$\|h\|_2 := \left(\iint h(\boldsymbol{x}, \boldsymbol{x}')^2 \mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{x}' \right)^{1/2}.$$

Note that the previous ATs for IPS, SIPS and C-SIPS are in the sense of L^∞-norm $\|h\|_\infty := \sup_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}} |h(\boldsymbol{x}, \boldsymbol{x}')|$; AT for IPDS is in the weaker L^2 sense. Later, some additional conditions are described to attain the uniform convergence in the AT of IPDS.

Here, we consider a kernel $g_* : \mathcal{Y}^2 \to \mathbb{R}$ defined for a non-empty compact set $\mathcal{Y} \subset \mathbb{R}^{K_*}$ ($K_* \in \mathbb{N}$). The kernel is symmetric, and square-integrable as it is continuous over the compact set $\mathcal{Y}^2$. Then, we have a spectral decomposition

$$g_*(\boldsymbol{y}, \boldsymbol{y}') = \sum_{k=1}^{\infty} \lambda_k \varphi_k(\boldsymbol{y}) \varphi_k(\boldsymbol{y}'), \tag{4.22}$$

where $\{\lambda_k\}_{k=1}^{\infty}$ represents eigenvalues, $\{\varphi_k(\boldsymbol{y})\}_{k=1}^{\infty}$ denotes eigenfunctions, and the convergence is in the sense of L^2-norm, i.e., $(\iint g(\boldsymbol{y}, \boldsymbol{y}')^2 \mathrm{d}\boldsymbol{y}\mathrm{d}\boldsymbol{y}')^{1/2}$. The spectral decomposition theorem is also known as Hilbert-Schmidt theorem. See, e.g., Riesz and Nagy (2012) Section 97 for $\mathcal{Y} = [a, b]$ with $-\infty < a < b < \infty$; it can be straightforwardly generalized to the compact set $\mathcal{Y} \subset \mathbb{R}^{K_*}$. Furthermore, Knapp (2005) Theorem 2.5 proves the continuity of all the eigenfunctions $\{\varphi_k(\boldsymbol{y})\}_{k=1}^{\infty}$ of the continuous kernel g_*.

Eigenfunctions are orthonormal, i.e., $\int \varphi_k(\boldsymbol{y}) \varphi_l(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \mathbb{1}(k = l)$; thus

$$\int g_*(\boldsymbol{y}, \boldsymbol{y}') \varphi_k(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} = \lambda_k \varphi_k(\boldsymbol{y}') \quad \text{and}$$
$$\int g_*(\boldsymbol{y}, \boldsymbol{y}') \varphi_k(\boldsymbol{y}) \varphi_l(\boldsymbol{y}') \mathrm{d}\boldsymbol{y}\mathrm{d}\boldsymbol{y}' = \begin{cases} \lambda_k & (k = l) \\ 0 & (k \neq l) \end{cases},$$

for all $k, l \in \mathbb{N}$.

Considering the decomposition (4.22) and assuming that $\boldsymbol{y} = \boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{y}' = \boldsymbol{f}_*(\boldsymbol{x}')$, it holds for the similarity that

$$\begin{aligned} h_*(\boldsymbol{x}, \boldsymbol{x}') &:= g_*(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) \\ &= \sum_{k:\lambda_k > 0} |\lambda_k| \varphi_k(\boldsymbol{f}_*(\boldsymbol{x})) \varphi_k(\boldsymbol{f}_*(\boldsymbol{x}')) - \sum_{k:\lambda_k < 0} |\lambda_k| \varphi_k(\boldsymbol{f}_*(\boldsymbol{x})) \varphi_k(\boldsymbol{f}_*(\boldsymbol{x}')) \end{aligned}$$

$$= \lim_{K_+,K_-\to\infty} \Bigg\{ \underbrace{\sum_{k=0}^{K_+} \lambda_k^+ \varphi_k^+(f_*(\boldsymbol{x}))\varphi_k^+(f_*(\boldsymbol{x}'))}_{(\star 1)} - \underbrace{\sum_{k=0}^{K_-} \lambda_k^- \varphi_k^-(f_*(\boldsymbol{x}))\varphi_k^-(f_*(\boldsymbol{x}'))}_{(\star 2)} \Bigg\},$$

by rearranging the eigenvalues so that $\lambda_1^+ \geq \lambda_2^+ \geq \cdots \geq \lambda_{K_+}^+ \geq \cdots \geq 0 \geq \cdots \geq \lambda_{K_-}^- \geq \cdots \geq \lambda_2^- \geq \lambda_1^-$. $\varphi_k^+(\boldsymbol{y}'), \varphi_k^-(\boldsymbol{y}')$, $(k \in \mathbb{N})$ are corresponding continuous eigenfunctions, and the convergence is in the sense of L^2-norm. Then, the terms $(\star 1)$ and $(\star 2)$ also can be written as

$$(\star 1) = \langle \tilde{\boldsymbol{\Phi}}_{K_+}^+(\boldsymbol{x}), \tilde{\boldsymbol{\Phi}}_{K_+}^+(\boldsymbol{x}') \rangle, \quad (\star 2) = \langle \tilde{\boldsymbol{\Phi}}_{K_-}^-(\boldsymbol{x}), \tilde{\boldsymbol{\Phi}}_{K_-}^-(\boldsymbol{x}') \rangle,$$

where the k-th entry of $\tilde{\boldsymbol{\Phi}}_{K_\pm}^\pm : \mathcal{X} \to \mathbb{R}^{K_\pm}$ is specified by $\sqrt{\lambda_k^\pm}\varphi_k^\pm(f_*(\boldsymbol{x}))$. Approximating $\tilde{\boldsymbol{\Phi}}_{K_+}^+, \tilde{\boldsymbol{\Phi}}_{K_-}^-$ by the NNs $f_{\mathrm{NN}}^+, f_{\mathrm{NN}}^-$ immediately leads to the following Theorem 4.7.

Theorem 4.7: AT for IPDS

Symbols and assumptions are the same as those of Theorem 4.2 but g_* is any arbitrary kernel, i.e., symmetric and continuous function. We consider 1-hidden layer NNs $f_{\mathrm{NN}}^+(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{\sigma}(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}) \in \mathbb{R}^{K_+}$ and $f_{\mathrm{NN}}^-(\boldsymbol{x}) = \boldsymbol{E}\boldsymbol{\sigma}(\boldsymbol{F}\boldsymbol{x} + \boldsymbol{o}) \in \mathbb{R}^{K_-}$ with m_+ and m_- hidden units, respectively. Their parameters $\boldsymbol{\theta}^+ = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{c}) \in \boldsymbol{\Theta}^+ := \mathbb{R}^{K_+ \times m_+} \times \mathbb{R}^{m_+ \times p} \times \mathbb{R}^{m_+}$ and $\boldsymbol{\theta}^- = (\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{o}) \in \boldsymbol{\Theta}^- := \mathbb{R}^{K_- \times m_-} \times \mathbb{R}^{m_- \times p} \times \mathbb{R}^{m_-}$ are to be estimated. Then, for any $\varepsilon > 0$, there exist $K_{+0} = K_{+0}(\varepsilon), K_{-0} = K_{-0}(\varepsilon) \in \mathbb{N}$ such that

$$\inf_{\substack{\boldsymbol{\theta}^+ \in \boldsymbol{\Theta}^+ \\ \boldsymbol{\theta}^- \in \boldsymbol{\Theta}^-}} \left\| g_*\left(f_*(\boldsymbol{x}), f_*(\boldsymbol{x}')\right) - \left(\langle f_{\mathrm{NN}}^+(\boldsymbol{x}), f_{\mathrm{NN}}^+(\boldsymbol{x}') \rangle - \langle f_{\mathrm{NN}}^-(\boldsymbol{x}), f_{\mathrm{NN}}^-(\boldsymbol{x}') \rangle\right) \right\|_2 < \varepsilon$$

with sufficiently large $m_+ = m_+(K_+, \varepsilon), m_- = m_-(K_-, \varepsilon) \in \mathbb{N}$, for all $K_+ \geq K_{+0}, K_- \geq K_{-0}$.

See Appendix B.3.1 for the rigorous proof.

Although the above Theorem 4.7 asserts that the IPDS can approximate arbitrary similarity, this theorem is proved in the sense of L^2-norm. However, considering an additional assumption

- all but a finite number of the non-zero eigenvalues of g_* are of the same sign, positive or negative,

the spectral decomposition (4.22) converges uniformly. See, e.g., Riesz and Nagy (2012) Section 98. Then, the L^2-norm in Theorem 4.7 can be simply replaced by the L^∞-norm.

In the AT of IPDS, the similarity h_* is essentially decomposed into the difference between two PD similarities. We here last note that, a similar decomposition can be found in the studies in line with reproducing kernel Kreĭn space (RKKS; Mary, 2003; Ong et al., 2004; Oglic and Gaertner, 2018), that further generalizes reproducing kernel Hilbert space (RKHS; Aronszajn, 1950). Therein, non-PD kernel g_* is considered for kernel methods. If the kernel g_* is dominated by some PD kernel, namely there exists a PD kernel $g^{(\mathrm{PD})}$ such that $g^{(\mathrm{PD})} - g_*$ is also PD, the kernel g_* is associated with a RKKS, and the g_* is proved to be decomposed into two PD kernels g^+, g^- such that $g_* = g^+ - g^-$. Therefore, arbitrary similarity h_* whose kernel satisfies the assumption can be decomposed into the difference between two PD similarities h^+, h^-, namely $h_* = h^+ - h^-$; IPDS approximates these two PD similarities h^+, h^- by the two IPSs.

4.7.4 Discussion on Kernel Eigenvalues

In this section, we last revisit the classes of kernels, from the perspective of the spectral decomposition (4.22). It is widely known that all the PD kernels contain only non-negative eigenvalues. However, it is harder to characterize general CPD kernels. As a clue to characterize the CPD kernels, we herein prove that any CPD kernel contains at most one negative eigenvalue.

Considering the decomposition (4.12), any CPD kernel g_* can be decomposed as

$$g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}') = g_*^{(\mathrm{PD})}(\boldsymbol{y}, \boldsymbol{y}') + \underbrace{v_*(\boldsymbol{y}) + v_*(\boldsymbol{y}')}_{(\star)} \tag{4.23}$$

for some PD kernel $g_*^{(\mathrm{PD})}$ and some continuous function $v_* : \mathcal{Y} \to \mathbb{R}$. If v_* is a zero function, (4.23) contains no negative eigenvalue; we herein assume that v_* is not a zero function, in order to avoid the trivial case.

By specifying

$$r := \int v_*(\boldsymbol{y}) \mathrm{d}\boldsymbol{y},\ \|v_*\|_2 := \left(\int v_*(\boldsymbol{y})^2 \mathrm{d}\boldsymbol{y} \right)^{1/2},\ \text{and}\ \xi := \sqrt{r^2 + \|v_*\|_2^2},$$

the eigenvalues of $(\star)$ are

$$\lambda_1 := r + \xi \ \text{ and } \ \lambda_2 := r - \xi.$$

As $\xi > |r|$, $(\star)$ is proved to have one negative eigenvalue. Namely, $(\star) = \psi(\boldsymbol{y})\psi(\boldsymbol{y}') - \rho(\boldsymbol{y})\rho(\boldsymbol{y}')$ for some (non-zero) functions $\psi, \rho : \mathcal{Y} \to \mathbb{R}$; we obtain another form of the CPD kernel

$$g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}') = \tilde{g}(\boldsymbol{y}, \boldsymbol{y}') - \rho(\boldsymbol{y})\rho(\boldsymbol{y}') \tag{4.24}$$

with a PD kernel $\tilde{g}(\boldsymbol{y}, \boldsymbol{y}') := g_*^{(\mathrm{PD})}(\boldsymbol{y}, \boldsymbol{y}') + \psi(\boldsymbol{y})\psi(\boldsymbol{y}')$. We last prove that the function (4.24) contains at most one negative eigenvalue.

Assuming that $g_*^{(\mathrm{CPD})}$ has more than one negative eigenvalues, we may consider two eigenfunctions $\varphi_1, \varphi_2 : \mathcal{Y} \to \mathbb{R}$ of $g_*^{(\mathrm{CPD})}$, where the corresponding eigenvalues are $\lambda_1, \lambda_2 < 0$. We define a set

$$\mathcal{S} := \mathrm{span}\{\varphi_1, \varphi_2\} \cap \{\rho\}^\perp$$

where $\mathrm{span}\{\varphi_1, \varphi_2\} := \{\alpha_1\varphi_1 + \alpha_2\varphi_2 \mid \alpha_1, \alpha_2 \in \mathbb{R}\}$ and $\{\rho\}^\perp := \{\eta : \mathcal{Y} \to \mathbb{R} \mid \int \eta(\boldsymbol{y})\rho(\boldsymbol{y})\mathrm{d}\boldsymbol{y} = 0\}$. Then, there exists a non-zero function $\tau \in \mathcal{S}$ as the codimension of $\{\rho\}^\perp$ is 1 and the dimension of $\mathrm{span}\{\varphi_1, \varphi_2\}$ is 2. Such a function $\tau \in \mathcal{S}$ leads to the following inequalities

$$\begin{aligned}
&\iint g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}')\tau(\boldsymbol{y})\tau(\boldsymbol{y}')\mathrm{d}\boldsymbol{y}\mathrm{d}\boldsymbol{y}' < 0 \quad (\because \tau \in \mathrm{span}\{\varphi_1, \varphi_2\}) \quad \text{and} \\
&\iint g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}')\tau(\boldsymbol{y})\tau(\boldsymbol{y}')\mathrm{d}\boldsymbol{y}\mathrm{d}\boldsymbol{y}' \\
&\qquad = \iint \tilde{g}(\boldsymbol{y}, \boldsymbol{y}')\tau(\boldsymbol{y})\tau(\boldsymbol{y}')\mathrm{d}\boldsymbol{y}\mathrm{d}\boldsymbol{y}' - \Bigg(\underbrace{\int \rho(\boldsymbol{y})\tau(\boldsymbol{y})\mathrm{d}\boldsymbol{y}}_{=0\,(\because \tau \in \{\rho\}^\perp)}\Bigg)^2 \geq 0.
\end{aligned}$$

These contradicting inequalities imply that the assumption cannot be true. Therefore, the CPD kernel $g_*^{(\mathrm{CPD})}$ has no more than one negative eigenvalue. □

Note that the converse is not always true; a kernel g_* that contains only one negative eigenvalue is not always CPD. One simple counterexample is $g_*(y, y') = -yy'$ for $\mathcal{Y} = [-\sqrt{3}, \sqrt{3}]$, where the eigenvalue and eigenfunction are -1 and y, respectively: g_* is not CPD as $\sum_{i,j=1}^2 c_i c_j g_*(y_i, y_j) = -1 < 0$ for $c_1 = 1, c_2 = -1, y_1 = 1, y_2 = 0$.

4.7.5 Further Development

IPDS is harder to optimize than IPS or SIPS without a proper regularization, because the expression of IPDS is redundant. For instance, considering NNs $\tilde{f}_{\mathrm{NN}}^+(\boldsymbol{x}) := (f_{\mathrm{NN}}^+(\boldsymbol{x}), \gamma), \tilde{f}_{\mathrm{NN}}^-(\boldsymbol{x}) := (f_{\mathrm{NN}}^-(\boldsymbol{x}), \gamma)$, it obviously holds for any $\gamma \in \mathbb{R}$ that

$$\begin{aligned}
&\langle \tilde{f}_{\mathrm{NN}}^+(\boldsymbol{x}), \tilde{f}_{\mathrm{NN}}^+(\boldsymbol{x}')\rangle - \langle \tilde{f}_{\mathrm{NN}}^-(\boldsymbol{x}), \tilde{f}_{\mathrm{NN}}^-(\boldsymbol{x}')\rangle \\
&\qquad = \langle f_{\mathrm{NN}}^+(\boldsymbol{x}), f_{\mathrm{NN}}^+(\boldsymbol{x}')\rangle - \langle f_{\mathrm{NN}}^-(\boldsymbol{x}), f_{\mathrm{NN}}^-(\boldsymbol{x}')\rangle.
\end{aligned}$$

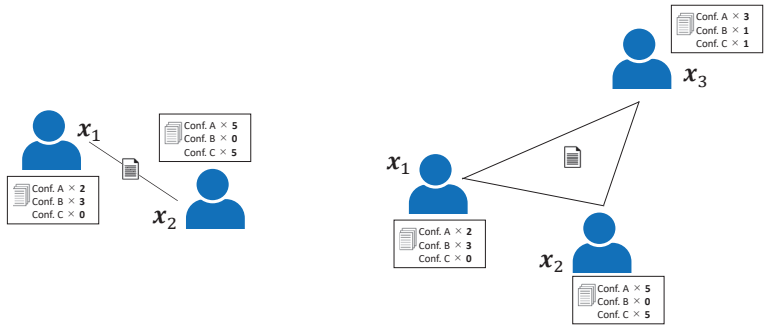
For solving the issue, weighted-IPS (WIPS) is presented and evaluated by extensive numerical experiments in a successive paper of ours (Kim et al., 2019).

4.8 Conclusion

We studied approximation capability of IPS, and developed more expressive similarities called shifted-IPS (SIPS) and constantly-SIPS (C-SIPS). We proved SIPS and C-SIPS can approximate CPD similarities whereas IPS is limited to PD similarities. We also evaluated the approximation error rates of IPS, SIPS and C-SIPS, and these similarities were performed on our numerical experiments. Furthermore, we also provided more expressive similarities called inner-product difference similarity (IPDS), that is capable of approximating arbitrary similarities.

5 Hyperlink Regression via Bregman Divergence

This chapter discusses the hyper-relational extension of link regression called hyperlink regression (HLR), which is shown in Figure 5.1.



(A) $U = 2$ (B) $U = 3$

FIGURE 5.1: Whereas (A) existing graph embedding considers the link weight w_{i_1,i_2} defined between a pair of $U = 2$ data vectors $(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2})$, (B) this chapter considers the hyperlink weight $w_{\boldsymbol{i}}$ defined for a set of more than 2 data vectors $\boldsymbol{X}_{\boldsymbol{i}} = (\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U})$ called tuple. In co-authorship network example, the hyperlink weight represents the number of co-authored papers written by $U \in \mathbb{N}$ researchers belonging to the tuple.

Although one of the main points in this chapter is hyper-relational learning called HLR, where we propose HLR using Bregman divergence called Bregman HLR (BHLR), this chapter also proves statistical estimation consistency for general BHLR, and provides a general and theoretically guaranteed stochastic optimization algorithm. The estimation consistency can be applied to general BHLR, which includes a variety of existing methods, such as graph embedding equipped with even the expressive SIPS and IPDS discussed in the previous Chapter 4.

Our contributions, descriptions, and materials including figures in this chapter are mostly attributed to a published journal paper of ours (Okuno and Shimodaira, 2020) and partially attributed to another conference paper of ours (Okuno and Shimodaira, 2019).

5.1 Introduction

Thus far, we have developed a multi-view extension of graph embedding (GE) in Chapter 3, and more expressive similarities for GE in Chapter 4. Along with the development of highly expressive GEs, estimation consistency, that relies on the loss function used for learning GE, also plays an important role in obtaining better performance.

For instance, LINE (Tang et al., 2015) leverages logistic loss, and PMvGE (Okuno, Hada, and Shimodaira, 2018) discussed in Chapter 3 and Okuno, Kim, and Shimodaira (2019) discussed in Chapter 4 leverage the loss function based on Poisson distribution, where maximizing the likelihood of link weights is equivalent to minimizing Kullback–Leibler (KL) divergence between link weights and the estimated similarities. In fact, the logistic loss and the KL divergence can be easily proved to exhibit the following two desirable properties:

- **(P-1) statistical consistency**, that is, it asymptotically recovers the underlying true conditional expectation of link weights given data vectors, and
- **(P-2) computational tractability**, that is, it can be computed efficiently by stochastic algorithms using a minibatch sampling for relational data.

However, it is unclear whether the properties (P-1) and (P-2) above hold for the limited function class, or if they hold for some larger function classes. Because only the limited loss functions are theoretically proven to exhibit such favorable properties, the present circumstance may limit the choice of the loss function and may result in a missed opportunity to improve the GE's performance.

In addition to the problem of specifying loss functions, existing GEs are limited to considering the link weight defined between only two nodes, despite the fact that link weights can be similarly defined for a set of three or more nodes. We call the weight defined for three or more nodes as *hyperlink weight*. A hyperlink weight appears in many practical situations; in a friend network, the existence of a group to which all the selected $U(\geq 2)$ people belong should be expressed as a binary hyperlink weight. Similarly, the number of co-authored papers written by all the selected $U(\geq 2)$ people in a co-authorship network should be represented as hyperlink weights assuming values in non-negative integers. The existing link regression, including metric learning and GE, cannot address such complicated hyperlink weights.

For simultaneously solving these two challenges, we propose Bregman hyperlink regression (BHLR) by (i) extending link regression to hyperlink regression (HLR) such that it predicts the hyperlink weight defined for a collection of $U(\in \mathbb{N})$ vectors called U-tuple, and (ii) employing the Bregman divergence (BD) that includes many loss functions such as logistic loss, KL divergence, and β-divergence as special cases. BHLR is a general framework for hyper-relational learning, that encompasses various existing methods;

BHLR is, in general, demonstrated to possess the two desirable properties, (P-1) statistical consistency and (P-2) computational tractability.

Our contribution in this chapter is summarized as follows.

1. In Section 5.3, we propose BHLR, that is a simple and general framework for hyper-relational learning. BHLR predicts hyperlink weight $w_{i_1,i_2,\ldots,i_U} \in \mathcal{S}\ (\subset \mathbb{R})$ from the corresponding tuple of data vectors $\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U} \in \mathcal{X}\ (\subset \mathbb{R}^p)$ through a user-specified symmetric similarity function $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U})$; highly expressive nonlinear functions can be employed for the similarity function.

2. In Section 5.4, we demonstrate that BHLR encompasses various existing methods, including the methods for representation learning, such as graph embedding ($U = 2$), matrix factorization ($U = 2$), tensor factorization ($U \geq 2$), and their variants equipped with arbitrary BD; obtained feature vectors through the representation learning methods can be used for a variety of downstream tasks (e.g., clustering and visualization) besides just predicting hyperlink weights.

3. In Section 5.5, we prove that general BHLR possesses the desirable properties, (P-1) statistical consistency and (P-2) computational tractability. Regarding (P-2), we consider a minibatch-based stochastic algorithm; we propose a novel generalized mini-batch sampling procedure for hyper-relations, for dealing with non-negligible computational complexity $O(n^U)$. Consequently, BHLR including several existing methods, that have been examined experimentally, is theoretically justified in a unified manner.

5.2 Preliminaries

In this section, we first define Bregman divergence in Section 5.2.1, and subsequently, we describe the problem setting in Section 5.2.2. In Section 5.2.3, we formally define conditional probabilities of hyperlink weights.

5.2.1 Bregman Divergence

In this section, we introduce Bregman divergence (BD) for formulating the Bregman hyperlink regression (BHLR) later in Section 5.3. By virtue of the nature of the BD, the proposed BHLR is also expected to be robust estimation; the intuition of the robustness is also discussed in Section 5.3.3.

Here, we consider an index set $\mathcal{I}$, which is specifically defined as the set of tuple indices in our problem setting explained in Section 5.2.2. With a continuously differentiable and strictly convex *generating function* $\varphi : \mathrm{dom}(\varphi) \to \mathbb{R}$ whose domain is a set $\mathrm{dom}(\varphi) \subset \mathbb{R}$, the BD (Bregman, 1967; Censor, Zenios,

et al., 1997) between $\boldsymbol{a} := \{a_{\boldsymbol{i}} \in \mathrm{dom}(\varphi) \mid \boldsymbol{i} \in \mathcal{I}\}$ and $\boldsymbol{b} := \{b_{\boldsymbol{i}} \in \mathrm{dom}(\varphi) \mid \boldsymbol{i} \in \mathcal{I}\}$ is defined by

$$D_\varphi(\boldsymbol{a}, \boldsymbol{b}) := \frac{1}{|\mathcal{I}|} \sum_{\boldsymbol{i} \in \mathcal{I}} d_\varphi(a_{\boldsymbol{i}}, b_{\boldsymbol{i}}), \tag{5.1}$$

where $d_\varphi : \mathrm{dom}(\varphi)^2 \to \mathbb{R}$ indicates the difference between $\varphi(a)$ and the first-order Taylor approximation of $\varphi(a)$ around $b \in \mathrm{dom}(\varphi)$ as

$$d_\varphi(a, b) := \varphi(a) - (\varphi(b) + \varphi'(b)(a - b)), \quad (a, b \in \mathrm{dom}(\varphi)).$$

Because φ is strictly convex, $d_\varphi(a, b)$ is always non-negative, and attains the minimum value 0 at $b = a$ for any fixed $a \in \mathrm{dom}(\varphi)$. Similarly, $D_\varphi(\boldsymbol{a}, \boldsymbol{b}) \geq 0$ $(\forall \boldsymbol{a}, \boldsymbol{b} \in \mathrm{dom}(\varphi)^{|\mathcal{I}|})$, and the equality holds if and only if $\boldsymbol{a} = \boldsymbol{b}$ (basic property 2 in (Cichocki et al., 2009) p.101). Thus, for any fixed $\boldsymbol{a} \in \mathrm{dom}(\varphi)^{|\mathcal{I}|}$, minimizing $D_\varphi(\boldsymbol{a}, \boldsymbol{b})$ with respect to $\boldsymbol{b} \in \mathrm{dom}(\varphi)^{|\mathcal{I}|}$ is expected to cause $\boldsymbol{b}$ to be closer to $\boldsymbol{a} \in \mathrm{dom}(\varphi)^{|\mathcal{I}|}$. In our proposed BHLR, $\boldsymbol{a}, \boldsymbol{b}$ are specifically defined as observed hyperlink weights and their predicted weights, respectively, as explained in Section 5.3.2; the predicted weights are expected to be closer to the observed weights, due to the BD's property.

Some of the BD family members such as the KL divergence are originally defined for measuring the difference between two probability distributions. That is, they assume that $\boldsymbol{a}, \boldsymbol{b}$ satisfy (1) $a_{\boldsymbol{i}}, b_{\boldsymbol{i}} \geq 0$ $(\forall \boldsymbol{i} \in \mathcal{I})$, and (2) $\sum_{\boldsymbol{i} \in \mathcal{I}} a_{\boldsymbol{i}} = \sum_{\boldsymbol{i} \in \mathcal{I}} b_{\boldsymbol{i}} = 1$. However, assumptions (1) and (2) are in fact not required for the BD to hold the favorable property above. Thus, we do not assume (1) and (2) hereinafter, similarly to some existing studies (Cichocki et al., 2009; Banerjee et al., 2005; Sra and Dhillon, 2006).

The BD includes a variety of loss functions such as the KL divergence, β-divergence, quadratic loss, and logistic loss, as shown in Table 5.1.

By removing the strict convexity assumption on φ and additionally assuming $a \in \{0, 1\}$, the BD includes margin-based loss functions. For instance, $\varphi(x) = \max\{-x, x - 1\}$ results in the misclassification loss $d_\varphi(a, b) = I(a \neq I(b > 1/2))$, where $I(\cdot)$ represents the indicator function; other examples can be found in Zhang, Jiang, and Shang (2009) Section 6.2.

5.2.2 Problem Setting

In previous Chapter 3 and 4, we discussed graph embedding that predicts link weight w_{ij} by a function $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ of data vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$. We hereby extend the problem to hyper-relation setting, i.e., hyperlink weight $w_{\boldsymbol{i}}$ for $\boldsymbol{i} = (i_1, i_2, \ldots, i_U)$ is predicted from a tuple of data vectors $\boldsymbol{X}_{\boldsymbol{i}} = (\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U})$; the extension reduces to the existing graph embedding by specifying $U = 2$.

For fixed $p, n, U \in \mathbb{N}$ and non-empty sets $\mathcal{X} \subset \mathbb{R}^p, \mathcal{S} \subset \mathbb{R}$, our dataset comprises p-dimensional data vectors $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathcal{X}$ and symmetric hyperlink weights $\{w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \subset \mathcal{S}$, where $\boldsymbol{i} = (i_1, i_2, \ldots, i_U)$ is a multiple index in a set

TABLE 5.1: Bregman divergence family. See, e.g. Cichocki et al. (2009) Section 2.4 and Banerjee et al. (2005) Table 1 for details.

$\varphi(x)$	$\mathrm{dom}(\varphi)$	$d_\varphi(a,b)$	Name of $D_\varphi(\boldsymbol{a},\boldsymbol{b})$
$x\log x+(1-x)\log(1-x)$	$[0,1]$	$\begin{array}{l}-a\log b-(1-a)\log(1-b)\\ +a\log a+(1-a)\log(1-a)\end{array}$	Logistic loss† (Banerjee et al., 2005)
$x\log x-x$	$\mathbb{R}_{\geq 0}$	$a\log\frac{a}{b}-(a-b)$	Kullback–Leibler div. (Cichocki et al., 2009)
$\frac{x^{1+\beta}}{\beta(1+\beta)}-\frac{x}{\beta}$	$\mathbb{R}_{\geq 0}$	$\frac{a^{1+\beta}}{\beta(1+\beta)}-\frac{ab^\beta}{\beta}+\frac{b^{1+\beta}}{1+\beta}$	β-div.‡ (Basu et al., 1998)
$-\log x$	$\mathbb{R}_{>0}$	$\frac{a}{b}-\log\frac{a}{b}-1$	Itakura-Saito div. (Cichocki et al., 2009)
$\frac{1}{x}$	$\mathbb{R}_{>0}$	$\frac{(a-b)^2}{ab^2}$	Inverse div. (Cichocki et al., 2009)
$\frac{x^2-x}{2}$	$\mathbb{R}$	$\frac{1}{2}(a-b)^2$	Quadratic loss (Cichocki et al., 2009)
$\exp(x)$	$\mathbb{R}$	$\exp(a)-(a-b+1)\exp(b)$	Exponential div. (Cichocki et al., 2009)
$\log(1+\exp(x))$	$\mathbb{R}$	$\log\frac{1+\exp(a)}{1+\exp(b)}-(a-b)\frac{\exp(b)}{1+\exp(b)}$	Dual logistic loss (Boissonnat, Nielsen, and Nock, 2010)

†By specifying $a\in\{0,1\}$ and $0\cdot\log 0=0$, the logistic loss reduces to $-a\log b-(1-a)\log(1-b)$.
$^\ddagger\beta>0$ is a user-specified parameter. β-div. generalizes the Kullback–Leibler div. ($\beta\downarrow 0$) and quadratic loss ($\beta=1$).

$\mathcal{I}_n^{(U)} \subset [n]^U$, and $[n]$ represents the set $\{1, 2, \ldots, n\}$. Formal descriptions for a tuple of data vectors, hyperlink weights, and the index set are provided in the following.

- **U-tuple** $\boldsymbol{X} = (\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}'', \ldots) \in \mathcal{X}^U$ is an array of U vectors, where $\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}'', \ldots \in \mathcal{X}\ (\subset \mathbb{R}^p)$ are p-dimensional vectors. For an index $\boldsymbol{i} = (i_1, i_2, \ldots, i_U) \in \mathcal{I}_n^{(U)}\ (\subset [n]^U)$, a collection of U data vectors $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_U} \in \mathcal{X}$ constitute U-tuple $\boldsymbol{X}_{\boldsymbol{i}} = (\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_U})$ indexed by $\boldsymbol{i}$. Although the order of the vectors is provided, it is in effect ignored in the proposed method, by considering only the symmetric function for the tuple. Note that two different tuples may share same data vectors. For instance, $\boldsymbol{X}_{(1,2,3)} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ and $\boldsymbol{X}_{(1,3,4)} = (\boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_4)$ share two data vectors $\boldsymbol{x}_1, \boldsymbol{x}_3$; we use the multiple index $\boldsymbol{i} \in \mathcal{I}_n^{(U)}$ for dealing with the duplicate data vectors that appear in several different tuples.

- **Hyperlink weight** $w_{\boldsymbol{i}} \in \mathcal{S}\ (\subset \mathbb{R})$ represents the strength of association defined for the U-tuple $\boldsymbol{X}_{\boldsymbol{i}}$. Hyperlink is also called hyperedge in hypergraph theory, and is assumed to be symmetric with respect to permutation of the entries $i_1, i_2, \ldots, i_U$ in the index $\boldsymbol{i}$. Although we practically consider non-negative hyperlink weights in many cases, i.e., $\mathcal{S} := \mathbb{R}_{\geq 0}$ such that the weight taking value 0 represents no association among the tuple, $\mathcal{S}$ is not restricted to be non-negative; $\mathcal{S}$ can be arbitrary specified depending on the setting.

- **Index set** $\mathcal{I}_n^{(U)} \subset [n]^U$ is typically defined as $\mathcal{I}_n^{(U)} = [n]^U$, or $\mathcal{I}_n^{(U)} = \{\boldsymbol{i} \in [n]^U \mid u \neq u' \Rightarrow i_u \neq i_{u'}\}$ such that any tuple do not contain any duplicate vectors in itself, though different tuples may share some data vectors. A particular set $\mathcal{I}_n^{(U)} = \mathcal{J}_n^{(U)} := \{\boldsymbol{i} \in [n]^U \mid 1 \leq i_1 < i_2 < \cdots < i_U\}$ is employed later in Section 5.5.1, for showing asymptotic properties of the proposed method. Although the examples of $\mathcal{I}_n^{(U)}$ mentioned above basically cover all the combinations of indices under some constraints, we can think of even a subset of them for $\mathcal{I}_n^{(U)}$ in order to allow the practical situation that a limited number of hyperlink weights are actually observed.

Such hyperlink weights defined for U-tuples appear in many practical situations. Two different types of hyperlink weights for $\mathcal{S} := \mathbb{R}_{\geq 0}$ are shown in the following Examples 5.2.2 and 5.2.2. They are also referred to as a hypernetwork (Jeffrey, 2013).

Example 5.1: Hyperlink weights in friend network

Data vector $\boldsymbol{x}_i$ represents the property of person $i \in [n]$, e.g., age, gender, education, etc., and the hyperlink weight $w_{\boldsymbol{i}} \in \{0, 1, 2, \ldots\}(\subset \mathcal{S})$ represents the number of social groups to which all the U people indexed by $\boldsymbol{i} = (i_1, i_2, \ldots, i_U)$ belong.

Example 5.2: Hyperlink weights in co-authorship network

Data vector $\boldsymbol{x}_i$ represents the attributes of researcher $i \in [n]$ such as number of publications in each journal, and the hyperlink weight $w_{\boldsymbol{i}} \in \{0, 1, 2, \ldots\} (\subset \mathcal{S})$ represents the number of co-authored papers written by all the U researchers indexed by $\boldsymbol{i} = (i_1, i_2, \ldots, i_U)$.

Here, we consider a user-specified parametric model of *similarity function* $\mu_{\boldsymbol{\theta}} : \mathcal{X}^U \to \mathcal{S}$ with parameter vector $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^q$. For U-tuple $\boldsymbol{X} = (\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}'', \ldots) \in \mathcal{X}^U$, we consider a random variable $w \in \mathcal{S}$ with conditional expectation $\mu_*(\boldsymbol{X}) := \mathbb{E}(w \mid \boldsymbol{X})$. w and $\boldsymbol{X}$ are linked by a conditional probability mass (or density) function q, as will be formally described in the following Section 5.2.3. Then, learning the similarity function $\mu_{\boldsymbol{\theta}}$ so that

$$\mu_{\boldsymbol{\theta}}(\boldsymbol{X}) \approx \mu_*(\boldsymbol{X}), \quad \boldsymbol{X} \in \mathcal{X}^U$$

is called hyperlink regression (HLR); this is analogous to the ordinary regression analysis, where w and $\boldsymbol{X}$ correspond to the response and explanatory variables, respectively.

Given our dataset consisting of data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ and hyperlink weights $\{w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}}$, the parameter vector $\boldsymbol{\theta}$ is optimized by minimizing an empirical loss function so that

$$w_{\boldsymbol{i}} \approx \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}), \quad \boldsymbol{i} \in \mathcal{I}_n^{(U)}$$

hold, similarly to graph embedding using the probabilistic model (4.1).

Although this chapter aims at providing a general framework for HLR named BHLR, such that it encompasses a variety of existing methods, this chapter also intends to provide theoretical guarantees for general BHLR; several existing methods, that have been examined experimentally, are also theoretically justified in a unified manner, by leveraging the BHLR framework.

5.2.3 Probability Distributions of Hyperlink Weights

In order to obtain the conditional expectation $\mu_*(\boldsymbol{X}_{\boldsymbol{i}}) = \mathbb{E}(w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}})$, we first formally define the conditional distribution of hyperlink weights given data vectors, by straightforwardly generalizing the probabilistic model (4.1) considered in the previous Chapters 3 and 4.

Here, we explain why the extra attention is required for defining the conditional distribution of hyperlink weights given data vectors. For any $\boldsymbol{i}'$ obtained by permutating the elements of $\boldsymbol{i}$, tuples $\boldsymbol{X}_{\boldsymbol{i}}, \boldsymbol{X}_{\boldsymbol{i}'}$ consist of the same vectors $\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U}$, and it holds that $w_{\boldsymbol{i}} = w_{\boldsymbol{i}'}$ since the hyperlink weights are assumed to be symmetric. In the case of $U = 2$, this symmetry coincides with considering undirected links; link weights should satisfy $w_{i_1 i_2} = w_{i_2 i_1}$ for all i_1 and i_2, implying the constraints on the distributions for w_{i_1, i_2} and $w_{i_2 i_1}$.

For specifying the distribution appropriately, we employ a simple idea. We first specify the conditional probability density function (cpdf) or conditional probability mass function (cpmf) $\tilde{q}$ only for $w_{i_1 i_2} \mid \boldsymbol{X}_{i_1 i_2}$ whose index

is in non-decreasing order $i_1 \leq i_2$. Then, the cpdf or cpmf q of $w_{i_2 i_1} \mid \boldsymbol{X}_{i_2 i_1}$ whose index is in reverse order, can be defined as that of $w_{i_1 i_2} \mid \boldsymbol{X}_{i_1 i_2}$, since the weights satisfy the symmetry $w_{i_1 i_2} = w_{i_2 i_1}$ and both tuples $\boldsymbol{X}_{i_1,i_2}, \boldsymbol{X}_{i_2 i_1}$ consist of the same vectors $\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}$. This idea of symmetry is readily generalized to $U \in \mathbb{N}$; we specify the cpdf or cpmf $\tilde{q}$ of $w_{\boldsymbol{i}'} \mid \boldsymbol{X}_{\boldsymbol{i}'}$ only for non-decreasing order index $\boldsymbol{i}' \in [n]^U$ such that $i'_1 \leq i'_2 \leq \cdots \leq i'_U$, and consider a mapping $r : \boldsymbol{i} \mapsto \boldsymbol{i}'$ such that $\boldsymbol{i}' = r(\boldsymbol{i})$ is obtained by sorting the elements of $\boldsymbol{i}$ in non-decreasing order. Then cpdf or cpmf q of $w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}}$ is defined as

$$q(w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}}) := \tilde{q}(w_{r(\boldsymbol{i})} \mid \boldsymbol{X}_{r(\boldsymbol{i})}), \quad (\boldsymbol{i} \in \mathcal{I}_n^{(U)}). \tag{5.2}$$

Therefore, we have well-defined conditional distribution for hyperlink weights. Then, the cpdf (or cpmf) of all the hyperlink weights $\{w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}}$ given data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathcal{X}^n$ is

$$\prod_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} q(w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}}), \tag{5.3}$$

meaning that hyperlink weight $w_{\boldsymbol{i}}$ is conditionally independently generated by following the probabilistic model (5.3).

Hyperlink weights $w_{\boldsymbol{i}}, w_{\boldsymbol{i}'}$ are independent, even if the corresponding tuples $\boldsymbol{X}_{\boldsymbol{i}}, \boldsymbol{X}_{\boldsymbol{i}'}$ share some data vectors $\boldsymbol{x}_{i_{j_1}}, \boldsymbol{x}_{i_{j_2}}, \ldots$ therein. However, their probability density functions are related via the shared data vectors; similar probabilistic models can be found in the case that $U = 2$. For $U = 2$, $q(w \mid \boldsymbol{X}) = \mu_*(\boldsymbol{X})^w (1 - \mu_*(\boldsymbol{X}))^{1-w}$ represents the cpmf of Bernoulli distribution whose expectation is $\mu_*(\boldsymbol{X}) := \mathbb{E}(w \mid \boldsymbol{X})$, and $\boldsymbol{X} = (\boldsymbol{x}, \boldsymbol{x}')$ is a pair of latent variables, the probabilistic model (5.3) is also known as latent position random graph (LPRG) model with kernel μ_*. LPRG model is considered in Tang, Sussman, and Priebe (2013) and Athreya et al. (2018) Definition 6, and it is originated from the random dot product graph model (Young and Scheinerman, 2007), that corresponds to a case $\mu_*(\boldsymbol{X}) := \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$ for $\boldsymbol{X} = (\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{X}^2$. Our probabilistic model (5.3) generalizes the LPRG model to arbitrary probability distribution with arbitrary $U \in \mathbb{N}$, though the previous studies focus on the spectral analyses on the matrix $\boldsymbol{W} = (w_{ij})$ of Bernoulli link weights with $U = 2$, and they assume that $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ are latent variables.

Hereinafter, we note the probability distribution of the tuple $\boldsymbol{X}_{\boldsymbol{i}}$. We will simply assume that the data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ are i.i.d. randomly generated from a distribution q_X in Section 5.5 for showing statistical consistency of BHLR. Then, the joint distribution over all the hyperlink weights and data vectors is specified as $\prod_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} q(w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}}) \prod_{i=1}^n q_X(\boldsymbol{x}_i)$. Note that the marginal distribution for $\boldsymbol{Z}_{\boldsymbol{i}} := (w_{\boldsymbol{i}}, \boldsymbol{X}_{\boldsymbol{i}})$ does not depend on the index $\boldsymbol{i}$, thus $\boldsymbol{Z}_{\boldsymbol{i}}, \boldsymbol{i} \in \mathcal{I}_n^{(U)}$ are identically distributed. However, even if data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ are i.i.d. generated, two different $\boldsymbol{Z}_{\boldsymbol{i}}, \boldsymbol{Z}_{\boldsymbol{i}'}$ can be dependent, as their tuples $\boldsymbol{X}_{\boldsymbol{i}}, \boldsymbol{X}_{\boldsymbol{i}'}$ may share same data vectors in common. For instance, $\boldsymbol{X}_{(1,2,3)} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ and $\boldsymbol{X}_{(1,3,4)} = (\boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_4)$ share two data vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$. Therefore, $\boldsymbol{Z}_{\boldsymbol{i}}, \boldsymbol{i} \in \mathcal{I}_n^{(U)}$ are NOT independently distributed. This property

for $U \geq 2$ makes our setting interesting and needs special care in the asymptotic theory. In this regard, theories for HLR, that predicts hyperlink weights from the constrained tuples, can be different from those of classical regression, that typically predicts response variables from i.i.d. data vectors. We consider such constrained tuples, and the statistical consistency for BHLR is proved later in Section 5.5.

5.3 Bregman Hyperlink Regression (BHLR)

In this section, we first discuss two different approaches to HLR in Section 5.3.1, and subsequently, we propose Bregman hyperlink regression (BHLR) in Section 5.3.2. By virtue of the nature of the Bregman divergence, the proposed BHLR is expected to be robust estimation in line with β-divergence (Basu et al., 1998), which is often employed in robust statistics; the intuition of the robustness is also discussed in Section 5.3.3.

5.3.1 Two Different Approaches to HLR

As hyperlink regression (HLR) is introduced in problem setting in Section 5.2.2, in this section, we show two different approaches to HLR with $\mathcal{S} := \mathbb{R}_{\geq 0}$, and explain why we employ the second approach. Although the case of $U = 1$ is illustrated here, it can be easily generalized to arbitrary $U \in \mathbb{N}$.

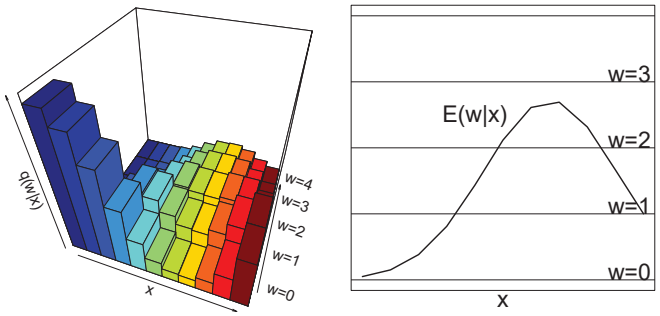
Considering a weight w_i taking a value in the set $\{0, 1, 2, \ldots\} \subset \mathcal{S}$ and a data vector $\boldsymbol{x}_i \in \mathbb{R}^p$ $(i = 1, 2, \ldots, n)$, HLR predicts the weight $w_i \in \mathcal{S}$ through the function $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \in \mathcal{S}$. However, there are two different approaches to this problem. The first approach is based on matching conditional probability mass function (pmf) $q(w_i \mid \boldsymbol{x}_i)$ and the parametric generative model $p_{\boldsymbol{\theta}}(w_i \mid \boldsymbol{x}_i)$ whose expectation is $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \sum_{w \in \mathbb{N}_0} w p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i)$. Although this approach naturally extends the maximum likelihood regression, there remain several challenges explained below. For solving these challenges, we also consider the second approach, that instead matches only the conditional expectation function $\mu_*(\boldsymbol{x}_i) := E(w_i \mid \boldsymbol{x}_i)$ and the model $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$. Consequently, we employ and generalize the second approach, and propose *Bregman-HLR (BHLR)* in Section 5.3.2.

Hereinafter, we describe the details of the two approaches to HLR.

The first approach is, matching the underlying conditional pmf $q(w_i \mid \boldsymbol{x}_i)$ and the parametric generative model $p_{\boldsymbol{\theta}}(w_i \mid \boldsymbol{x}_i)$. Let $q_{iw} = q(w \mid \boldsymbol{x}_i)$ and $p_{\boldsymbol{\theta},iw} = p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i)$ for $w \in \mathbb{N}_0$, $i = 1, \ldots, n$. They are put together as vectors $\boldsymbol{q}_i := (q_{i0}, q_{i1}, q_{i2}, \ldots)$, $\boldsymbol{p}_{\boldsymbol{\theta},i} := (p_{\boldsymbol{\theta},i0}, p_{\boldsymbol{\theta},i1}, p_{\boldsymbol{\theta},i2}, \ldots)$, so that each of vectors $\boldsymbol{q}_i, \boldsymbol{p}_{\boldsymbol{\theta},i}$ represents the distribution of $w_i \mid \boldsymbol{x}_i$. Then, we may estimate $\boldsymbol{\theta}$ by minimizing

$$\frac{1}{n} \sum_{i=1}^{n} D_\varphi(\boldsymbol{q}_i, \boldsymbol{p}_{\boldsymbol{\theta},i}), \tag{5.4}$$

where φ is a user-specified generating function. However, the underlying conditional distributions $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n$ used in (5.4) cannot be observed in



(A) $q(w \mid \boldsymbol{x})$ (B) $E(w \mid \boldsymbol{x})$

FIGURE 5.2: Examples of (A) underlying conditional probability mass function $q(w \mid \boldsymbol{x})$ whose conditional expectation is $E(w \mid \boldsymbol{x}) = \sum_{w \in \mathbb{N}_0} w q(w \mid \boldsymbol{x})$, and (B) the conditional expectation function $\mu_*(\boldsymbol{x}) = E(w \mid \boldsymbol{x})$.

practice; we instead consider the empirical conditional distribution $\hat{\boldsymbol{q}}_i = (\hat{q}_{i0}, \hat{q}_{i1}, \hat{q}_{i2}, \ldots)$ whose w_i-th entry is 1 and 0 otherwise, for $i = 1, 2, \ldots, n$. Considering $\mathcal{I} = \mathbb{N}$,

$$
\begin{aligned}
|\mathcal{I}| D_\varphi(\hat{\boldsymbol{q}}_i, \boldsymbol{p}_{\boldsymbol{\theta},i}) &= \sum_{w \in \mathbb{N}} d_\varphi(\hat{q}_{iw}, p_{\theta,iw}) \\
&= \sum_{w \in \mathbb{N}} \{\varphi(\hat{q}_{iw}) - \varphi(p_{\theta,iw}) - \varphi'(p_{\theta,iw})(\hat{q}_{iw} - p_{\theta,iw})\} \\
&= \sum_{w \in \mathbb{N}} \{\varphi'(p_{\theta,iw}) p_{\theta,iw} - \varphi(p_{\theta,iw}) - \varphi'(p_{\theta,iw}) \hat{q}_{iw}\} + \text{Const.} \\
&= \sum_{w \in \mathbb{N}} \{\varphi'(p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i)) p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i) - \varphi(p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i))\} \\
&\qquad - \varphi'(p_{\boldsymbol{\theta}}(w_i \mid \boldsymbol{x}_i)) + \text{Const.} \\
&\qquad \left(\because p_{\boldsymbol{\theta},iw} = p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i), \hat{q}_{iw} = \begin{cases} 1 & (w = w_i) \\ 0 & (w \neq w_i) \end{cases} \right)
\end{aligned}
$$

holds; minimizing (5.4) equipped with the empirical distributions $\{\hat{\boldsymbol{q}}_i\}_{i=1}^n$ is equivalent to minimizing

$$
\frac{1}{n} \sum_{i=1}^{n} \Bigg\{ \underbrace{\sum_{w \in \mathbb{N}_0} (\varphi'(p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i)) p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i) - \varphi(p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i)))}_{(\star)} - \varphi'(p_{\boldsymbol{\theta}}(w_i \mid \boldsymbol{x}_i)) \Bigg\}. \tag{5.5}
$$

(5.5) appears in some existing studies, such as Ghosh, Basu, et al. (2013) for β-divergence in Table 5.1. However, the term $(\star)$ in eq. (5.5) is computationally intractable due to the infinite summation $\sum_{w \in \mathbb{N}_0}$; there remain a computational challenge in this approach. The infinite summation similarly appears in eq. (4) of Kawashima and Fujisawa (2019), and they compute the term by the finite-sum approximation instead. Note that, the term $(\star)$ reduces to $\sum_{w \in \mathbb{N}_0} p_{\boldsymbol{\theta}}(w \mid \boldsymbol{x}_i) = 1$ if the generating function is specified as $\varphi(x) = x \log x - x$; the computational issue does not occur if KL-divergence

is considered.

For solving the computational challenge, we also consider the second approach. This second approach simply matches the underlying expectation function $\mu_*(\boldsymbol{x}_i) = \mathbb{E}(w_i \mid \boldsymbol{x}_i)$ and the parametric model $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ without assuming any specific probability distribution for $w_i \mid \boldsymbol{x}_i$; we may obtain the estimator of $\boldsymbol{\theta}$ by minimizing

$$D_\varphi(\{\mu_*(\boldsymbol{x}_i)\}_{i=1}^n, \{\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\}_{i=1}^n), \tag{5.6}$$

where φ is a user-specified generating function whose domain $\mathrm{dom}(\varphi)$ includes the set $\mathcal{S}$. However, the underlying expectation function μ_* cannot be observed in practice; we instead minimize

$$\begin{aligned} &D_\varphi(\{w_i\}_{i=1}^n, \{\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\}_{i=1}^n) \\ &\quad = \frac{1}{n}\sum_{i=1}^n \left\{\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \varphi(\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) - w_i\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\right\} + C \end{aligned} \tag{5.7}$$

that approximates (5.6) in the sense that the underlying true conditional expectation $\mu_*(\boldsymbol{x}_i) = \mathbb{E}(w_i \mid \boldsymbol{x}_i)$ is replaced with the observation w_i. $C := \frac{1}{n}\sum_{i=1}^n \varphi(w_i)$ is a constant independent of the parameter $\boldsymbol{\theta}$. (5.7) reduces to Zhang, Jiang, and Shang (2009) eq. (20), if the model is specified as $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}) = \nu(\boldsymbol{\theta}^\top \boldsymbol{x})$ for some non-linear function $\nu : \mathbb{R} \to \mathbb{R}$, whereas arbitrary similarity function $\mu_{\boldsymbol{\theta}}$ is considered in this study.

The second approach bypasses the computational challenge of the first approach, since (5.7) does not include any infinite summation; we consequently employ the second approach, and generalize it from $U = 1$ to $U \in \mathbb{N}$ as shown in the next section.

5.3.2 Proposed BHLR

We here consider HLR with arbitrary $U \in \mathbb{N}$, for predicting the hyperlink weights $w_{\boldsymbol{i}}$ taking values in a set $\mathcal{S} \subset \mathbb{R}$ via a user-specified symmetric similarity function $\mu_{\boldsymbol{\theta}} : \mathcal{X}^U \to \mathcal{S}$. By generalizing the loss function (5.7) from $U = 1$ to $U \in \mathbb{N}$, we propose to minimize a simple loss function

$$\begin{aligned} L_{\varphi,n}(\boldsymbol{\theta}) &:= D_\varphi(\{w_{\boldsymbol{i}}\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}}, \{\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}}) \\ &= \frac{1}{|\mathcal{I}_n^{(U)}|}\sum_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}} \Big\{\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) \\ &\qquad\qquad - \varphi(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) - w_{\boldsymbol{i}}\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))\Big\} + C, \end{aligned} \tag{5.8}$$

where φ is a user-specified generating function whose domain $\mathrm{dom}(\varphi)$ includes the set $\mathcal{S}$, and $C := \frac{1}{|\mathcal{I}_n^{(U)}|}\sum_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}} \varphi(w_{\boldsymbol{i}})$ is a constant independent of

the parameter $\boldsymbol{\theta}$. Subsequently, the estimator is defined as

$$\hat{\boldsymbol{\theta}}_{\varphi,n} := \operatorname*{arg\,min}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} L_{\varphi,n}(\boldsymbol{\theta}). \tag{5.9}$$

Once the estimator $\hat{\boldsymbol{\theta}}_{\varphi,n}$ is obtained, we may predict $w_{\boldsymbol{i}}$ by the estimated similarity function $\mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}(\boldsymbol{X}_{\boldsymbol{i}})$. We formally define predicting $w_{\boldsymbol{i}}$ by the function $\mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}(\boldsymbol{X}_{\boldsymbol{i}})$ as the BHLR.

Since the hyperlink weights are symmetric, we assume that the function $\mu_{\boldsymbol{\theta}}$ also satisfies the symmetry

$$\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U}) = \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_{i'_1}, \boldsymbol{x}_{i'_2}, \ldots, \boldsymbol{x}_{i'_U}) \tag{5.10}$$

for any $\boldsymbol{i}' = (i'_1, i'_2, \ldots, i'_U)$ obtained by permutating the elements of $\boldsymbol{i} = (i_1, i_2, \ldots, i_U) \in \mathcal{I}_n^{(U)}$. This symmetry should hold for all $\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U} \in \mathcal{X}$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$; the similarity function $\mu_{\boldsymbol{\theta}}$ in effect ignores the order of the vectors, as long as (5.10) is assumed. An example of such a symmetric similarity function is

$$\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U}) = \nu\left(\langle \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_2}), \cdots, \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_U})\rangle\right), \tag{5.11}$$

where $\boldsymbol{f}_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^K$ represents a function parametrized by $\boldsymbol{\theta}$, e.g., vector-valued neural networks $\boldsymbol{f}_{\mathrm{NN}}$, $\nu : \mathbb{R} \to \mathcal{S}$ is a link function, e.g., exponential function for $\mathcal{S} = \mathbb{R}_{\geq 0}$ and sigmoid function for $\mathcal{S} = [0, 1]$, and $\langle \boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}'', \ldots\rangle := \sum_{k=1}^K y_k y'_k y''_k \cdots$. The function (5.11) is employed in our numerical experiments later in Section 5.6, and it reduces to tensor decomposition explained in Section 5.4.3 if $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$, $\nu(z) = z$, and $\boldsymbol{x}_i$ is a 1-hot vector.

The similarity (5.11) reduces to IPS defined in the previous Chapter 4 by specifying $U = 2$. Then, more expressive SIPS and IPDS, that are also defined in the previous Chapter 4, can be employed as the similarity model (5.11).

The BHLR reduces to several existing methods, such as logistic regression ($U = 1$), Poisson regression ($U = 1$), and link prediction ($U = 2$), by specifying $\mu_{\boldsymbol{\theta}}$ and φ. Furthermore, BHLR also reduces to several methods for representation learning, such as graph embedding ($U = 2$), matrix factorization ($U = 2$), tensor factorization ($U \geq 2$), and their variants equipped with arbitrary BD. We describe the relation between the BHLR and these existing methods in Section 5.4.

In addition to the rich examples for the BHLR family, the BHLR possesses the following two favorable properties: (P-1) statistical consistency, and (P-2) computational tractability. We further explain these properties (P-1) and (P-2) in Section 5.5.1 and Section 5.5.2, respectively, along with the proposal of a novel and generalized minibatch sampling procedure for hyper-relational data that can be used for efficient stochastic algorithms.

Furthermore, BHLR is also expected to be a robust estimation, by virtue of the Bregman divergence; we discuss the intuition of the estimation robustness in the following Section 5.3.3.

5.3.3 Estimation Robustness of BHLR

BHLR is expected to be a robust estimation, by virtue of the nature of the Bregman divergence. In this section, we first describe the estimating equation and the intuition behind the BHLR, for subsequently describing the intuition of the estimation robustness.

Estimating Equation

The estimating equation for BHLR is

$$\frac{\partial L_{\varphi,n}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \gamma_\varphi(\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})) \left\{ (\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i}) - w_{\boldsymbol{i}}) \frac{\partial \log \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})}{\partial \boldsymbol{\theta}} \right\} = \mathbf{0} \tag{5.12}$$

where $\gamma_\varphi(z) := z\varphi''(z)$.

Let $w, \boldsymbol{X}$ be random variables of hyperlink weight and the corresponding tuple, that are linked by the conditional probability function q defined in Section 5.2.3. Under some assumptions with sufficiently large n, the law of large numbers (LLN), that is formally described in Theorem C.1 in Appendix C.3.1 for hyper-relational setting, indicates that

$$\mathbb{E}_{\mathcal{X}^U} \left(\underbrace{\gamma_\varphi(\mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}(\boldsymbol{X}))}_{(\star 1)} \underbrace{(\mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}(\boldsymbol{X}) - \mu_*(\boldsymbol{X}))}_{(\star 2)} \underbrace{\frac{\partial \log \mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}(\boldsymbol{X})}{\partial \boldsymbol{\theta}}}_{(\star 3)} \right) \approx \mathbf{0}, \tag{5.13}$$

where $\mu_*(\boldsymbol{X}) := \mathbb{E}(w \mid \boldsymbol{X})$ and $\mathbb{E}_{\mathcal{X}^U}$ takes expectation with respect to the tuple $\boldsymbol{X}$. The factor $(\star 3)$ represents a vector-valued function, usually high-dimensional and non-zero; at least either of factors $(\star 1), (\star 2)$ is expected to be approximately 0 for each $\boldsymbol{X}$, for simultaneously satisfying the entry-wise estimating equation (5.13).

Intuition behind the BHLR

We hereinafter consider a typical case $\mathcal{S} \subset \mathbb{R}_{\geq 0}$, meaning that the hyperlink weights take only the non-negative values. Friend network and co-authorship network shown in Examples 5.2.2 and 5.2.2 coincide with this case. When considering KL-divergence, the function γ_φ reduces to a constant

$$\gamma_{\varphi_{\mathrm{KL}}}(z) = 1 \quad (\forall z \in \mathcal{S}),$$

namely $(\star 1) = 1$. Then, the factor $(\star 2)$ is expected to be approximately 0 for all $\boldsymbol{X}$, and it indicates that $\mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}} \approx \mu_*$.

On the other hand, we here consider replacing the KL divergence with other Bregman divergences. Regarding the other divergences listed in Table 5.1, we obtain

$$\gamma_{\varphi_{\text{Logistic}}}(z) = \frac{1}{1-z}, \quad \gamma_{\varphi_\beta}(z) = z^\beta, \quad \gamma_{\varphi_{\text{Itakura-Saito}}}(z) = \frac{1}{z},$$
$$\gamma_{\varphi_{\text{Inverse}}}(z) = \frac{2}{z^2}, \quad \gamma_{\varphi_{\text{Exponential}}}(z) = z\exp(z).$$

Therefore, employing Bregman divergence corresponds to incorporating weight function γ_φ into the estimating equation of KL-divergence. We plot the function γ_φ for Bregman divergences in the following Figure 5.3.

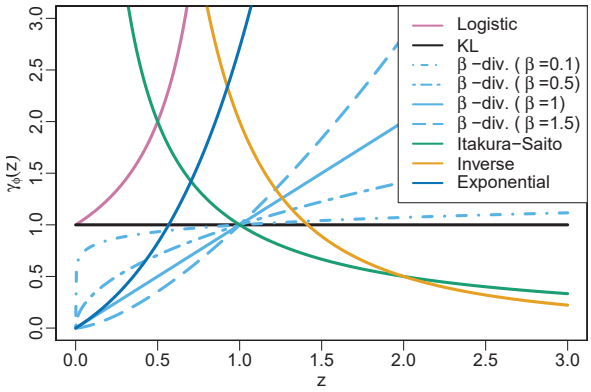


FIGURE 5.3: $\gamma_\varphi(z)$ is plotted for each Bregman divergence listed in Table 5.1.

As can be seen in Figure 5.3, the function $\gamma_\varphi(z)$ for β-divergence and exponential divergence takes values near 0 if $z \approx 0$. Therefore, $(\star 1)$ in (5.13) is not always non-zero unlike the KL-divergence case; $(\star 2)$ does not need to be nearly equal to 0, as long as these divergences are employed and $\mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}(\boldsymbol{X}) \approx 0$. This property yields the robustness of the estimation, that is also discussed from a different perspective in the original paper of β-divergence (Basu et al., 1998). Then, the robustness property in our setting is roughly described as follows.

Estimation Robustness of BHLR

By referring to the above intuition, we here describe the intuition of the estimation robustness of BHLR. We consider the case that $\mu_*(\boldsymbol{X}) = \mathbb{E}(w \mid \boldsymbol{X})$ is represented as a combination of the true function μ_{true} and the contamination function $\mu_{\text{cont.}}$, i.e.,

$$\mu_*(\boldsymbol{X}) = \mu_{\text{true}}(\boldsymbol{X}) + \alpha\mu_{\text{cont.}}(\boldsymbol{X}) \tag{5.14}$$

for some small but nonzero $\alpha > 0$. We assume that these functions $\mu_{\text{true}}, \mu_{\text{cont.}}$ are "far apart", and $\mu_{\text{cont.}}$ is also assumed to have a finite "small" support for description simplicity, as illustrated in the following Figure 5.4. Then, we aim at estimating the true function $\mu_{\text{true.}}$ via the contaminated function μ_*.

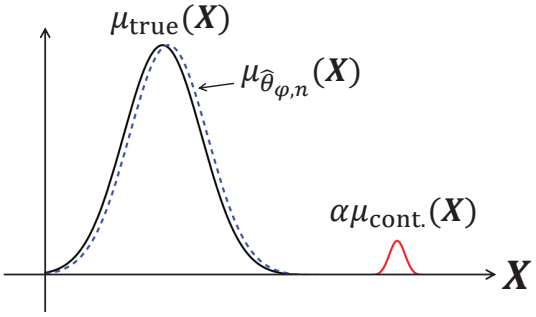


FIGURE 5.4: Assuming that (i) functions μ_{true} and $\mu_{\text{cont.}}$ are "far apart", and (ii) $\mu_{\text{cont.}}$ has a finite "small" support.

In this setting, the function $\mu_{\hat{\theta}_{\varphi,n}}$ estimated via KL-divergence fits μ_* for all $\boldsymbol{X}$ (as the factor $(\star 2)$ is equally weighted by $(\star 1) = 1$ for all $\boldsymbol{X}$), namely,

$$\mu_{\hat{\theta}_{\varphi_{\text{KL}},n}} \approx \mu_* \quad (= \mu_{\text{true}} + \alpha\mu_{\text{cont.}}).$$

It means that the function fits both functions μ_{true} and $\mu_{\text{cont.}}$.

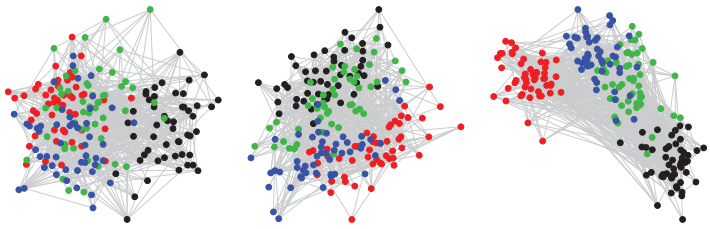
However, on the other hand, β-divergence may ignore $\mu_{\text{cont.}}$ as long as the estimated function satisfies $\mu_{\hat{\theta}_{\varphi,n}}(\boldsymbol{X}) \approx 0$ for all the $\boldsymbol{X}$ in the support of $\mu_{\text{cont.}}$. This is because, as described in the intuition behind the BHLR, the factor $(\star 1)$ in (5.13) takes values near 0 if $\mu_{\hat{\theta}_{\varphi,n}}(\boldsymbol{X}) \approx 0$, whereupon $(\star 2)$ does not need to be 0. Similar also holds for exponential divergence. Therefore, the function estimated via β or exponential divergence may ignore the contamination function $\mu_{\text{cont.}}$ and approximate the true function μ_{true}, i.e.,

$$\mu_{\hat{\theta}_{\varphi_\beta,n}} \approx \mu_{\text{true}}, \quad \text{and} \quad \mu_{\hat{\theta}_{\varphi_{\text{Exponential}},n}} \approx \mu_{\text{true}},$$

depending on the setting and the user-specified function μ_θ. Although the above estimation depends on which of the local minimum the estimator $\hat{\boldsymbol{\theta}}_{\varphi,n}$ approaches, and the explanation is adapted to our setting, it is empirically examined that the β-divergence based estimation is robust against contamination functions, especially in regression case ($U = 1$; Basu et al. (1998) and Kanamori and Fujisawa (2015)). The estimation based on β-divergence is considered in a variety of studies (Minami and Eguchi, 2002; Tan and Févotte, 2012), including matrix factorization and tensor factorization (Cichocki et al., 2009).

In addition to the robust regression ($U = 1$), β-divergence can be applied to graph embedding ($U = 2$) instead of leveraging KL-divergence, as published in a paper of ours (Okuno and Shimodaira, 2019). The graph embedding based on β-divergence is especially called β-graph embedding, and the estimation robustness is also empirically illustrated in the following Figure 5.5.

Although the β-graph embedding is limited to the case $U = 2$, it is further generalized to arbitrary U and arbitrary divergences by the proposed BHLR. In this sense, BHLR is expected to be robust against contamination function in (5.14), by employing several specific divergences.



(A) KL ($\Leftrightarrow \beta = 0$) (B) $\beta = 0.5$ (C) $\beta = 1.0$

FIGURE 5.5: β-graph embedding using linear transformation $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$ is applied to $n = 200$ synthetic data vectors having graph-structured associations across 4 different clusters. Node color represents the cluster to which the corresponding data vector belongs, and the data vectors belonging to the same cluster are associated. Gray lines represent (contaminated) positive link weights across different clusters; comparing to the KL-based estimation shown in (A), β-GE shown in (B) and (C) well recover the colored clusters. See, Appendix C.1 for further details on the synthetic dataset.

We last note the case that the remaining divergences are employed. Whereas the function $\gamma_\varphi(z)$ increases as z increases for β and exponential divergences, it decreases for Itakura-Saito and inverse divergences as shown in Figure 5.3. Therefore, the BHLR equipped with Itakura-Saito or inverse divergence has the opposite effect. Empirically speaking, the optimization of BHLR using these divergences is not numerically stable in our problem setting; these divergences are not employed in our numerical experiments in Section 5.6.

5.4 BHLR Family Members

In this section, we describe the BHLR family members by specifying $U \in \mathbb{N}$ and the generating function φ in Sections 5.4.1–5.4.3 and Table 5.2.

Before explaining the BHLR family members, we first explicitly derive the corresponding loss functions $L_{\varphi,n}(\boldsymbol{\theta})$ associated with some generating functions $\varphi_{\text{Logistic}}(x) := x\log x + (1-x)\log(1-x)$, $\varphi_{\text{KL}}(x) := x\log x - x$, $\varphi_{\text{Quad.}}(x) := x^2 - x$ and $\varphi_\beta(x) := \frac{x^{1+\beta}}{\beta(1+\beta)} - \frac{x}{\beta}$, that are listed in Table 5.1. Subsequently, for an arbitrary $U \in \mathbb{N}$, we have

$$L_{\varphi_{\text{Logistic}},n}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{i \in \mathcal{I}_n^{(U)}} \{ -w_i \log \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i) - (1-w_i)\log(1-\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i))\} + C^{(U)}_{\text{Logistic}}, \tag{5.15}$$

$$L_{\varphi_{\text{KL}},n}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{i \in \mathcal{I}_n^{(U)}} \{-w_i \log \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i) + \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i)\} + C^{(U)}_{\text{KL}}, \tag{5.16}$$

$$L_{\varphi_{\text{Quad.}},n}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{i \in \mathcal{I}_n^{(U)}} (w_i - \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i))^2, \tag{5.17}$$

$$L_{\varphi_\beta,n}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}} \left\{ -\frac{1}{\beta} w_{\boldsymbol{i}} \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})^{\beta} + \frac{1}{1+\beta} \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})^{1+\beta} \right\} + C_\beta^{(U)}, \quad (5.18)$$

respectively, where

$$C_{\text{Logistic}}^{(U)} := \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}} \{w_{\boldsymbol{i}} \log w_{\boldsymbol{i}} + (1-w_{\boldsymbol{i}}) \log(1-w_{\boldsymbol{i}})\},$$

$$C_{\text{KL}}^{(U)} := \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}} \{w_{\boldsymbol{i}} \log w_{\boldsymbol{i}} - w_{\boldsymbol{i}}\}, \quad C_\beta^{(U)} := \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}} \frac{w_{\boldsymbol{i}}^{1+\beta}}{\beta(1+\beta)}$$

are constants independent of the parameter $\boldsymbol{\theta}$. By utilizing these loss functions (5.15)–(5.18), and sets

$$\begin{aligned}
\mathcal{A}(p,K) &:= \{\boldsymbol{\theta} = (\theta_{ij}) \in \mathbb{R}^{p\times K} \mid \theta_{ij} \geq 0,\ \forall (i,j) \in [p]\times[K]\},\\
\mathcal{F}(p,K) &:= \{\boldsymbol{\theta} \mid \boldsymbol{\theta} \text{ is a parameter for the vector-valued}\\
&\qquad\qquad \text{neural network } \boldsymbol{f}_{\boldsymbol{\theta}} : \mathbb{R}^p \to \mathbb{R}^K\},\\
\mathcal{C}(n_1,n_2,\ldots,n_U) &:= \{\boldsymbol{i} = (i_1,i_2,\ldots,i_U) \mid i_1 = 1,2,\ldots,n_1;\\
&\qquad\qquad i_2 = n_1+1, n_1+2,\ldots,n_1+n_2;\cdots;\\
&\qquad\qquad i_U = \sum_{u=1}^{U-1} n_u + 1,\ldots,\sum_{u=1}^{U} n_u\},
\end{aligned}$$

various existing methods can be regarded as the BHLR family members, as shown in Table 5.2. A detailed explanation of the BHLR family members are provided in Section 5.4.1 for $U = 1$, Section 5.4.2 for $U = 2$, and Section 5.4.3 for $U \geq 2$.

5.4.1 $U = 1$

- **Least-squares (LS) regression** (Bishop, 2006) is equivalent to minimizing $L_{\varphi_{\text{Quad.}},n}(\boldsymbol{\theta})$, and similarly, **logistic regression** (Bishop, 2006) and **Poisson regression** (Cameron and Trivedi, 2007) minimize $L_{\varphi_{\text{Logistic}}}(\boldsymbol{\theta})$ and $L_{\varphi_{\text{KL}}}(\boldsymbol{\theta})$, respectively. The regression function $f_{\boldsymbol{\theta}} : \mathbb{R}^p \to \mathbb{R}$ used in the regression methods above can be specified arbitrarily. Whereas linear transformation $\boldsymbol{\theta}^\top \boldsymbol{x}_i \in \mathbb{R}$ is typically used (Zhang, Jiang, and Chai, 2010), NNs are incorporated currently for enhancing the expressive power of the regression function.

- **Parametric Bregman-divergence regression** (PBDR; Zhang, Jiang, and Shang (2009)) generalizes Poisson regression, logistic regression and least squares (LS) regression; it is equivalent to the BHLR equipped with arbitrary generating functions φ and functions $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})$ in the form of $\nu(\boldsymbol{\theta}^\top \boldsymbol{x}_{i_1})$ for some function $\nu : \mathbb{R} \to \mathbb{R}$. The PBDR is a special case of the BHLR. However, PBDR considers only the limited form of functions $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})$, whereas BHLR can employ arbitrary function including neural networks.

TABLE 5.2: BHLR family members.

	Method	$\mathcal{S}$	φ	$\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i)$	$\boldsymbol{\Theta}$	$\mathcal{I}_n^{(U)}$	$\{\boldsymbol{x}_i\}_{i=1}^n$
$U=1$	Poisson reg. (Cameron and Trivedi, 2007)	$\mathbb{R}_{\geq 0}$	φ_{KL}	$\exp(\boldsymbol{\theta}^\top \boldsymbol{x}_{i_1})$ or $\exp(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}))$	$\mathbb{R}^p$ or $\mathcal{F}(p,1)$	$[n]$	observed
	Logistic reg. (Bishop, 2006)	$[0,1]$	$\varphi_{\mathrm{Logistic}}$	$\sigma(\boldsymbol{\theta}^\top \boldsymbol{x}_{i_1})$ or $\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}))$	$\mathbb{R}^p$ or $\mathcal{F}(p,1)$	$[n]$	observed
	LS reg. (Bishop, 2006)	$\mathbb{R}$	$\varphi_{\mathrm{Quad.}}$	$\boldsymbol{\theta}^\top \boldsymbol{x}_{i_1}$ or $f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1})$	$\mathbb{R}^p$ or $\mathcal{F}(p,1)$	$[n]$	observed
	PBDR (Zhang, Jiang, and Shang, 2009)	any	any†	$g(\boldsymbol{\theta}^\top \boldsymbol{x}_i)$ for some g	$\mathbb{R}^p$	$[n]$	observed
$U=2$	Matrix Fact. (Koren, Bell, and Volinsky, 2009)	any	any†	$\langle \boldsymbol{\theta}^\top \boldsymbol{x}_{i_1}, \boldsymbol{\theta}^\top \boldsymbol{x}_{i_2} \rangle$	$\mathbb{R}^{(n_1+n_2)\times K}$	$\mathcal{C}(n_1,n_2)$	1-hot $\in \{0,1\}^{n_1+n_2}$
	NMF (Cichocki et al., 2009)	any	any†	$\langle \boldsymbol{\theta}^\top \boldsymbol{x}_{i_1}, \boldsymbol{\theta}^\top \boldsymbol{x}_{i_2} \rangle$	$\mathcal{A}(n_1+n_2,K)$	$\mathcal{C}(n_1,n_2)$	1-hot $\in \{0,1\}^{n_1+n_2}$
	LINE (Tang et al., 2015)	$[0,1]$	$\varphi_{\mathrm{Logistic}}$	$\sigma(\langle f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}), f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_2}) \rangle)$	$\mathcal{F}(p,K)$	any	1-hot $\in \{0,1\}^n$
	KL-GE (Okuno, Hada, and Shimodaira, 2018)	$\mathbb{R}_{\geq 0}$	φ_{KL}	$\exp(\langle f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}), f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_2}) \rangle)$	$\mathcal{F}(p,K)$	any	observed
	β-GE (Okuno and Shimodaira, 2019)	$\mathbb{R}_{\geq 0}$	φ_β	$\exp(\langle f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}), f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_2}) \rangle)$	$\mathcal{F}(p,K)$	any	observed
	Poincaré Emb. (Nickel and Kiela, 2017)	$[0,1]$	$\varphi_{\mathrm{Logistic}}$	$\sigma(-d_{\text{Poincaré}}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}), f_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_2})))$	$\mathcal{F}(p,K)$	any	1-hot $\in \{0,1\}^n$
	SBM (Holland, Laskey, and Leinhardt, 1983)	$[0,1]$	$\varphi_{\mathrm{Logistic}}$	$\theta_1 \mathbf{1}(x_{i_1} = x_{i_2}) + \theta_2 \mathbf{1}(x_{i_1} \neq x_{i_2})$	$[0,1]^2$	$[n]^2$	cluster indicator $\in [C]$
$U \geq 2$	PARAFAC (Bro, 1997)	any	any†	$\langle \boldsymbol{\theta}^\top \boldsymbol{x}_{i_1}, \boldsymbol{\theta}^\top \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{\theta}^\top \boldsymbol{x}_{i_U} \rangle$	$\mathbb{R}^{(\sum_{u=1}^U n_u)\times K}$	$\mathcal{C}(n_1,n_2,\cdots,n_U)$	1-hot $\in \{0,1\}^{\sum_{u=1}^U n_u}$
	NTF (Cichocki et al., 2009)	any	any†	$\langle \boldsymbol{\theta}^\top \boldsymbol{x}_{i_1}, \boldsymbol{\theta}^\top \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{\theta}^\top \boldsymbol{x}_{i_U} \rangle$	$\mathcal{A}(\sum_{u=1}^U n_u, K)$	$\mathcal{C}(n_1,n_2,\cdots,n_U)$	1-hot $\in \{0,1\}^{\sum_{u=1}^U n_u}$

†domain of the generating function φ should include the set $\mathcal{S}$.

5.4.2 $U = 2$

- **Matrix factorization** (MF; Koren, Bell, and Volinsky (2009)) decomposes a given matrix $\boldsymbol{V} = (v_j) \in \mathbb{R}^{n_1 \times n_2}$ into matrices $\boldsymbol{\xi}^{(u)} \in \mathbb{R}^{n_u \times K}$ $(u = 1, 2)$, by minimizing the BD between entries of $\boldsymbol{V}$ and those of $\boldsymbol{\xi}^{(1)}\boldsymbol{\xi}^{(2)\top}$. Subsequently, we can expect that $\boldsymbol{V} \approx \boldsymbol{\xi}^{(1)}\boldsymbol{\xi}^{(2)\top}$.

  Here, we briefly explain that the BHLR includes MF as a special case, by considering link weights

$$\boldsymbol{W} = (w_{\boldsymbol{i}}) = \begin{pmatrix} \boldsymbol{O}_{n_1 \times n_1} & \boldsymbol{V} \\ \boldsymbol{V}^\top & \boldsymbol{O}_{n_2 \times n_2} \end{pmatrix}, \tag{5.19}$$

  and $(n_1 + n_2)$-dimensional 1-hot data vectors $\{\boldsymbol{x}_i\}_{i=1}^{n_1+n_2}$.

  Using the parameter $\boldsymbol{\theta} = (\boldsymbol{\xi}^{(1)\top}, \boldsymbol{\xi}^{(2)\top})^\top \in \mathbb{R}^{(n_1+n_2) \times K}$ and an index set $\mathcal{C}(n_1, n_2) := \{(i_1, i_2) \mid i_1 = 1, 2, \ldots, n_1; i_2 = n_1 + 1, n_1 + 2, \ldots, n_1 + n_2\}$, it holds that

$$\begin{aligned} &D_\varphi(\{v_{\boldsymbol{j}}\}_{\boldsymbol{j} \in [n_1] \times [n_2]}, \{(\boldsymbol{\xi}^{(1)}\boldsymbol{\xi}^{(2)\top})_{\boldsymbol{j}}\}_{\boldsymbol{j} \in [n_1] \times [n_2]}) \\ &\quad = D_\varphi(\{w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{C}(n_1, n_2)}, \{\langle \boldsymbol{\theta}^\top \boldsymbol{x}_{i_1}, \boldsymbol{\theta}^\top \boldsymbol{x}_{i_2} \rangle\}_{\boldsymbol{i} \in \mathcal{C}(n_1, n_2)}), \end{aligned} \tag{5.20}$$

  where $v_{\boldsymbol{j}}$ and $w_{\boldsymbol{i}}$ represent elements of the matrices $\boldsymbol{V}$ and $\boldsymbol{W}$, respectively. Thus, MF minimizing the objective on the left-hand side is equivalent to the BHLR minimizing the objective on the right-hand side. Although MF employs the quadratic loss $L_{\varphi_{\text{Quad.}}, n}(\boldsymbol{\theta})$ in many cases, MF is in fact defined with an arbitrary BD (Cichocki et al., 2009).

  MF $(U = 2)$ can be generalized to $U \geq 2$, where the generalization is called tensor factorization (TF). We describe TF in the following section, and its relation to the BHLR is described in detail in Appendix C.2.

  Finally, MF is called a **non-negative MF** (NMF; Cichocki et al. (2009)) if the entries of the decomposed matrices $\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}$ are restricted to be non-negative.

- **Graph embedding** discussed thus far is a method for representation learning, that trains the transformation $\boldsymbol{f}_{\boldsymbol{\theta}} : \mathcal{X} (\subset \mathbb{R}^p) \to \mathbb{R}^K$ with a user-specified dimension $K \in \mathbb{N}$, such that the link weight $w_{\boldsymbol{i}} \geq 0$ is predicted through $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) = \nu(\langle \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_2}) \rangle)$. As discussed in Chapter 4, we may employ more expressive SIPS and IPDS instead of the IPS therein. $\boldsymbol{\theta}$ is a parameter vector to be estimated by minimizing $L_{\varphi_{\text{Logistic}}, n}(\boldsymbol{\theta})$ with sigmoid function $\nu(\cdot) = \sigma(\cdot)$ in **LINE** (Tang et al., 2015), and $L_{\varphi_{\text{KL}}, n}(\boldsymbol{\theta})$ with $\nu(\cdot) = \exp(\cdot)$ in 1-view version of PMvGE (Okuno, Hada, and Shimodaira, 2018) discussed in Chapter 3, which we denote as **KL-GE** herein. The KL-divergence is replaced with β-divergence in β-graph embedding (β-GE; Okuno and Shimodaira (2019)), as discussed in Section 5.3.3.

- **Stochastic block model** (SBM; Holland, Laskey, and Leinhardt (1983)) considers a graph for which each node $i \in [n]$ is associated with the cluster index $x_i \in [C]$. The SBM learns $\theta_1, \theta_2 \in [0, 1]$, representing probabilities that a link exists between two nodes belonging to the same cluster and different clusters, respectively. As the probability $\mathbb{P}(w_i = 1 \mid \boldsymbol{X}_i)$ is expressed as $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i) := \theta_1 \mathbf{1}(x_{i_1} = x_{i_2}) + \theta_2 \mathbf{1}(x_{i_1} \neq x_{i_2})$ and the parameter $\boldsymbol{\theta} = (\theta_1, \theta_2)$ is learned by minimizing $L_{\varphi_{\text{Logistic}}, n}(\boldsymbol{\theta})$, the SBM is a special case of the BHLR.

5.4.3 $U \geq 2$

- **PARAFAC** (Cichocki et al., 2009; Cong et al., 2015), that is also called TF, CP-decomposition, and CANDECOMP, decomposes a given tensor $\boldsymbol{V} := (v_j) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_U}$ into matrices $\boldsymbol{\xi}^{(u)} := (\xi_{jk}^{(u)}) \in \mathbb{R}^{n_u \times K}$ $(u \in [U])$, by minimizing the BD between entries of $\boldsymbol{V}$ and $[\![\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \ldots, \boldsymbol{\xi}^{(U)}]\!]$ whose $\boldsymbol{j} = (j_1, j_2, \ldots, j_U)$-th entry is specified as $\sum_{k=1}^{K} \xi_{j_1 k}^{(1)} \xi_{j_2 k}^{(2)} \cdots \xi_{j_U k}^{(U)}$. Subsequently, we can expect that $\boldsymbol{V} \approx [\![\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \ldots, \boldsymbol{\xi}^{(U)}]\!]$. TF $(U \geq 2)$ generalizes the MF $(U = 2)$ explained in Section 5.4.2 because $[\![\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}]\!] = \boldsymbol{\xi}^{(1)} \boldsymbol{\xi}^{(2)\top}$. Similar to MF, TF is a special case of the BHLR. See Appendix C.2 for details.

- PARAFAC is called a **non-negative tensor factorization** (NTF; Cichocki et al. (2009) and Kolda and Bader (2009)) or non-negative PARAFAC, if the entries of the decomposed matrices $\boldsymbol{\xi}^{(u)}$ $(u \in [U])$ are restricted to be non-negative. Although this PARAFAC-based NTF can be applied to general $U \in \mathbb{N}$ (Kolda and Bader, 2009), many different types of NTFs have been developed especially for $U = 3$; by referring to Cichocki et al. (2009) p. 54 Table 1.2, NTF1, NTF2 (Cichocki and Zdunek, 2006), and shifted NTF (Harshman, Hong, and Lundy, 2003) decompose a given tensor into 2 matrices and a tensor, and convolutive NTF (CNTF) and C2NTF (Mørup and Schmidt, 2006) decompose the tensor into a matrix and 2 tensors.

Other Related Works

- **LPP** (He and Niyogi, 2004) and **CDMCA** (Shimodaira, 2016), that are explained in Sections 2.2 and 2.4 in Chapter 2, can be used as methods for $U = 2$. Although they are not in line with the proposed BHLR, they are approximately included in BHLR by considering the quadratic approximation, as described in Appendix A.1.

- **HIMFAC** (Nori, Bollegala, and Kashima, 2012) explained in Section 2.4 in Chapter 2 can be used as a method for $U \geq 2$, though HIMFAC eventually becomes equivalent to CDMCA, that considers the case $U = 2$.

- For $U \geq 2$, **Coordinated matrix minimization (CMM)** (Zhang et al., 2018) efficiently infers a subset of user-specified candidate hyperlinks that are the most suitable to fill the training hypernetworks using a low-rank approximation. However, CMM can find hyperlinks only among the training nodes, implying that it cannot be used for obtaining hyperlinks among test nodes outside the training dataset. CMM is neither an HLR or a BHLR.

- For $U \geq 2$, **Deep Sets** (Zaheer et al., 2017) provides a permutation invariant expressive similarity function $\tilde{\mu}_{\boldsymbol{\theta}} : 2^{\mathcal{X}} \to \mathbb{R}$ defined for sets of data vectors. The function $\tilde{\mu}_{\boldsymbol{\theta}}$ is trained by leveraging KL-divergence and logistic loss, whereas BHLR is equipped with arbitrary Bregman divergence. Although the similarity function of Deep Sets can be used for BHLR, the functional form is more restrictive than those considered in our setting. For paying the price of arbitrary size of vector sets, their Theorem 2 proves that a function $\tilde{\mu}_{\boldsymbol{\theta}} : 2^{\mathcal{X}} \to \mathbb{R}$ is permutation invariant if and only if $\tilde{\mu}_{\boldsymbol{\theta}}$ is in the form of $\tilde{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots) = \rho_{\boldsymbol{\theta}}(\sum_u \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_u}))$ for some functions $\rho_{\boldsymbol{\theta}}$ and $\phi_{\boldsymbol{\theta}}$, by assuming that the set $\mathcal{X}$ is countable, or the dimension of $\mathcal{X}$ is 1.

5.5 BHLR Properties

In this section, we show two favorable properties of BHLR. The first property (P-1) statistical consistency: the BHLR asymptotically recovers the true conditional expectation of link weights, is explained in Section 5.5.1. Additionally, we explain the second property (P-2) computational tractability: the BHLR can be efficiently computed by stochastic algorithms in Section 5.5.2.

5.5.1 Statistical Consistency

In this section, we demonstrate via Theorem 5.5.1 that the similarity function $\mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}(\boldsymbol{X}_{\boldsymbol{i}})$ estimated by the BHLR asymptotically recovers the true conditional expectation $\mu_*(\boldsymbol{X}_{\boldsymbol{i}}) = \mathbb{E}(w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}})$. For proving the asymptotic properties of BHLR in Proposition 5.5.1 and Theorem 5.5.1, only in this section, we specify the increasing order index set as

$$\mathcal{J}_n^{(U)} = \{\boldsymbol{i} \in [n]^U \mid 1 \leq i_1 < i_2 < \cdots < i_U \leq n\}, \tag{5.21}$$

such that the set includes all the possible combinations of U different entries $i_1, i_2, \ldots, i_U \in [n]$, whereas no two distinct indices $\boldsymbol{i}, \boldsymbol{i}' \in \mathcal{J}_n^{(U)}$ are obtained from each other by permutation. Then, hyperlink weights $\{w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{J}_n^{(U)}}$ are free from the symmetry constraints described in Section 5.2.2; the underlying conditional distribution of $w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}}$ can be defined without the constraints, thus making the theoretical development easier.

In the following, we list conditions (C-1)–(C-5) needed for theoretical development. w represents a random variable that follows a cpdf (or cpmf) q of $w \mid \boldsymbol{X}$, for $\boldsymbol{X} = (\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}'', \ldots) \in \mathcal{X}^U$.

(C-1) $\boldsymbol{\Theta}$ is compact.

(C-2) Real-valued functions $\mu_{\boldsymbol{\theta}}(\boldsymbol{X})$ and $\mu_*(\boldsymbol{X}) := \mathbb{E}(w \mid \boldsymbol{X})$ are continuous on $\boldsymbol{\Theta} \times \mathcal{X}^U$ and $\mathcal{X}^U$, respectively. Especially, the function $\mu_{\boldsymbol{\theta}}(\boldsymbol{X})$ is Lipschitz continuous on $\boldsymbol{\Theta}$ for each $\boldsymbol{X} \in \mathcal{X}^U$.

(C-3) Hyperlink weights $\{w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}}$ follow a distribution whose cpdf (or cpmf) are specified as $\prod_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} q(w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}})$, and data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ i.i.d. follow a pdf q_X, where the support of q_X is compact.

(C-4) $\mathbb{E}(w^2 \mid \boldsymbol{X}) < \infty$ and $\mathbb{E}(\varphi(w)^2 \mid \boldsymbol{X}) < \infty$ for all $\boldsymbol{X} \in \mathcal{X}^U$.

(C-5) φ is C^2 and strongly convex.

It is noteworthy that all the functions listed in Table 5.1 satisfy the condition (C-5); all the conditions (C-1)–(C-5) are not difficult to satisfy in practice. Using these conditions, we demonstrate in the following Proposition 5.5.1 that $L_{\varphi,n}(\boldsymbol{\theta})$ empirically approximates the expected value of $d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}))$ up to a constant.

Proposition 5.1

Let $U \in \mathbb{N}$, $\mathcal{I}_n^{(U)} = \mathcal{J}_n^{(U)}$ defined in eq. (5.21) and suppose that (C-1)–(C-5) hold. Let $\mathbb{E}_{\mathcal{X}^U}$ represent the expectation with respect to the density of the U-tuple $\boldsymbol{X} = (\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}'', \ldots) \in \mathcal{X}^U$; more specifically, $\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}'', \ldots$ i.i.d. follow a pdf q_X. Then, for $n \to \infty$, it holds that

$$L_{\varphi,n}(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{X}^U}\left(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}))\right) + C_\varphi + O_p(1/\sqrt{n})$$

for each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, where $C_\varphi := \mathbb{E}_{\mathcal{X}^U}\left(\mathbb{E}(\varphi(w) \mid \boldsymbol{X}) - \varphi(\mu_*(\boldsymbol{X}))\right)$ is a constant independent of the parameter $\boldsymbol{\theta}$.

Proof is obtained by applying the law of large numbers for multiple indexed partially dependent random variables. See Appendix C.3.2 for details.

As explained in Section 5.2.3, different tuples $\boldsymbol{X}_{\boldsymbol{i}}, \boldsymbol{X}_{\boldsymbol{i}'}$ may be constrained as they may share some data vectors, even if data vectors $\boldsymbol{x}_i$ are i.i.d. generated; theories for HLR can be different from those of classical regression, that predicts response variables from i.i.d. explanatory variables. Due to the constraint, the convergence rate of the loss function for BHLR is $O(1/\sqrt{n})$ whereas the estimation leverages $|\mathcal{I}_n^{(U)}| = O(n^U)$ samples. The convergence rate is similar to U-statistic (Lee, 1990), and is different from the rate $O(1/\sqrt{n^U})$ for classical regression using $O(n^U)$ i.i.d. data vectors.

Proposition 5.5.1 leads to the following Theorem 5.5.1, which claims that the estimated model $\mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}$ converges to μ_* in probability, by considering that $d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}))$ with fixed $\mu_*(\boldsymbol{X})$ is minimized if $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}) = \mu_*(\boldsymbol{X})$.

Theorem 5.1: Statistical estimation consistency

The symbols and conditions are the same as those of Proposition 5.5.1 except for the additional condition: there exists $\boldsymbol{\theta}_* \in \boldsymbol{\Theta}$ such that $\mu_{\boldsymbol{\theta}*} = \mu_*$. Using a norm $\|f\| := \mathbb{E}_{\mathcal{X}^U}(f(\boldsymbol{X})^2)^{1/2}$ defined for functions $f : \mathcal{X}^U \to \mathbb{R}$, it holds that

$$\|\mu_* - \mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}\| \xrightarrow{p} 0, \quad (n \to \infty), \tag{5.22}$$

where $\hat{\boldsymbol{\theta}}_{\varphi,n}$ is the estimator (5.9) computed with n data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ and their hyperlink weights $\{w_{\boldsymbol{i}}\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}}$.

Proof is provided in Appendix C.3.3. As indicated in Theorem 5.5.1 above, the estimated similarity function $\mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}$ asymptotically recovers the underlying expectation function μ_* in probability, regardless of the choice of φ. Thus, the BHLR is statistically consistent.

Interestingly, Theorem 5.5.1 does not rely on the underlying conditional distribution of hyperlink weights; BHLR is also robust against the distributional misspecification for the weights, as long as the set of user-specified similarity functions $\{\mu_{\boldsymbol{\theta}}(\boldsymbol{X})\}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}$ includes the conditional expectation $\mu_*(\boldsymbol{X}) := \mathbb{E}(w \mid \boldsymbol{X})$ therein.

Note that similar property is already known for exponential linear regression models (e.g., Poisson regression model), that correspond to BHLR with $U = 1$. See Cameron and Trivedi (2013) Section 2.4.2 and 3.2.3 for details.

5.5.2 Stochastic Optimization and Its Convergence

In this section, we discuss the optimization for the BHLR. We first consider applying the classical fullbatch gradient descent (GD), i.e., GD using all data for computing gradients to obtain the estimator (5.9). Subsequently, we demonstrate that the fullbatch-based methods require considerable computational cost when considering $U \geq 2$. For reducing the computational complexity, we introduce an efficient algorithm based on minibatch stochastic GD (SGD), i.e., GD using a sampled small dataset for computing gradients. Furthermore, we prove the asymptotics of the minibatch SGD, and demonstrate that it increases the ROC–AUC test score in our numerical experiments.

For notational simplicity, $n, U \in \mathbb{N}$, generating function φ, index set $\mathcal{I}_n^{(U)} (\neq \emptyset) \subset [n]^U$, hyperlink weights $\{w_{\boldsymbol{i}}\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}}$, and data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$ are fixed in this section. It is noteworthy that the index set $\mathcal{I}_n^{(U)} \subset [n]^U$ can be arbitrary specified hereinafter, whereas the set $\mathcal{I}_n^{(U)}$ was restricted to have a specific form $\mathcal{J}_n^{(U)} = (5.21)$ in the previous Section 5.5.1 for making the theory easier. For example, both $(1,2)$ and $(2,1)$ can be included in $\mathcal{I}_n^{(2)}$ while only $(1,2)$ was included in $\mathcal{J}_n^{(2)}$.

We begin by obtaining the estimator (5.9) by applying the fullbatch GD with $T \in \mathbb{N}$ iterations started from a randomly initialized vector $\boldsymbol{\theta}^{(1)}$:

$$\boldsymbol{\theta}^{(t+1)} := \mathcal{Q}_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}^{(t)} - \gamma^{(t)} g(\boldsymbol{\theta}^{(t)})\right), \quad t = 1, 2, \ldots, T, \tag{5.23}$$

where $\{\gamma^{(t)}\}_{t=1,2,\ldots,T} \subset \mathbb{R}_{>0}$ are step sizes, $g(\boldsymbol{\theta})$ is the gradient function, and $\mathcal{Q}_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) := \arg\min_{\boldsymbol{\theta}' \in \boldsymbol{\Theta}} \|\boldsymbol{\theta}' - \boldsymbol{\theta}\|_2$ is the projection to the parameter space where $\|\boldsymbol{z}\|_2 := (\sum_{i=1}^{q} z_i^2)^{1/2}$ for $\boldsymbol{z} = (z_1, z_2, \ldots, z_q) \in \mathbb{R}^q$. This projection is required for ensuring that the estimator $\hat{\boldsymbol{\theta}}^{(t)}$ is included in the parameter space $\boldsymbol{\Theta}$; the projection can be ignored if $\boldsymbol{\Theta} = \mathbb{R}^p$. The gradient function is expressed as

$$g(\boldsymbol{\theta}) := \frac{\partial L_{\varphi,n}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{|\mathcal{I}_n^{(U)}|}\Bigg\{ \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}} - \sum_{\boldsymbol{i} \in \mathcal{P}_n^{(U)}} w_{\boldsymbol{i}} \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}} \Bigg\}, \tag{5.24}$$

where $\mathcal{P}_n^{(U)} := \{\boldsymbol{i} \in \mathcal{I}_n^{(U)} \mid w_{\boldsymbol{i}} \neq 0\}$ is a set of indices whose corresponding weights are non-zero. After the T iterations, $\boldsymbol{\theta}^{(T+1)}$ converges to the estimator (5.9) as $T \to \infty$ under some assumptions (Dunn, 1981). However, computing the gradient (5.24) requires considerable computational cost $O(|\mathcal{I}_n^{(U)}|) = O(n^U)$; the significant computational complexity is non-negligible especially for $U \geq 2$.

For efficiently computing the estimator (5.9), we alternatively employ minibatch SGD (Ruder, 2016) that iteratively updates the parameter as

$$\tilde{\boldsymbol{\theta}}^{(t+1)} := \mathcal{Q}_{\boldsymbol{\Theta}}\left(\tilde{\boldsymbol{\theta}}^{(t)} - \gamma^{(t)} \tilde{g}_{\eta}^{(t)}(\tilde{\boldsymbol{\theta}}^{(t)})\right), \quad t = 1, 2, \ldots, T, \tag{5.25}$$

where $\tilde{g}_{\eta}^{(t)}(\boldsymbol{\theta})$ is a stochastic gradient as will be defined in (5.29) using the sampled small dataset called *minibatch*.

Although minibatch sampling can be easily formulated in the case of $U = 1$, several different sampling patterns may occur when $U \geq 2$. For instance, when $U = 2$, the negative-sampling used in skip-gram (Mikolov et al., 2013) first randomly fixes the first entry i_1 in the index $\boldsymbol{i} = (i_1, i_2)$ and subsequently samples a minibatch as shown in Figure 5.6. On the other hand, the minibatch SGD used for PMvGE, which is shown in Section 3.4.2 in Chapter 3, samples a minibatch without fixing any entries in the index. Thus, we unify both of these existing methods in this study, and propose a general procedure for sampling a minibatch that can be used for both $U = 1, 2$ and $U \geq 3$. The proposed general procedure is explained in the following and Algorithm 1.

In the proposed procedure, that generalizes negative sampling ($U = 2, v = 1$) used in skip-gram (Mikolov et al., 2013), we first specify $v \in \{0, 1, 2, \ldots, U -$

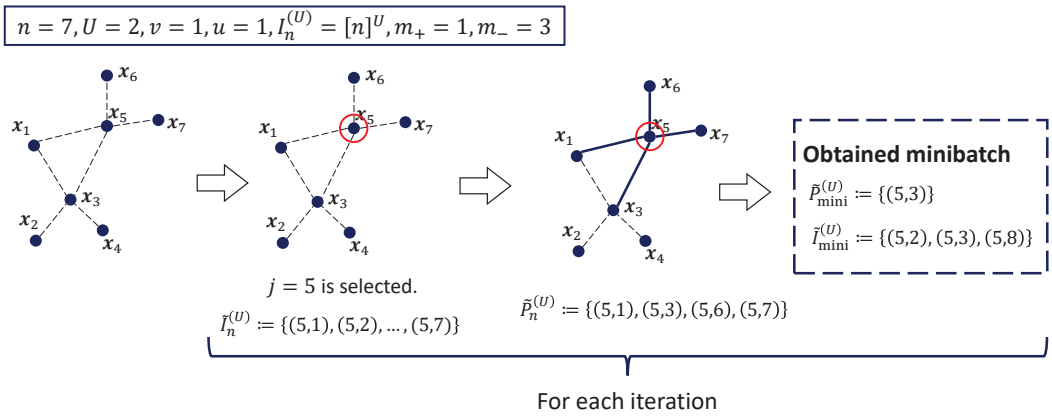


FIGURE 5.6: Negative-sampling used in skip-gram (Mikolov et al., 2013), that is illustrated with $n = 7, u = 1, \mathcal{I}_n^{(U)} = [n]^U$ in this figure, is a special case ($U = 2, v = 1$) of the proposed procedure. For each iteration, (i) $j \in [n]^v (= \{1, 2, \ldots, 7\})$ is randomly selected; $j = 5$ is selected herein. (ii) $\tilde{\mathcal{I}}_n^{(U)}$ is a set of all the possible indices whose $u(= 1)$-th entry is fixed as $j(= 5)$, i.e., $\tilde{\mathcal{I}}_n^{(U)} = \{(5,1), (5,2), (5,3), ..., (5,7)\}$, and $\tilde{\mathcal{P}}_n^{(U)}$ represents a set of indices whose corresponding weights are non-zero (shown as dot lines in this figure), i.e., $\tilde{\mathcal{P}}_n^{(U)} = \{(5,1), (5,3), (5,6), (5,7)\}$. (iii) m_+, m_- entries are randomly selected from sets $\tilde{\mathcal{P}}_n^{(U)}, \tilde{\mathcal{I}}_n^{(U)}$, and denote the sets as $\tilde{\mathcal{P}}_{\text{mini}}^{(U)}, \tilde{\mathcal{I}}_{\text{mini}}^{(U)}$; they are called "minibatch". We further generalize the negative sampling from $U = 2$ to arbitrary $U \in \mathbb{N}$.

1}, that represents the number of entries in the index $\boldsymbol{i}$ to be fixed. $v = 0$ indicates that no entry is fixed; we herein consider $v \geq 1$. For fixing the entries, we specify $\boldsymbol{u}$ in a set

$$\{\boldsymbol{u} = (u_1, u_2, \ldots, u_v) \in [U]^v \mid u_1 < u_2 < \cdots < u_v\}. \tag{5.26}$$

Then, the proposed procedure is summarized in Algorithm 1 using a set of $\boldsymbol{i} \in \mathcal{I}_n^{(U)}$ whose $\boldsymbol{u} = (u_1, u_2, \ldots, u_v)$-th entry is fixed as $\boldsymbol{j} = (j_1, j_2, \ldots, j_v) \in [n]^v$, that is

$$\mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j}) := \{\boldsymbol{i} := (i_1, i_2, \ldots, i_U) \mid \boldsymbol{i} \in \mathcal{I}_n^{(U)}, i_{u_1} = j_1, \ldots, i_{u_v} = j_v\} \tag{5.27}$$

for $\boldsymbol{j} \in [n]^v$, and and a set

$$\mathcal{K}_{\boldsymbol{u}} := \{\boldsymbol{j} \in [n]^v \mid \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j}) \neq \emptyset\}, \tag{5.28}$$

that decomposes the index set as $\mathcal{I}_n^{(U)} = \bigcup_{\boldsymbol{j} \in \mathcal{K}_{\boldsymbol{u}}} \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j})$ without any overlap. $p_{\boldsymbol{j}}$ represents the probability to choose $\boldsymbol{j}$ from the set $\mathcal{K}_{\boldsymbol{u}}$; we employ $p_{\boldsymbol{j}} = 1/|\mathcal{K}_{\boldsymbol{u}}|$ later in Theorem 5.5.2, whereas it can be arbitrarily specified by users in practice. The proposed minibatch sampling for hyper-relations is also illustrated in Example 5.5.2.

Algorithm 1 Proposed minibatch sampling procedure $\mathcal{M}_v(\mathcal{I}_n^{(U)}, \boldsymbol{u}, \{p_{\boldsymbol{j}}\}_{\boldsymbol{j} \in \mathcal{K}_{\boldsymbol{u}}}, m_+, m_-)$.

Require: An index set $\mathcal{I}_n^{(U)} \subset [n]^U$, numbers of minibatch samples $m_+, m_- \in \mathbb{N}$, a vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_v)$ in the set (5.26), and probability $p_{\boldsymbol{j}}$ that samples $\boldsymbol{j}$ from the set $\mathcal{K}_{\boldsymbol{u}}$. Note that $\boldsymbol{u}, \{p_{\boldsymbol{j}}\}_{\boldsymbol{j} \in \mathcal{K}_{\boldsymbol{u}}}$ are not required for $v = 0$.

if $v \geq 1$ **then**

Randomly choose a $\boldsymbol{j}$ from the set $\mathcal{K}_{\boldsymbol{u}}$ defined in (5.28), with the probability $p_{\boldsymbol{j}}$.

$\tilde{\mathcal{I}}_n^{(U)} := \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j})$ defined in (5.27).

else if $v = 0$ **then**

$\tilde{\mathcal{I}}_n^{(U)} := \mathcal{I}_n^{(U)}$, as $v = 0$ indicates no fixed entry in the index $\boldsymbol{i}$.

end if

$\tilde{\mathcal{P}}_n^{(U)} := \{\boldsymbol{i} \mid \boldsymbol{i} \in \tilde{\mathcal{I}}_n^{(U)}, w_{\boldsymbol{i}} \neq 0\}$.

Choose m_+, m_- entries uniformly and randomly from $\tilde{\mathcal{P}}_n^{(U)}, \tilde{\mathcal{I}}_n^{(U)}$, and denote the sets as $\tilde{\mathcal{P}}_{\text{mini}}^{(U)}, \tilde{\mathcal{I}}_{\text{mini}}^{(U)}$.

$s_+ := |\tilde{\mathcal{P}}_n^{(U)}|/m_+, s_- := |\tilde{\mathcal{I}}_n^{(U)}|/m_-$.

Ensure: $(\tilde{\mathcal{P}}_{\text{mini}}^{(U)}, \tilde{\mathcal{I}}_{\text{mini}}^{(U)}, s_+, s_-)$.

Example 5.3: Minibatch sampling for hyper-relations

We consider $n = 7, U = 4, v = 2, \mathcal{I}_n^{(U)} = [n]^U, \boldsymbol{u} = (1,3)$, and $\boldsymbol{j} = (2,5)$ is herein randomly selected. We define a set of indices whose $\boldsymbol{u} = (1,3)$-th entry is fixed as $\boldsymbol{j} = (2,5)$, i.e.,

$$\tilde{\mathcal{I}}_n^{(U)} = \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j}) := \{(2,a,5,b) \in \mathcal{I}_n^{(U)} \mid a,b \in \{1,2,\ldots,7\}\},$$

and a set of indices whose corresponding hyperlink weights are non-zero, i.e., $\tilde{\mathcal{P}}_n^{(U)} = \{\boldsymbol{i} \in \tilde{\mathcal{I}}_n^{(U)} \mid w_{\boldsymbol{i}} \neq 0\}$; they are sets of candidate indices to be resampled. We uniformly and randomly choose m_+, m_- indices from sets $\tilde{\mathcal{P}}_n^{(U)}, \tilde{\mathcal{I}}_n^{(U)}$, and denote the sets as $\mathcal{P}_{\text{mini}}^{(U)}, \mathcal{I}_{\text{mini}}^{(U)}$; they are used for computing the gradient (5.29) and update the parameter by (5.25).

It is noteworthy that the sampling procedure in Algorithm 1 can efficiently pick up non-zero weights even if most of the weights $\{w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}}$ are zero. Similarly to Mikolov et al. (2013) and the stochastic optimization algorithm shown in Section 3.4.2 in Chapter 3, the gradient $g(\boldsymbol{\theta})$ at the iteration t can be stochastically approximated by

$$\tilde{g}_\eta^{(t)}(\boldsymbol{\theta}) := s_-^{(t)} \sum_{\boldsymbol{i} \in \tilde{\mathcal{I}}_{\text{mini}}^{(t)}} \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}} - \eta \cdot s_+^{(t)} \sum_{\boldsymbol{i} \in \tilde{\mathcal{P}}_{\text{mini}}^{(t)}} w_{\boldsymbol{i}} \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}}, \tag{5.29}$$

where the minibatch $\mathcal{M}^{(t)} := (\tilde{\mathcal{P}}_{\text{mini}}^{(t)}, \tilde{\mathcal{I}}_{\text{mini}}^{(t)}, s_+^{(t)}, s_-^{(t)})$ is obtained via Algorithm 1 and $\eta > 0$ is a user-specified parameter. The coefficient $s_-^{(t)} = |\tilde{\mathcal{I}}_n^{(U)}| / |\tilde{\mathcal{I}}_{\text{mini}}^{(t)}|$ is needed for adjusting the first term in the stochastic gradient (5.29), since only the fixed size of minibatch $\tilde{\mathcal{I}}_{\text{mini}}^{(t)}$ is sampled from the set $\tilde{\mathcal{I}}_n^{(U)}$ whose size may depend on the selected $\boldsymbol{j} \in \mathcal{K}_{\boldsymbol{u}}$. Similarly, $s_+^{(t)} = |\tilde{\mathcal{P}}_n^{(U)}| / |\tilde{\mathcal{P}}_{\text{mini}}^{(t)}|$ is needed for adjusting the second term. Although these coefficients $s_+^{(t)}, s_-^{(t)}$ are required for theoretical development, they may be ignored in practice, as demonstrated later.

The computational complexity for the stochastic gradient (5.29) is $O(m_+ + m_-)$, and it can be significantly less than the complexity $O(n^U)$ of the full-batch gradient (5.24), at least for each iteration. Moreover, the minibatch SGD (5.25) using (5.29) reaches approximately the optimal value within a reasonable number of iterations, as will be empirically demonstrated at the last of this section; BHLR can be efficiently computed by the minibatch SGD.

The minibatch SGD equipped with Algorithm 1 and (5.29), can be applied to general $U \geq 2$ and $v \geq 0$ whereas it encompasses several existing methods; in our context, it reduces to the minibatch SGD using the negative sampling

for skip-gram (Mikolov et al., 2013) if $(U, v, \varphi, m_+) = (2, 1, \varphi_{\text{Logistic}}, 1)$, and it also reduces to the stochastic optimization algorithm used for PMvGE (Section 3.4.2 in Chapter 3) if $(U, v, \varphi) = (2, 0, \varphi_{\text{KL}})$. The sampling procedures are called "negative sampling: unigram" $(v = 1)$ and "uniform link sampling" $(v = 0)$ in Veitch et al. (2019). Other major stochastic algorithms such as AdaGrad (Duchi, Hazan, and Singer, 2011) and Adam (Kingma and Ba, 2015) can be employed as well, once the minibatch-based stochastic gradient (5.29) is formally defined with Algorithm 1.

Hereinafter, we discuss the asymptotics of the minibatch SGD when the number of iterations is sufficiently large, by employing Ghadimi and Lan (2013) Theorem 2.1 (a).

Whereas the standard stochastic optimization algorithms preliminary determine the number of iterations T, for theoretical purposes, Ghadimi and Lan (2013) randomly choose the number of iterations τ from the set $[T] = \{1, 2, \ldots, T\}$ with the probability $\mathbb{P}(\tau)$, and update the parameter $\boldsymbol{\theta}$ within τ iterations. In this setting, the expectation of the stochastic gradient $\tilde{g}_\eta^{(\tau)}(\tilde{\boldsymbol{\theta}}^{(\tau)})$ is proved to approach $\mathbf{0}$ as $T \to \infty$; considering the Bregman divergence between the hyperlink weights multiplied by a user-specified constant $\eta > 0$ and the similarities $\{\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i)\}_{i \in \mathcal{I}_n^{(U)}}$, i.e.,

$$Q_\eta(\boldsymbol{\theta}) := D_\varphi(\{\eta w_i\}_{i \in \mathcal{I}_n^{(U)}}, \{\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i)\}_{i \in \mathcal{I}_n^{(U)}}), \tag{5.30}$$

we apply Ghadimi and Lan (2013) to our setting, and show in the following Theorem 5.5.2 that the gradient of $Q_\eta(\boldsymbol{\theta})$ approaches to $\mathbf{0}$ as T increases.

For applying Ghadimi and Lan (2013), we further assume the following conditions (D-1)–(D-3):

(D-1) **Differentiability of $Q_\eta(\boldsymbol{\theta})$:** the loss function $Q_\eta(\boldsymbol{\theta})$ defined in eq. (5.30) is differentiable with respect to $\boldsymbol{\theta}$.

(D-2) **Lipschitz continuity for the gradient of $Q_\eta(\boldsymbol{\theta})$:** using the coefficient $\alpha := \begin{cases} |\mathcal{I}_n^{(U)}| / |\mathcal{K}_{\boldsymbol{u}}| & (v = 1) \\ |\mathcal{I}_n^{(U)}| & (v = 0) \end{cases}$, the gradient $\alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta})$ is H-Lipschitz continuous for some $H > 0$, i.e., $\|\alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta}) - \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta}')\|_2 \leq H \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$, $(\forall \boldsymbol{\theta}, \boldsymbol{\theta}' \in \boldsymbol{\Theta})$.

(D-3) **Bounded variance for the stochastic gradient:** variance of the minibatch-based stochastic gradient $\tilde{g}_\eta^{(1)}(\boldsymbol{\theta})$ is uniformly bounded with respect to resampling the minibatch, i.e., $\sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \operatorname{tr} \mathbb{V}_{\mathcal{M}^{(1)}}(\tilde{g}_\eta^{(1)}(\boldsymbol{\theta})) < \infty$.

Symbols $\mathbb{E}_{\mathcal{M}^{(t)}}(\cdot), \mathbb{V}_{\mathcal{M}^{(t)}}(\cdot)$ represent the expectation and the variance covariance matrix with respect to resampling the minibatch $\mathcal{M}^{(t)}$, and $\mathbb{E}_\tau(\cdot)$ takes expectation with respect to selecting $\tau \in [T]$. $\operatorname{tr} \boldsymbol{Z}$ represents the trace of the matrix $\boldsymbol{Z} = (z_{ij}) \in \mathbb{R}^{p \times p}$, i.e., $\operatorname{tr} \boldsymbol{Z} = \sum_{i=1}^p z_{ii}$, and $\|\boldsymbol{z}\|_2 := (\sum_{i=1}^q z_i^2)^{1/2}$ for $\boldsymbol{z} = (z_1, z_2, \ldots, z_q) \in \mathbb{R}^q$.

(D-1)–(D-3) are assumed in Ghadimi and Lan (2013), and they are not unusually strong assumptions in our setting; when assuming (C-1) compactness of the parameter set $\boldsymbol{\Theta}$, $Q_\eta(\boldsymbol{\theta})$ using any generating function listed in Table 5.1 and the similarity function (5.11) equipped with vector-valued neural networks $\boldsymbol{f}_{\boldsymbol{\theta}} : \mathcal{X}^U \to \mathbb{R}^K$ activated by sigmoid function, satisfies the assumptions (D-1)–(D-2). Then, (D-3) also holds since the stochastic gradient $\tilde{g}_\eta^{(1)}(\boldsymbol{\theta})$ is C^1 on the compact set $\boldsymbol{\Theta}$ and the minibatch $\mathcal{M}^{(t)}$ is a realization of random variable taking value in a finite set.

Theorem 5.2: Convergence of SGD

Let $m_+, m_-, q, T, U \in \mathbb{N}, v \in \{0, 1, \ldots, U-1\}, \eta > 0, \boldsymbol{\Theta} := \mathbb{R}^q$, and $\{\tilde{\boldsymbol{\theta}}^{(t)}\}_{t=1}^T$ is a sequence of the minibatch SGD (5.25), and the conditions (D-1)–(D-3) are assumed. If $v \geq 1$, let $\boldsymbol{u}$ be a vector in the set (5.26), and $p_{\boldsymbol{j}} := 1/|\mathcal{K}_{\boldsymbol{u}}|$ for all $\boldsymbol{j} \in \mathcal{K}_{\boldsymbol{u}}$. By specifying $\gamma^{(t)} = \gamma t^{-1}$ with $\gamma \in (0, 2/H)$, and choosing the number of iterations $\tau \in [T]$ with the probability $\mathbb{P}(\tau = t) = \frac{2\gamma/t - H\gamma^2/t^2}{\sum_{t=1}^T (2\gamma/t - H\gamma^2/t^2)}$, it holds that

$$\mathbb{E}_\tau \left(\mathbb{E}_{\{\mathcal{M}^{(t)}\}_{t\in[\tau]}} \left(\left\| \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\tilde{\boldsymbol{\theta}}^{(\tau)}) \right\|_2^2 \right) \right) = O(1/\log T) \to 0, \quad (T \to \infty).$$

See Appendix C.3.4 for the proof.

Theorem 5.5.2 indicates that the gradient

$$\frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\tilde{\boldsymbol{\theta}}^{(\tau)}) = \frac{\partial}{\partial \boldsymbol{\theta}} D_\varphi(\{\eta w_{\boldsymbol{i}}\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}}, \{\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}}) \Big|_{\boldsymbol{\theta}=\tilde{\boldsymbol{\theta}}^{(\tau)}}$$

approaches $\mathbf{0}$ as $T \to \infty$. Considering $\lim_{T\to\infty} \mathbb{P}(\tau \leq T') = 0$ for any fixed constant $T' \in \mathbb{N}$, indicating that large τ tends to be selected when T is sufficiently large, the estimator $\tilde{\boldsymbol{\theta}}^{(t)}$ computed through the iterative update (5.25) approaches a set of stationary points of $D_\varphi(\{\eta w_{\boldsymbol{i}}\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}}, \{\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}})$ as t increases. Although the estimator can be trapped in local minimizers or saddle points during the iterative update, gradient descent using randomly perturbed gradients is proved to escape saddle points efficiently (Jin et al., 2017). The similar is expected for minibatch SGD; the estimator may approach a good minimizer efficiently, depending on the situation. When the estimator approaches a global minimizer, under some assumptions, we can expect that

$$\eta\mu_*(\boldsymbol{X}) \approx \mu_{\tilde{\boldsymbol{\theta}}^{(t)}}(\boldsymbol{X}), \quad (\forall \boldsymbol{X} \in \mathcal{X}^U) \tag{5.31}$$

for some sufficiently large $n, t \in \mathbb{N}$, by considering Theorem 5.5.1 with $\mathbb{E}(\eta w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}}) = \eta\mu_*(\boldsymbol{X}_{\boldsymbol{i}})$. Although specifying $\eta = 1$ appears better in terms of exactly recovering the underlying true similarity function μ_*, it is not necessarily so in practice; only the ratio $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})/\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}'})$ is required to infer which of the tuples $\boldsymbol{X}_{\boldsymbol{i}}, \boldsymbol{X}_{\boldsymbol{i}'}$ exhibits a stronger relation. Thus η can be arbitrarily

specified by users. In practice, we may set $s_+^{(t)} = s_-^{(t)} = 1, \eta = 1$ in (5.29), which is justified if the ratio $|\tilde{\mathcal{I}}_n^{(U)}|/|\tilde{\mathcal{P}}_n^{(U)}|$ is constant; this in effect specifies $\eta = (|\tilde{\mathcal{I}}_n^{(U)}|m_+)/(|\tilde{\mathcal{P}}_n^{(U)}|m_-)$ in (5.31) and $\gamma^{(t)}$ being multiplied by $|\tilde{\mathcal{I}}_n^{(U)}|/m_-$ in (5.25). Theorem 5.5.2 can be applied to general $U \in \mathbb{N}$, whereas only a few theoretical aspects of stochastic algorithms have been investigated even for $U = 2$ (Veitch et al., 2019).

Here, we empirically demonstrate that a stochastic optimization algorithm called Adam (Kingma and Ba, 2015) equipped with the proposed mini-batch sampling procedure shown in Algorithm 1 appropriately optimizes the similarity function within the reasonable number of iterations, in Table 5.3.

5.6 Numerical Experiments

In this section, we describe the numerical experiments that we conducted on real-world datasets. In Section 5.6.1 and 5.6.2, we employ the attributed DBLP co-authorship network dataset (Desmier et al., 2012) for performing the BHLR with $U = 2$ and $U = 3$, corresponding to link regression and hyperlink regression, respectively.

Hereinafter, we incorporate a regularization $\varphi_{\mathrm{KL}}(z) = z\log(z+\varepsilon)$ with a small constant $\varepsilon := 10^{-4}$ into the KL divergence, for numerically stabilizing the experimental results.

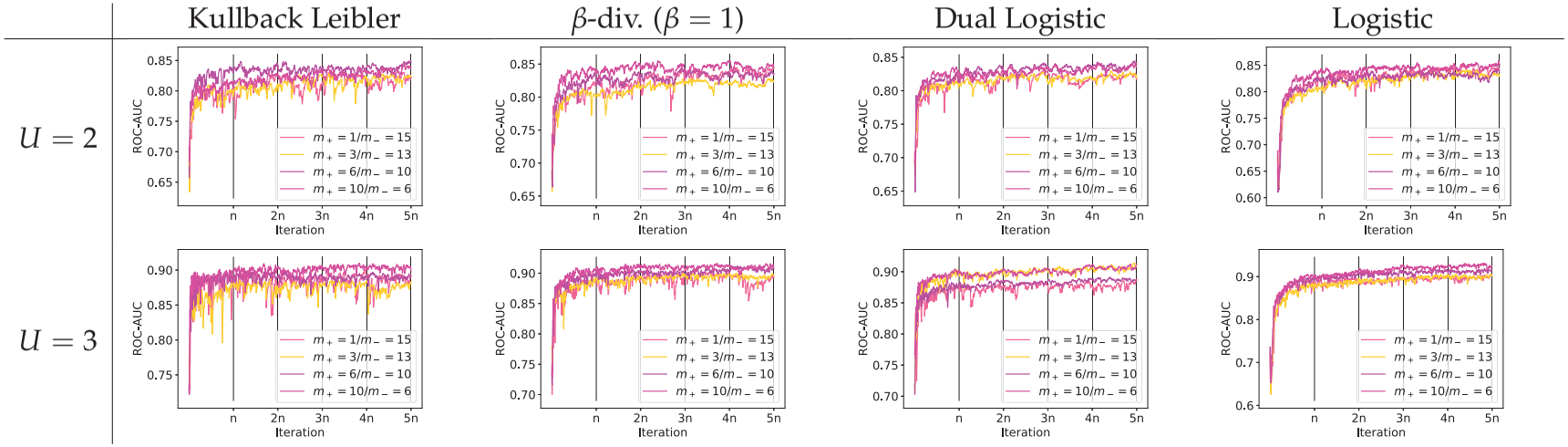
5.6.1 Link Regression ($U = 2$)

- **Dataset:** We utilize a network comprising $n = 2{,}723$ attributed nodes and 37,322 positive binary link weights, that aggregates 9 snapshots of the DBLP dynamic co-authorship network dataset (Desmier et al., 2012). In the aggregated network, each binary link weight represents whether the corresponding authors have at least one co-authorship relation in the 9 snapshots; $w_{i_1 i_2} = 1$ if the authors i_1 and i_2 have the relation, and 0 otherwise. Each node has $p = 43$ dimensional data vectors, representing the number of publications, summed up over the 9 snapshots, in each of the selected 43 journals/conferences.

- **Similarity function architecture:** Vector-valued NN $\boldsymbol{f}_{\boldsymbol{\theta}} : \mathbb{R}^{43} \to \mathbb{R}^K$ is a 1-hidden-layer multilayer perceptron with 1,000 hidden units activated by the ReLU and K unactivated output units. Using $\boldsymbol{f}_{\boldsymbol{\theta}}$, we exploit a similarity function $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i) := \sigma(\langle \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_2})\rangle)$, where $\sigma(z) := (1+\exp(-z))^{-1}$ is a sigmoid function.

- **Learning similarity functions:** NN $\boldsymbol{f}_{\boldsymbol{\theta}}$ in the similarity function is trained by Adam optimizer (Kingma and Ba, 2015) using Algorithm 1 for mini-batch sampling. For computing the stochastic gradient (5.29), we utilize $s_+^{(t)} = s_-^{(t)} = 1, \eta = 1$, and batch sizes (m_+, m_-) are selected the set $\{(1,15),(3,13),(6,10),(10,6)\}$. For each of batch sizes (m_+, m_-), the weight decay is grid searched over $\{10^{-2}, 10^{-3}\}$.

TABLE 5.3: For $U = 2, 3$, we plot the changes in the ROC–AUC test score over the Adam iterations (Kingma and Ba, 2015) using Algorithm 1 with $v = 1$, initial step size 10^{-3}, and weight decay 10^{-2}. The x-axis represents the iteration number, where n is the number of data vectors in the training dataset, and the y-axis represents the ROC–AUC test score. The results indicate that the ROC–AUC test score reaches approximately the maximum value within approximately $2n$ iterations. The experimental details are same as those in Section 5.6.1 and 5.6.2 (a) with $K = 40$.



- **Evaluation:** The set of data vectors is randomly divided into 3 non-overlapping sets for training, validation, and test, whose numbers are $n_{\text{train}} = 1,907$ (70%), $n_{\text{valid}} = 408$ (15%), $n_{\text{test}} = 408$ (15%). In the test dataset, 10 pairs are sampled from the set $\{\boldsymbol{i} = (i_1, i_2) \mid w_{\boldsymbol{i}}^{(\text{test})} = 0\}$ for each $i_1 = 1, 2, \ldots, n_{\text{test}}$, and combined with positive pairs $\{\boldsymbol{i} = (i_1, i_2) \mid w_{\boldsymbol{i}}^{(\text{test})} > 0\}$; we compute the ROC-AUC score (Bradley, 1997) using these link weights, and record the scores for each of the 50 iterations. Similarly, we compute the ROC–AUC score for the validation dataset. At the end of the iteration ($T = 3n_{\text{train}}$), we record the test score whose validation score is the best. We repeat this experiment 40 times, and compute the sample average and the standard error for each (m_+, m_-); the best validated score amongst all (m_+, m_-) is also computed.

- **Baselines:** We employ LINE (Tang et al., 2015), KL-GE representing PMvGE (Okuno, Hada, and Shimodaira, 2018) discussed in Chapter 3 with $D = 1$, and graph embedding using β-divergence, that is proposed in (Okuno and Shimodaira, 2019) and is discussed in Section 5.3.3. They correspond to the BHLR equipped with $L_{\varphi_{\text{Logistic}},n}(\boldsymbol{\theta})$, $L_{\varphi_{\text{KL}},n}(\boldsymbol{\theta})$, and $L_{\varphi_\beta,n}(\boldsymbol{\theta})$, respectively. LPP (He and Niyogi, 2004) is also conducted for obtaining the linearly transformed feature vectors $\tilde{\boldsymbol{y}}_i := \hat{\boldsymbol{A}}^\top \boldsymbol{x}_i$ for $i \in [n]$. Subsequently, similarities for the feature vectors are computed by $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) = \sigma(\langle \tilde{\boldsymbol{y}}_{i_1}, \tilde{\boldsymbol{y}}_{i_2} \rangle)$.

Results: The experimental results are shown in Table 5.4. Overall, the NN-based methods outperformed the LPP as the NN is highly expressive whereas the LPP is linear. In addition, NN-based methods demonstrated better performance by increasing the dimension K of the feature vectors, unlike the LPP that imposes a quadratic constraint on the feature vectors $\{\boldsymbol{y}_i\}_{i=1}^n$. Overall, the exponential divergence and logistic loss demonstrated good performances; particularly, the exponential divergence demonstrated the best performance among the KL divergence, β-divergence, logistic loss, dual logistic loss, and exponential divergence employed in this experiment. In terms of selecting m_+ and m_-, in this case, using more than one positive minibatch sample ($m_+ > 1$) is better.

5.6.2 Hyperlink Regression ($U = 3$)

Experimental settings are almost similar to those of $U = 2$. We employ the same dataset used in Section 5.6.1, and compute synthetic hyperlink weights from their link weights.

- **Similarity function architecture:** using $\boldsymbol{f}_{\boldsymbol{\theta}}$ defined in Section 5.6.1, we exploit a similarity function: $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) := \sigma\left(\langle \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_2}), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_3}) \rangle\right)$, where $\langle \boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}'' \rangle = \sum_{k=1}^K y_k y'_k y''_k$. Similarity functions are trained and evaluated similarly to those of $U = 2$.

- **Evaluation:** We first divide the set of data vectors into training, validation, and test sets, similarly to $U = 2$. However, these datasets contain

TABLE 5.4: Link prediction ($U = 2$) is conducted on the attributed DBLP co-authorship network dataset (Desmier et al., 2012), and the sample average and standard error of the ROC–AUC test scores for 40 experiments are listed. **A higher score is better**. Scores are multiplied by 100; the best score is **bolded**, and the second best score is underlined.

$K = 10$	Method	m_+/m_- 1/15	3/13	6/10	10/6	Best (validated)
Neural network	**BHLR + exponential div.**	**81.5** $\pm$ 0.4	82.5 $\pm$ 0.2	**82.7** $\pm$ 0.4	82.7 $\pm$ 0.3	**83.0** $\pm$ 0.4
	BHLR + dual logistic loss	80.0 $\pm$ 0.1	81.4 $\pm$ 0.2	81.7 $\pm$ 0.2	81.5 $\pm$ 0.1	81.7 $\pm$ 0.2
	KL-GE^{†,1} (Okuno, Hada, and Shimodaira, 2018)	80.1 $\pm$ 0.2	81.5 $\pm$ 0.3	82.1 $\pm$ 0.2	82.1 $\pm$ 0.2	82.2 $\pm$ 0.3
	β-GE^{†,2} (Okuno and Shimodaira, 2019) ($\beta = 0.1$)	81.4 $\pm$ 0.1	82.3 $\pm$ 0.2	82.3 $\pm$ 0.2	82.7 $\pm$ 0.2	82.3 $\pm$ 0.3
	β-GE^{†,2} (Okuno and Shimodaira, 2019) ($\beta = 0.5$)	80.6 $\pm$ 0.3	82.2 $\pm$ 0.2	82.5 $\pm$ 0.2	**82.9** $\pm$ 0.2	82.2 $\pm$ 0.3
	β-GE^{†,2} (Okuno and Shimodaira, 2019) ($\beta = 1$)	81.2 $\pm$ 0.3	82.2 $\pm$ 0.2	82.4 $\pm$ 0.3	82.4 $\pm$ 0.2	82.2 $\pm$ 0.3
	LINE^{†,3} (Tang et al., 2015)	81.4 $\pm$ 0.2	**82.6** $\pm$ 0.1	82.0 $\pm$ 0.2	82.3 $\pm$ 0.3	82.8 $\pm$ 0.2
Linear	LPP† (He and Niyogi, 2004)	78.9 $\pm$ 0.3				

$K = 40$	Method	m_+/m_- 1/15	3/13	6/10	10/6	Best (validated)
Neural network	**BHLR + exponential div.**	82.7 $\pm$ 0.2	83.4 $\pm$ 0.3	**83.8** $\pm$ 0.2	83.3 $\pm$ 0.2	**83.6** $\pm$ 0.3
	BHLR + dual logistic loss	82.2 $\pm$ 0.2	81.8 $\pm$ 0.2	82.4 $\pm$ 0.3	82.1 $\pm$ 0.2	82.2 $\pm$ 0.3
	KL-GE^{†,1} (Okuno, Hada, and Shimodaira, 2018)	82.0 $\pm$ 0.2	82.4 $\pm$ 0.2	83.1 $\pm$ 0.2	82.7 $\pm$ 0.2	82.9 $\pm$ 0.3
	β-GE^{†,2} (Okuno and Shimodaira, 2019) ($\beta = 0.1$)	81.9 $\pm$ 0.2	82.6 $\pm$ 0.2	82.7 $\pm$ 0.2	**83.5** $\pm$ 0.1	82.8 $\pm$ 0.3
	β-GE^{†,2} (Okuno and Shimodaira, 2019) ($\beta = 0.5$)	81.5 $\pm$ 0.2	82.5 $\pm$ 0.2	82.8 $\pm$ 0.2	83.1 $\pm$ 0.2	82.7 $\pm$ 0.2
	β-GE^{†,2} (Okuno and Shimodaira, 2019) ($\beta = 1$)	82.5 $\pm$ 0.3	83.3 $\pm$ 0.2	83.3 $\pm$ 0.2	83.2 $\pm$ 0.3	83.3 $\pm$ 0.2
	LINE^{†,3} (Tang et al., 2015)	**83.0** $\pm$ 0.2	**83.5** $\pm$ 0.2	83.1 $\pm$ 0.2	83.0 $\pm$ 0.2	83.4 $\pm$ 0.2
Linear	LPP† (He and Niyogi, 2004)	73.8 $\pm$ 0.4				

†Baselines, ^{1}BHLR + KL-div., ^{2}BHLR + β-div., ^{3}BHLR + logistic loss.

only the link weights ($U = 2$) but not hyperlink weights ($U = 3$); in each of the datasets, we compute synthetic hyperlink weights $\boldsymbol{W} := (w_{\boldsymbol{i}})$ in two different ways:

(a) $w_{\boldsymbol{i}} = w_{i_1 i_2 i_3} = 1$ if $\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \boldsymbol{x}_{i_3}$ are connected, i.e., a path exists between any of the two in $\boldsymbol{i} = (i_1, i_2, i_3)$, and $w_{\boldsymbol{i}} = 0$ otherwise.

(b) $w_{\boldsymbol{i}} = w_{i_1 i_2 i_3} = 1$ if $\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \boldsymbol{x}_{i_3}$ are fully connected, i.e., all of two in $\boldsymbol{i} = (i_1, i_2, i_3)$ are connected, and $w_{\boldsymbol{i}} = 0$ otherwise.

In the test dataset, 15 tuples are sampled from the set $\{\boldsymbol{i} = (i_1, i_2, i_3) \mid w_{\boldsymbol{i}}^{(\mathrm{test})} = 0\}$ for each $i_1 = 1, 2, \ldots, n_{\mathrm{test}}$, and combine them with positive tuples $\{\boldsymbol{i} = (i_1, i_2, i_3) \mid w_{\boldsymbol{i}}^{(\mathrm{test})} > 0\}$. Using these tuples, we evaluated the experimental results by ROC-AUC score, similarly to $U = 2$.

- **Baseline:** We employ HIMFAC (Nori, Bollegala, and Kashima, 2012) for obtaining the linearly transformed feature vectors $\tilde{\boldsymbol{y}}_i := \hat{\boldsymbol{A}}^\top \boldsymbol{x}_i$ $(i \in [n])$. Subsequently, similarities for the feature vectors are computed by (i) $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) := \sigma(\langle \tilde{\boldsymbol{y}}_{i_1}, \tilde{\boldsymbol{y}}_{i_2}, \tilde{\boldsymbol{y}}_{i_3} \rangle)$ and (ii) $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) := \sigma(\sum_{1 \leq k < l \leq 3} \langle \tilde{\boldsymbol{y}}_{i_k}, \tilde{\boldsymbol{y}}_{i_l} \rangle)$.

Results: The experimental results are shown in Table 5.5 for the setting (a) and Table 5.6 for (b). Overall, the NN-based methods outperformed HIMFAC, since the NN is highly expressive whereas HIMFAC is linear. NN-based methods demonstrated a slight improvement by increasing the dimension K of the feature vectors. There is significant difference between the settings (a) and (b) for HIMFAC, unlike NN-based methods. Regarding the setting (a), the logistic loss, exponential divergence and β-divergence with $\beta = 1$ demonstrated good performances for $K = 10$. On the other hand, the β-divergence with $\beta = 0.5$ and KL-divergence, whose scores for $K = 10$ were not that high, demonstrated good performance for $K = 40$; experimental results depend on the choice of K. HIMFAC with (i) demonstrates a low performance, since their feature vectors are consequently obtained via LPP, that is based on the simple inner product $\langle \boldsymbol{y}, \boldsymbol{y}' \rangle$ whereas (i) is based on the similarity for triplets $\langle \boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}'' \rangle$. On the other hand, HIMFAC with (ii) demonstrates much higher performance than (i), since HIMFAC is compatible with the simple inner product. In terms of selecting m_+ and m_-, in this case, using more than one positive minibatch sample $(m_+ > 1)$ is better. Regarding the setting (b), tendency of the results are almost similar to the setting (a).

5.7 Conclusion

In this study, we considered hyperlink weight $w_{\boldsymbol{i}}$ defined for U-tuple $\boldsymbol{X}_{\boldsymbol{i}}$ that is a collection of U data vectors $(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U})$. The hyperlink weights are assumed to be symmetric with respect to permutation of the entries $i_1, i_2, \ldots, i_U$ in the index. We proposed the BHLR that learns a user-specified symmetric similarity function $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})$ such that it predicts a tuple's hyperlink weight $w_{\boldsymbol{i}}$ through data vectors $(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U})$ stored in the corresponding U-tuple

TABLE 5.5: Hyperlink prediction ($U = 3$) with the setting (a) is conducted on the attributed DBLP co-authorship network dataset (Desmier et al., 2012), and the sample average and standard error of the ROC-AUC test scores for 40 experiments are listed. **A higher score is better**. Scores are multiplied by 100; the best score is **bolded**, and the second best score is underlined.

$K = 10$	Method	m_+/m_- 1/15	3/13	6/10	10/6	Best (validated)
Neural network	**BHLR + exponential div.**	**86.1** $\pm$ 0.3	**87.3** $\pm$ 0.3	87.5 $\pm$ 0.3	**87.2** $\pm$ 0.3	87.3 $\pm$ 0.3
	BHLR + dual logistic loss	85.4 $\pm$ 0.3	86.1 $\pm$ 0.2	86.0 $\pm$ 0.3	86.7 $\pm$ 0.3	86.2 $\pm$ 0.3
	BHLR + KL-div.	85.2 $\pm$ 0.3	85.1 $\pm$ 0.2	85.3 $\pm$ 0.3	85.6 $\pm$ 0.2	85.4 $\pm$ 0.3
	BHLR + β-div. ($\beta = 0.1$)	85.3 $\pm$ 0.3	85.5 $\pm$ 0.3	85.8 $\pm$ 0.3	85.8 $\pm$ 0.2	85.8 $\pm$ 0.3
	BHLR + β-div. ($\beta = 0.5$)	85.3 $\pm$ 0.3	86.1 $\pm$ 0.2	86.9 $\pm$ 0.3	86.3 $\pm$ 0.3	86.6 $\pm$ 0.3
	BHLR + β-div. ($\beta = 1$)	85.7 $\pm$ 0.3	86.5 $\pm$ 0.2	86.8 $\pm$ 0.3	87.0 $\pm$ 0.3	87.3 $\pm$ 0.2
	BHLR + logistic loss	86.0 $\pm$ 0.3	**87.3** $\pm$ 0.3	**87.9** $\pm$ 0.2	**87.2** $\pm$ 0.2	**87.4** $\pm$ 0.3
Linear	HIMFAC† (Nori, Bollegala, and Kashima, 2012) + (i)			48.4 $\pm$ 0.5		
	HIMFAC† (Nori, Bollegala, and Kashima, 2012) + (ii)			76.9 $\pm$ 0.3		

$K = 40$	Method	m_+/m_- 1/15	3/13	6/10	10/6	Best (validated)
Neural network	**BHLR + exponential div.**	88.7 $\pm$ 0.3	89.4 $\pm$ 0.3	88.7 $\pm$ 0.3	89.3 $\pm$ 0.3	89.8 $\pm$ 0.2
	BHLR + dual logistic loss	87.3 $\pm$ 0.3	87.8 $\pm$ 0.3	89.1 $\pm$ 0.2	88.0 $\pm$ 0.2	89.0 $\pm$ 0.3
	BHLR + KL-div.	87.9 $\pm$ 0.3	88.4 $\pm$ 0.2	**89.2** $\pm$ 0.3	89.6 $\pm$ 0.3	90.6 $\pm$ 0.2
	BHLR + β-div. ($\beta = 0.1$)	**89.3** $\pm$ 0.2	89.0 $\pm$ 0.2	89.0 $\pm$ 0.3	89.3 $\pm$ 0.3	90.4 $\pm$ 0.2
	BHLR + β-div. ($\beta = 0.5$)	88.9 $\pm$ 0.2	89.4 $\pm$ 0.2	89.6 $\pm$ 0.3	90.0 $\pm$ 0.3	**90.8** $\pm$ 0.2
	BHLR + β-div. ($\beta = 1$)	88.4 $\pm$ 0.3	**89.7** $\pm$ 0.2	89.0 $\pm$ 0.2	89.4 $\pm$ 0.2	90.5 $\pm$ 0.2
	BHLR + logistic loss	88.3 $\pm$ 0.2	88.9 $\pm$ 0.3	89.3 $\pm$ 0.2	**90.2** $\pm$ 0.2	89.9 $\pm$ 0.2
Linear	HIMFAC† (Nori, Bollegala, and Kashima, 2012) + (i)			49.6 $\pm$ 0.5		
	HIMFAC† (Nori, Bollegala, and Kashima, 2012) + (ii)			75.4 $\pm$ 0.4		

†Baselines

TABLE 5.6: Hyperlink prediction ($U = 3$) with the setting (b) is conducted on the attributed DBLP co-authorship network dataset (Desmier et al., 2012), and the sample average and standard error of the ROC-AUC test scores for 40 experiments are listed. **A higher score is better**. Scores are multiplied by 100; the best score is **bolded**, and the second best score is underlined.

$K = 10$	Method	m_+/m_- 1/15	3/13	6/10	10/6	Best (validated)
Neural network	**BHLR + exponential div.**	85.7 $\pm$ 0.3	86.6 $\pm$ 0.3	86.7 $\pm$ 0.3	86.9 $\pm$ 0.3	86.7 $\pm$ 0.3
	BHLR + dual logistic loss	85.7 $\pm$ 0.4	85.9 $\pm$ 0.4	86.0 $\pm$ 0.3	86.2 $\pm$ 0.3	86.4 $\pm$ 0.3
	BHLR + KL-div.	84.5 $\pm$ 0.4	85.0 $\pm$ 0.4	85.6 $\pm$ 0.4	85.3 $\pm$ 0.5	86.1 $\pm$ 0.5
	BHLR + β-div. ($\beta = 0.1$)	84.9 $\pm$ 0.4	85.7 $\pm$ 0.3	85.5 $\pm$ 0.3	85.8 $\pm$ 0.3	85.9 $\pm$ 0.4
	BHLR + β-div. ($\beta = 0.5$)	85.0 $\pm$ 0.4	85.7 $\pm$ 0.3	85.9 $\pm$ 0.3	86.3 $\pm$ 0.4	86.5 $\pm$ 0.4
	BHLR + β-div. ($\beta = 1$)	85.4 $\pm$ 0.4	86.0 $\pm$ 0.4	86.7 $\pm$ 0.3	86.4 $\pm$ 0.3	86.6 $\pm$ 0.3
	BHLR + logistic loss	**85.9** $\pm$ 0.3	**86.8** $\pm$ 0.3	**87.2** $\pm$ 0.3	**87.3** $\pm$ 0.3	**86.8** $\pm$ 0.3
Linear	HIMFAC† (Nori, Bollegala, and Kashima, 2012) + (i)			49.1 $\pm$ 1.3		
	HIMFAC† (Nori, Bollegala, and Kashima, 2012) + (ii)			82.6 $\pm$ 0.4		

$K = 40$	Method	m_+/m_- 1/15	3/13	6/10	10/6	Best (validated)
Neural network	**BHLR + exponential div.**	88.1 $\pm$ 0.3	89.2 $\pm$ 0.2	88.6 $\pm$ 0.3	**89.4** $\pm$ 0.2	89.7 $\pm$ 0.2
	BHLR + dual logistic loss	87.0 $\pm$ 0.3	88.1 $\pm$ 0.3	88.0 $\pm$ 0.2	87.9 $\pm$ 0.2	89.0 $\pm$ 0.2
	BHLR + KL-div.	87.7 $\pm$ 0.2	88.8 $\pm$ 0.3	89.3 $\pm$ 0.3	89.1 $\pm$ 0.3	90.0 $\pm$ 0.2
	BHLR + β-div. ($\beta = 0.1$)	**88.2** $\pm$ 0.2	89.2 $\pm$ 0.2	**89.9** $\pm$ 0.2	88.8 $\pm$ 0.3	**90.3** $\pm$ 0.1
	BHLR + β-div. ($\beta = 0.5$)	87.8 $\pm$ 0.2	**89.6** $\pm$ 0.2	88.9 $\pm$ 0.2	88.5 $\pm$ 0.3	90.1 $\pm$ 0.2
	BHLR + β-div. ($\beta = 1$)	**88.2** $\pm$ 0.3	88.5 $\pm$ 0.2	89.0 $\pm$ 0.2	88.4 $\pm$ 0.3	89.5 $\pm$ 0.2
	BHLR + logistic loss	88.1 $\pm$ 0.2	88.6 $\pm$ 0.2	88.9 $\pm$ 0.1	88.7 $\pm$ 0.2	89.3 $\pm$ 0.1
Linear	HIMFAC† (Nori, Bollegala, and Kashima, 2012) + (i)			48.6 $\pm$ 0.8		
	HIMFAC† (Nori, Bollegala, and Kashima, 2012) + (ii)			82.9 $\pm$ 0.6		

†Baselines

$\boldsymbol{X_i}$. The BHLR encompassed various existing methods such as logistic regression ($U = 1$), Poisson regression ($U = 1$), graph embedding ($U = 2$), matrix factorization ($U = 2$), stochastic block model ($U = 2$), tensor factorization ($U \geq 2$), and their variants equipped with arbitrary BD. We provided theoretical guarantees for BHLR including several existing methods, in the sense that general BHLR possessed the following two favorable properties: (P-1) statistical consistency and (P-2) computational tractability. Novel minibatch-sampling procedure for hyper-relations and theoretical guarantee for the entire stochastic optimization was also provided.

6 Conclusion

In this thesis, we discussed the neural network-based graph embedding and its extensions. In Chapter 1, we described the background of graph embedding methods, that are equipped with neural networks (NNs). Background knowledge for representation learning, including neural networks, graph embedding, and multi-view methods was described in Chapter 2. Considering the background, we posed the following three challenges, towards a theoretically-guaranteed unified framework of NN-based graph embedding.

Challenges

(1) NN-based graph embedding cannot be applied to multi-view data vectors. Although some existing methods such as CDMCA (Shimodaira, 2016) can be applied to multi-view datasets, they leverage linear models. Thus NN-based multi-view graph embedding is required.

(2) Although universal approximation theorems for standard NNs are widely developed, a neural network used in the graph embedding can be regarded as a siamese neural network (Bromley et al., 1994); only a little work sheds light on its expressive power. Theoretical analysis of the expressive power and more expressive similarities are required for improving graph embedding performance.

(3) Existing graph embedding methods are limited to considering the link weight defined between only two nodes, despite the fact that link weights can be similarly defined for a set of three or more nodes. Furthermore, objective function plays an indispensable role for hyperlink regression (including graph embedding); clarifying a class of functions, that has desirable theoretical properties in general hyperlink regression setting, is also a challenge considered in this thesis.

For solving these three challenges, we provided the following contributions.

Contributions

(i) In Chapter 3, the challenge (1) was discussed. We proposed PMvGE, a multi-view extension of NN-based graph embedding. Various existing methods are approximately included in PMvGE. We gave a practical estimation algorithm to PMvGE. Experiments on real-world datasets showed that PMvGE outperforms existing methods.

(ii) In Chapter 4, the challenge (2) was discussed. We studied the expressive power of inner-product similarity (IPS) used in NN-based graph embedding, and proved that IPS is limited to approximated positive-definite (PD) similarities. To go beyond the limitation, we proposed shifted-IPS (SIPS). SIPS is capable of approximating arbitrary conditionally PD (CPD) similarities, which is a wider class of PD similarities. We empirically evaluated the SIPS model, and we developed a more expressive inner product difference similarity (IPDS). We proved that IPDS can approximate arbitrary similarities.

(iii) In Chapter 5, the challenge (3) was discussed. We proposed Bregman Hyperlink Regression (BHLR) that learns a user-specified symmetric similarity function $\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i)$ such that it predicts a tuple's hyperlink weight w_i through data vectors $(\boldsymbol{x}_{i_1}, \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{x}_{i_U})$ stored in the corresponding U-tuple $\boldsymbol{X}_i$. The BHLR encompasses various existing methods. We provided theoretical guarantees for BHLR including several existing methods, in the sense that general BHLR possessed the following two favorable properties: (P-1) statistical consistency and (P-2) computational tractability. A novel minibatch-sampling procedure for hyper-relations and a theoretical guarantee for the entire stochastic optimization were also provided.

Considering the contributions, we further list remaining future works in the following.

Future Works

- It would be worthwhile to simultaneously learn several BHLRs with different sizes of tuples; it is straightforward to modify our method to incorporate several U values. Because a single BHLR first fixes the tuple size $U \in \mathbb{N}$, the association strengths for the different sizes of tuples cannot be measured by the similarity function. Although we empirically demonstrated the BHLR only for $U = 1, 2, 3$ in this study, a BHLR with a larger U can be conducted, and it would be natural to learn tuples with several sizes at the same time.

- For BHLR, another interesting direction is designing a better similarity function for U-tuples. Although we employed limited forms of similarity functions in our numerical experiments in the current study, arbitrary similarity functions can be employed for the BHLR; the expressive SIPS and IPDS discussed in Chapter 4 may be simply generalized to the setting of the BHLR with general $U \in \mathbb{N}$.

- We considered only the independent hyperlink weights for theories of the BHLR. Although such independent weights are theoretically tractable, in real-world situations, hyperlink weights can be dependent, especially when their corresponding tuples share some data vectors. Thus we may consider the dependent hyperlink weights.

- The last direction is finite evaluation of the expressive power of the graph embedding methods. Approximation theorems discussed in Chapter 4 assumes that the dimension K of feature vectors goes to ∞. However, in NN-based graph embedding, K is usually specified smaller than the dimension p of the original data vectors; it would be worthwhile to consider finite approximation instead. Although an approximation error bound is already provided in Theorem 4.6 for SIPS, the evaluation is not tight; better bound may be obtained under some stronger assumptions, and it may give us an intuition to design better expressive similarities.

A Appendix to Chapter 3

A.1 Linear PMvGE Approximates CDMCA

We argue an approximate relation between CDMCA and PMvGE. In the below, we will derive the solution $\hat{\boldsymbol{\theta}}_{\text{CDMCA}}$ of a slightly modified version of CDMCA, and an approximate solution $\hat{\boldsymbol{\theta}}_{\text{Apr.PMvGE}}$ of PMvGE with linear transformations. We then show that these two solutions are equivalent up to a scaling in each axis of the shared space.

Solution of a modified CDMCA

As explained in Section 2.4.2 in Chapter 2, the original CDMCA imposes a quadratic constraint

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\boldsymbol{\theta}^{(d_i)\top}\boldsymbol{x}_i\boldsymbol{x}_i^{\top}\boldsymbol{\theta}^{(d_i)} = \boldsymbol{I} \tag{A.1}$$

for maximizing the objective function (3.8). Here we replace w_{ij} in (A.1) with $\delta_{ij} = \mathbb{1}(i = j)$ so that the constraint becomes

$$\sum_{i=1}^{n} \boldsymbol{\theta}^{(d_i)\top}\boldsymbol{x}_i\boldsymbol{x}_i^{\top}\boldsymbol{\theta}^{(d_i)} = \boldsymbol{I}.$$

This modification changes the scaling in the solution, but the computation below is essentially the same as that in Shimodaira (2016). Let us define the augmented data vector, called "simple coding" (Shimodaira, 2016), $\tilde{\boldsymbol{x}}_i := (\boldsymbol{0}_{p_1}, \ldots, \boldsymbol{0}_{p_{d_i-1}}, \boldsymbol{x}_i, \boldsymbol{0}_{p_{d_i+1}}, \ldots, \boldsymbol{0}_{p_D}) \in \mathbb{R}^p$ where $p := p_1 + p_2 + \cdots + p_D$. Now, data matrix is $\tilde{\boldsymbol{X}} := (\tilde{\boldsymbol{x}}_1^{\top}, \tilde{\boldsymbol{x}}_2^{\top}, \ldots, \tilde{\boldsymbol{x}}_n^{\top})^{\top} \in \mathbb{R}^{n\times p}$, and the parameter matrix is $\boldsymbol{\theta} := (\boldsymbol{\theta}^{(1)\top}, \boldsymbol{\theta}^{(2)\top}, \ldots, \boldsymbol{\theta}^{(D)\top})^{\top} \in \mathbb{R}^{p\times K}$. With this augmented representation, the D-view embedding is now interpreted as a 1-view embedding. CDMCA maximizes the objective function

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\langle \boldsymbol{\theta}^{(d_i)\top}\boldsymbol{x}_i, \boldsymbol{\theta}^{(d_j)\top}\boldsymbol{x}_j\rangle = \operatorname{tr}\left(\boldsymbol{\theta}^{\top}\boldsymbol{H}\boldsymbol{\theta}\right) \quad (\boldsymbol{H} := \tilde{\boldsymbol{X}}^{\top}\boldsymbol{W}\tilde{\boldsymbol{X}}),$$

with respect to $\boldsymbol{\theta}$ under constraint $\boldsymbol{\theta}^{\top}\boldsymbol{G}\boldsymbol{\theta} = \boldsymbol{I}$ where $\boldsymbol{G} := \tilde{\boldsymbol{X}}^{\top}\tilde{\boldsymbol{X}}$. Let $\boldsymbol{U}_K$ be the matrix composed of the top-K eigenvectors of $\boldsymbol{G}^{-1/2}\boldsymbol{H}\boldsymbol{G}^{-1/2}$. Then the solution of the modified CDMCA is

$$\hat{\boldsymbol{\theta}}_{\text{CDMCA}} := \boldsymbol{G}^{-1/2}\boldsymbol{U}_K.$$

Approximation of linear PMvGE

MLE of PMvGE maximizes $\ell_n(\boldsymbol{\alpha}, \boldsymbol{\theta})$ defined in (3.4). Here we modify it by adding an extra term as

$$\tilde{\ell}_n(\boldsymbol{\alpha}, \boldsymbol{\theta}) := \ell_n(\boldsymbol{\alpha}, \boldsymbol{\theta}) - \frac{1}{2} \sum_{i:(d_i,d_i)\in\mathcal{D}} \mu_{\boldsymbol{\alpha},\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_i; d_i, d_i).$$

The difference approaches zero for large n, because $\frac{|\ell_n(\boldsymbol{\alpha},\boldsymbol{\theta}) - \tilde{\ell}_n(\boldsymbol{\alpha},\boldsymbol{\theta})|}{|\ell_n(\boldsymbol{\alpha},\boldsymbol{\theta})|} = O(n^{-1})$. Since the parameter $\boldsymbol{\alpha}$ is not considered in CDMCA, we assume that $\mathcal{D}$ includes all the view pairs and $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}^{(de)}), \bar{\alpha}^{(de)} \equiv \alpha_0 > 0\ (\forall d, e)$. We further assume that the transformation of PMvGE is linear: $\boldsymbol{f}_{\mathrm{NN}}^{(d)}(\boldsymbol{x}) = \boldsymbol{\theta}^{(d)\top}\boldsymbol{x}\ (\forall d)$, and data vectors in each view are centered. With this setting, we will show that the maximizer of a quadratic approximation of $\tilde{\ell}_n$ is equivalent to CDMCA.

To rewrite the likelihood function as

$$\tilde{\ell}_n(\bar{\boldsymbol{\alpha}}, \boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij}(g_{ij}(\boldsymbol{\theta})),$$

we define $g_{ij}(\boldsymbol{\theta}) := \langle \boldsymbol{\theta}^{(d_i)\top}\boldsymbol{x}_i, \boldsymbol{\theta}^{(d_j)\top}\boldsymbol{x}_j \rangle$ and $S_{ij}(g) := w_{ij} \log(\alpha_0 \exp(g)) - \alpha_0 \exp(g)$, $g \in \mathbb{R}$. Since $S_{ij}(g)$ is approximated quadratically around $g = 0$ by

$$S_{ij}^{Q}(g) = \alpha_0 \left\{ -\frac{1}{2}g^2 + \left(\frac{w_{ij}}{\alpha_0} - 1 \right) g \right\} + S_{ij}(0),$$

$\tilde{\ell}_n(\bar{\boldsymbol{\alpha}}, \boldsymbol{\theta})$ is approximated quadratically by

$$\begin{aligned}
\tilde{\ell}_n^{Q}(\boldsymbol{\theta}) &:= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij}^{Q}(g_{ij}(\boldsymbol{\theta})) \\
&= \alpha_0 \left\{ -\frac{1}{2}\mathrm{tr}\left((\boldsymbol{\theta}^\top \boldsymbol{G}\boldsymbol{\theta})^2 \right) + \frac{1}{\alpha_0}\mathrm{tr}\left(\boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} \right) - \underbrace{\mathrm{tr}\left(\boldsymbol{\theta}^\top \tilde{\boldsymbol{X}}^\top \mathbf{1}_n \mathbf{1}_n^\top \tilde{\boldsymbol{X}}\boldsymbol{\theta} \right)}_{=0\ (\because \{\boldsymbol{x}_i\} \text{ is centered.})} \right\} \\
&\qquad + \underbrace{\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij}(0)}_{\text{Const.}},
\end{aligned} \tag{A.2}$$

where $\boldsymbol{G} = \tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{X}}, \boldsymbol{H} = \tilde{\boldsymbol{X}}^\top \boldsymbol{W} \tilde{\boldsymbol{X}}$. The function $\tilde{\ell}_n^{Q}(\boldsymbol{\theta})$ has rotational degrees of freedom: $\tilde{\ell}_n^{Q}(\boldsymbol{\theta}) = \tilde{\ell}_n^{Q}(\boldsymbol{\theta}\boldsymbol{O})$ for any orthogonal matrix $\boldsymbol{O} \in \mathbb{R}^{K\times K}$. Thus, we impose an additional constraint $\boldsymbol{\theta}^\top \boldsymbol{G}\boldsymbol{\theta} = \boldsymbol{\Gamma}_K := \mathrm{diag}(\gamma_1, \gamma_2, \ldots, \gamma_K)$ for any $(\gamma_1, \ldots, \gamma_K) \in \mathbb{R}_{\geq 0}^K$. $\boldsymbol{\theta}$ satisfying this constraint is written as $\boldsymbol{\theta} = \boldsymbol{G}^{-1/2}\boldsymbol{V}_K \boldsymbol{\Gamma}_K^{1/2}$ where $\boldsymbol{V}_K \in \mathbb{R}^{P\times K}$ is a column-orthogonal matrix such that $\boldsymbol{V}_K^\top \boldsymbol{V}_K = \boldsymbol{I}$. By

substituting $\boldsymbol{\theta} = \boldsymbol{G}^{-1/2}\boldsymbol{V}_K\boldsymbol{\Gamma}_K^{1/2}$ into eq. (A.2), we have

$$\begin{aligned}\tilde{\ell}_n^Q(\boldsymbol{\theta}) &= \alpha_0 \left\{ -\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Gamma}_K^2\right) + \frac{1}{\alpha_0}\mathrm{tr}\left(\boldsymbol{\Gamma}_K \boldsymbol{S}_K\right) \right\} + \text{Const.} \\ &= \frac{\alpha_0}{2} \left\{ \left\| \frac{1}{\alpha_0}\boldsymbol{S}_K \right\|_{\mathrm{F}}^2 - \left\| \boldsymbol{\Gamma}_K - \frac{1}{\alpha_0}\boldsymbol{S}_K \right\|_{\mathrm{F}}^2 \right\} + \text{Const.}, \end{aligned} \tag{A.3}$$

where $\boldsymbol{S}_K = \boldsymbol{V}_K^\top \boldsymbol{G}^{-1/2}\boldsymbol{H}\boldsymbol{G}^{-1/2}\boldsymbol{V}_K$ and $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. This objective function is maximized when $\boldsymbol{\Gamma}_K = \frac{1}{\alpha_0}\boldsymbol{S}_K$ and $\boldsymbol{V}_K = \boldsymbol{U}_K$, because $\min_{\boldsymbol{\Gamma}_K} \|\boldsymbol{\Gamma}_K - \frac{1}{\alpha_0}\boldsymbol{S}_K\|_{\mathrm{F}}^2 = 0$ is achieved by $\boldsymbol{\Gamma}_K = \frac{1}{\alpha_0}\boldsymbol{S}_K$, and $\max_{\boldsymbol{V}_K} \|\boldsymbol{S}_K\|_{\mathrm{F}}^2$ is achieved by $\boldsymbol{V}_K = \boldsymbol{U}_K$ where $\boldsymbol{U}_K$ is the matrix composed of the top-K eigenvectors of $\boldsymbol{G}^{-1/2}\boldsymbol{H}\boldsymbol{G}^{-1/2}$. Therefore, $\tilde{\ell}_n^Q(\boldsymbol{\theta})$ is maximized by

$$\hat{\boldsymbol{\theta}}_{\text{Apr.PMvGE}} = \boldsymbol{G}^{-1/2}\boldsymbol{U}_K\boldsymbol{\Gamma}_K^{1/2}.$$

By substituting $\boldsymbol{V}_K = \boldsymbol{U}_K$ into $\boldsymbol{\Gamma}_K = \frac{1}{\alpha_0}\boldsymbol{S}_K$, we verify that $\boldsymbol{\Gamma}_K$ is a diagonal matrix with $\gamma_k := \lambda_k/\alpha_0$ where λ_k is the k-th largest eigenvalue of $\boldsymbol{G}^{-1/2}\boldsymbol{H}\boldsymbol{G}^{-1/2}$ $(k = 1, 2, \ldots, K)$.

Linear PMvGE approximates CDMCA

By comparing the two solutions, we have

$$\hat{\boldsymbol{\theta}}_{\text{Apr.PMvGE}} = \hat{\boldsymbol{\theta}}_{\text{CDMCA}}\boldsymbol{\Gamma}_K^{1/2}. \tag{A.4}$$

This simply means that each axis in the shared space is scaled by the factor $\sqrt{\gamma_k}$, $k = 1, \ldots, K$. Let $\hat{\boldsymbol{y}}$, $\hat{\boldsymbol{y}}'$ be feature vectors in the shared space computed by the approximate PMvGE with linear transformations, and $\boldsymbol{y}$, $\boldsymbol{y}'$ be feature vectors in the shared space computed by the modified CDMCA. Then the inner product is weighted in PMvGE as

$$\langle \hat{\boldsymbol{y}}, \hat{\boldsymbol{y}}' \rangle = \sum_{k=1}^{K} \hat{y}_k \hat{y}'_k = \sum_{k=1}^{K} \gamma_k y_k y'_k.$$

B Appendix to Chapter 4

B.1 Approximation Theorems for IPS and SIPS

B.1.1 Proof of Theorem 4.2 (AT for IPS)

Since $g_* : [-M', M']^{2K_*} \to \mathbb{R}$ is a positive definite kernel on a compact set, it follows from Mercer's theorem that there exist non-negative eigenvalues $\{\lambda_k\}_{k=1}^{\infty}$ and continuous eigenfunctions $\{\phi_k\}_{k=1}^{\infty}$ such that

$$g_*^{(\mathrm{PD})}(\boldsymbol{y}_*, \boldsymbol{y}'_*) = \sum_{k=1}^{\infty} \lambda_k \phi_k(\boldsymbol{y}_*)\phi_k(\boldsymbol{y}'_*), \quad \boldsymbol{y}_*, \boldsymbol{y}'_* \in [-M', M']^{K_*},$$

where the convergence is absolute and uniform (Minh, Niyogi, and Yao, 2006). The uniform convergence implies that for any $\varepsilon_1 > 0$ there exists $K_0 \in \mathbb{N}$ such that

$$\sup_{(\boldsymbol{y}_*, \boldsymbol{y}'_*) \in [-M', M']^{2K_*}} \left| g_*^{(\mathrm{PD})}(\boldsymbol{y}_*, \boldsymbol{y}'_*) - \sum_{k=1}^{K} \lambda_k \phi_k(\boldsymbol{y}_*)\phi_k(\boldsymbol{y}'_*) \right| < \varepsilon_1, \quad K \geq K_0.$$

This means $g_*^{(\mathrm{PD})}(\boldsymbol{y}_*, \boldsymbol{y}'_*) \approx \langle \boldsymbol{\Phi}_K(\boldsymbol{y}_*), \boldsymbol{\Phi}_K(\boldsymbol{y}'_*) \rangle$ for a feature map $\boldsymbol{\Phi}_K(\boldsymbol{y}_*) = (\sqrt{\lambda_k}\phi_k(\boldsymbol{y}_*))_{k=1}^{K}$.

We fix K and consider approximation of $h_k(\boldsymbol{x}) := \sqrt{\lambda_k}\phi_k(\boldsymbol{f}_*(\boldsymbol{x}))$ below. Since h_k are continuous functions on a compact set, there exists $C = C(K) > 0$ such that

$$\sup_{\boldsymbol{x} \in \mathcal{X}} |h_k(\boldsymbol{x})| < C, \quad k = 1, \ldots, K.$$

Let us write the neural networks as $\boldsymbol{f}_{\mathrm{NN}} = (f_1, f_2, \ldots, f_K)$ where $f_k : \mathbb{R}^p \to \mathbb{R}$, $k = 1, 2, \ldots, K$ are 1-hidden layer neural networks with T hidden units. $\boldsymbol{f}_{\mathrm{NN}}$ is parametrized by $\boldsymbol{\theta}$. Since h_k is continuous, h_k is approximated by the NN. For any $\varepsilon_2 > 0$, there exists a sufficiently large $T = T(K, \varepsilon_2) \in \mathbb{N}$ such that

$$\inf_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sup_{\boldsymbol{x} \in \mathcal{X}} |h_k(\boldsymbol{x}) - f_k(\boldsymbol{x})| < \varepsilon_2, \quad k = 1, \ldots, K,$$

as indicated by the universal approximation theorem (Cybenko, 1989; Telgarsky, 2017). Therefore, we have

$$\sup_{(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{X}^2} \left| g_*^{(\mathrm{PD})}\left(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')\right) - \sum_{k=1}^{K} f_k(\boldsymbol{x}) f_k(\boldsymbol{x}') \right|$$

$$
\begin{aligned}
&\leq \sup_{(x,x')\in\mathcal{X}^2}\left|g_*^{(\mathrm{PD})}\left(f_*(x),f_*(x')\right)-\sum_{k=1}^{K}h_k(x)h_k(x')\right| \\
&\quad+\sup_{(x,x')\in\mathcal{X}^2}\left|\sum_{k=1}^{K}h_k(x)\left(h_k(x')-f_k(x')\right)\right| \\
&\quad+\sup_{(x,x')\in\mathcal{X}^2}\left|\sum_{k=1}^{K}\left(h_k(x)-f_k(x)\right)f_k(x')\right| \\
&\leq \sup_{y_*,y_*'\in[-M',M']^{K_*}}\left|g_*^{(\mathrm{PD})}\left(y_*,y_*'\right)-\sum_{k=1}^{K}\lambda_k\phi_k(y_*)\phi_k(y_*')\right| \\
&\quad+\sum_{k=1}^{K}\sup_{x\in\mathcal{X}}|h_k(x)|\sup_{x'\in\mathcal{X}}\left|h_k(x')-f_k(x')\right| \\
&\quad+\sum_{k=1}^{K}\sup_{x\in\mathcal{X}}|h_k(x)-f_k(x)|\sup_{x'\in\mathcal{X}}|f_k(x')|,
\end{aligned}
$$

and applying $\inf_{\theta\in\Theta}$ to both sides yields

$$
\inf_{\theta\in\Theta}\sup_{(x,x')\in\mathcal{X}^2}\left|g_*^{(\mathrm{PD})}\left(f_*(x),f_*(x')\right)-\sum_{k=1}^{K}f_k(x)f_k(x')\right|<\varepsilon_1+KC\varepsilon_2+K\varepsilon_2(C+\varepsilon_2).
$$

By letting $\varepsilon_1=\varepsilon/2,\varepsilon_2=\min\left(C,\varepsilon/(6KC)\right)$ and $\|\cdot\|_\infty:=\sup_{(x,x')\in\mathcal{X}^2}|\cdot|$, the last formula becomes smaller than ε, thus proving

$$
\inf_{\theta\in\Theta}\left\|g_*^{(\mathrm{PD})}\left(f_*(x),f_*(x')\right)-\underbrace{\sum_{k=1}^{K}f_k(x)f_k(x')}_{=\langle f_{\mathrm{NN}}(x),f_{\mathrm{NN}}(x')\rangle}\right\|_\infty<\varepsilon
$$

with $m=m(K,\varepsilon):=KT(K,\varepsilon_2)$, for all $K\geq K_0$. □

B.1.2 Proof of Theorem 4.3 (AT for SIPS)

Since $g_*^{(\mathrm{CPD})}:\mathcal{Y}^2\to\mathbb{R}$ is a conditionally positive definite kernel on a compact set, Lemma 2.1 of Berg, Christensen, and Ressel (1984) indicates that

$$
g_0(y_*,y_*'):=g_*^{(\mathrm{CPD})}(y_*,y_*')-g_*^{(\mathrm{CPD})}(y_*,y_0)-g_*^{(\mathrm{CPD})}(y_0,y_*')+g_*^{(\mathrm{CPD})}(y_0,y_0)
$$

is positive definite for arbitrary $y_0\in\mathcal{Y}$. We fix y_0 in the argument below. According to Theorem 4.2 in this thesis, for any $\varepsilon>0$, there exists $K_0=K_0(\varepsilon)\in\mathbb{N}$ such that

$$
\inf_{\theta\in\Theta}\left\|g_0\left(f_*(x),f_*(x')\right)-\langle f_{\mathrm{NN}}(x),f_{\mathrm{NN}}(x')\rangle\right\|_\infty<\varepsilon/3,
$$

with sufficiently large $m_f=m_f(K,\varepsilon)\in\mathbb{N}$, for all $K\geq K_0$. $f_{\mathrm{NN}}(x):\mathbb{R}^p\to\mathbb{R}^K$ is 1-hidden layer NN having m_f hidden units and parametrized by θ. Next,

let us consider a continuous function $u_*(\boldsymbol{x}) := g_*(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{y}_0) - \frac{1}{2} g_*(\boldsymbol{y}_0, \boldsymbol{y}_0)$. The universal approximation theorem (Cybenko, 1989; Yarotsky, 2018) indicates that, for any $\varepsilon > 0$, by specifying sufficiently large $m_u = m_u(\varepsilon) \in \mathbb{N}$, we have

$$\inf_{\boldsymbol{\psi} \in \boldsymbol{\Psi}} \|u_*(\boldsymbol{x}) - u_{\mathrm{NN}}(\boldsymbol{x})\|_\infty < \varepsilon/3,$$

where u_{NN} is 1-hidden layer NN having m_u hidden units and parametrized by $\boldsymbol{\psi}$. Therefore, we have

$$\begin{aligned}
&\inf_{\substack{\boldsymbol{\theta} \in \boldsymbol{\Theta} \\ \boldsymbol{\psi} \in \boldsymbol{\Psi}}} \left\| g_*^{(\mathrm{CPD})}\left(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')\right) - \left\{ \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle + u_{\mathrm{NN}}(\boldsymbol{x}) + u_{\mathrm{NN}}(\boldsymbol{x}') \right\} \right\|_\infty \\
&= \inf_{\substack{\boldsymbol{\theta} \in \boldsymbol{\Theta} \\ \boldsymbol{\psi} \in \boldsymbol{\Psi}}} \Big\| \left(g_0\left(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')\right) - \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle \right) \\
&\qquad\qquad + \left(u_*(\boldsymbol{x}) - u_{\mathrm{NN}}(\boldsymbol{x})\right) + \left(u_*(\boldsymbol{x}') - u_{\mathrm{NN}}(\boldsymbol{x}')\right) \Big\|_\infty \\
&\leq \inf_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left\| \left(g_0\left(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')\right) - \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle \right) \right\|_\infty \\
&\qquad + \inf_{\boldsymbol{\psi} \in \boldsymbol{\Psi}} \left\| u_*(\boldsymbol{x}) - u_{\mathrm{NN}}(\boldsymbol{x}) \right\|_\infty + \inf_{\boldsymbol{\psi} \in \boldsymbol{\Psi}} \left\| u_*(\boldsymbol{x}') - u_{\mathrm{NN}}(\boldsymbol{x}') \right\|_\infty \\
&< \varepsilon/3 + 2\varepsilon/3 = \varepsilon
\end{aligned}$$

for all $K \geq K_0$. Thus proving the assertion.

□

B.1.3 Proof of Theorem 4.4 (AT for C-SIPS)

With fixed $\boldsymbol{y}_0 \in \mathcal{Y}$, it follows from Berg, Christensen, and Ressel (1984) Lemma 2.1 and CPD-ness of the kernel $g_*^{(\mathrm{CPD})}$ that

$$g_0(\boldsymbol{y}, \boldsymbol{y}') := g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}') - g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}_0) - g_*^{(\mathrm{CPD})}(\boldsymbol{y}_0, \boldsymbol{y}') + g_*^{(\mathrm{CPD})}(\boldsymbol{y}_0, \boldsymbol{y}_0)$$

is PD. Since $\mathcal{Y}$ is compact, $\sup_{\boldsymbol{y} \in \mathcal{Y}} |g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}_0)| = a^2$ is bounded. Let us take a sufficiently large $r > a$ and define $\tau(\boldsymbol{y}) := \sqrt{r^2 + g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}_0)}$. We consider a new kernel

$$g_1(\boldsymbol{y}, \boldsymbol{y}') := g_0(\boldsymbol{y}, \boldsymbol{y}') + 2\tau(\boldsymbol{y})\tau(\boldsymbol{y}').$$

Since both $g_0(\boldsymbol{y}, \boldsymbol{y}')$ and $\tau(\boldsymbol{y})\tau(\boldsymbol{y}')$ are PD, $g_1(\boldsymbol{y}, \boldsymbol{y}')$ is also PD. Applying Taylor's expansion $\sqrt{1+x} = 1 + x/2 + O(x^2)$, we have

$$\begin{aligned}
\tau(\boldsymbol{y})\tau(\boldsymbol{y}') &= \sqrt{r^2 + g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}_0)} \sqrt{r^2 + g_*^{(\mathrm{CPD})}(\boldsymbol{y}', \boldsymbol{y}_0)} \\
&= r^2 \sqrt{1 + g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}_0)/r^2} \sqrt{1 + g_*^{(\mathrm{CPD})}(\boldsymbol{y}', \boldsymbol{y}_0)/r^2} \\
&= r^2 (1 + g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}_0)/2r^2 + O(r^{-4}))(1 + g_*^{(\mathrm{CPD})}(\boldsymbol{y}', \boldsymbol{y}_0)/2r^2 + O(r^{-4}))
\end{aligned}$$

$$= r^2 + \frac{1}{2}(g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}_0) + g_*^{(\mathrm{CPD})}(\boldsymbol{y}', \boldsymbol{y}_0)) + O(r^{-2}),$$

thus proving

$$g_1(\boldsymbol{y}, \boldsymbol{y}') = g_0(\boldsymbol{y}, \boldsymbol{y}') + 2\tau(\boldsymbol{y})\tau(\boldsymbol{y}') = g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}') + g_*^{(\mathrm{CPD})}(\boldsymbol{y}_0, \boldsymbol{y}_0) + 2r^2 + O(r^{-2}).$$

Let us define $\gamma := g_*^{(\mathrm{CPD})}(\boldsymbol{y}_0, \boldsymbol{y}_0) + 2r^2 = O(r^2)$. As the kernel $g_1(\boldsymbol{y}, \boldsymbol{y}') = g_*^{(\mathrm{CPD})}(\boldsymbol{y}, \boldsymbol{y}') + \gamma + O(r^{-2})$ is PD, AT for IPS can be employed. For any $\varepsilon > 0$, there exists $K_0 = K_0(\varepsilon) \in \mathbb{N}$ such that

$$\begin{aligned}
&\inf_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left\| g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) - \left(\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle - \gamma \right) \right\|_\infty \\
&= \inf_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left\| g_1(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) - \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle \right\|_\infty + O(r^{-2}) \\
&= \varepsilon + O(r^{-2})
\end{aligned}$$

with sufficiently large $m = m(K, \varepsilon) \in \mathbb{N}$, for all $K \geq K_0$. Thus proving the assertion.

□

B.2 Approximation Error Rate

In Sections B.2.2 and B.2.3, we prove Theorems 4.5 and 4.6, which indicate the approximation error rates for IPS and SIPS, respectively. For showing the error rates, we begin with describing the error rate for Mercer's theorem, in the following Section B.2.1.

B.2.1 Error rate for Mercer's theorem

We evaluate the error rate for Mercer's theorem (shown as Theorem 4.1 in this thesis) to approximate PD kernels g_* satisfying conditions (C-1) and (C-2) of Section 4.5.

We define the error rate for Mercer's theorem as

$$\varepsilon_1(K) := \sup_{\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}} \left| g_*(\boldsymbol{y}, \boldsymbol{y}') - \sum_{k=1}^{K} \lambda_k \phi_k(\boldsymbol{y}) \phi_k(\boldsymbol{y}') \right|. \tag{B.1}$$

Then, the error rate is given in the lemma below.

Lemma B.1: Error rate for kernels

For compact set $\mathcal{Y} \subset \mathbb{R}^{K^*}$, $K^* \in \mathbb{N}$, we consider a PD kernel $g_* : \mathcal{Y}^2 \to \mathbb{R}$ which satisfies conditions (C-1) and (C-2). Then, $\varepsilon_1(K) = O(K^{-1/K^*})$.

For proving the lemma, we first show a result of the decay rate for eigenvalues. The theorem below is a special case of Theorem 4 of Cobos and Kühn (1990) by assuming μ as Lebesgue measure, and $\mathbf{\Omega} = \mathcal{Y}$.

Theorem B.1: Cobos and Kühn (1990)

Let $\mathcal{Y} \subset \mathbb{R}^L$ be a non-empty compact set for $L \in \mathbb{N}$, and let $g : \mathcal{Y}^2 \to \mathbb{R}$ be a positive definite kernel satisfying $\int_{\mathcal{Y}} \|g(\boldsymbol{t}, \cdot)\|_{C^\alpha} \mathrm{d}\boldsymbol{t} < \infty$, where $0 < \alpha \leq 1$ and

$$\|g(\boldsymbol{t}, \cdot)\|_{C^\alpha} := \max \left\{ \sup_{\boldsymbol{y} \in \mathcal{Y}} |g(\boldsymbol{t}, \boldsymbol{y})|, \sup_{\substack{\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y} \\ \boldsymbol{y} \neq \boldsymbol{y}'}} \frac{|g(\boldsymbol{t}, \boldsymbol{y}) - g(\boldsymbol{t}, \boldsymbol{y}')|}{\|\boldsymbol{y} - \boldsymbol{y}'\|_2^\alpha} \right\}.$$

Then, the k-th largest eigenvalue of g is

$$\lambda_k = O(k^{-1-\alpha/L}).$$

We apply Theorem B.1 to g_* by letting $L = K^*$ and $\alpha = 1$. Then the eigenvalues of g_* satisfy

$$\lambda_k = O(k^{-1-1/K^*}), \tag{B.2}$$

where the condition of g in Theorem B.1 will be verified later. On the other hand, Mercer's theorem and the condition (C-1) lead to

$$\begin{aligned} \varepsilon_1(K) &= \sup_{\boldsymbol{y}, \boldsymbol{y}' \in \mathcal{Y}} \left| \sum_{k=K+1}^{\infty} \lambda_k \phi_k(\boldsymbol{y}) \phi_k(\boldsymbol{y}') \right| \leq \sum_{k=K+1}^{\infty} \lambda_k \sup_{\boldsymbol{y} \in \mathcal{Y}, l \in \mathbb{N}} |\phi_l(\boldsymbol{y})| \sup_{\boldsymbol{y}' \in \mathcal{Y}, l' \in \mathbb{N}} |\phi_{l'}(\boldsymbol{y}')| \\ &= \left(\sup_{\boldsymbol{y} \in \mathcal{Y}, k \in \mathbb{N}} |\phi_k(\boldsymbol{y})| \right)^2 \sum_{k=K+1}^{\infty} \lambda_k = O\left(\sum_{k=K+1}^{\infty} \lambda_k \right). \end{aligned} \tag{B.3}$$

Therefore, substituting (B.2) into (B.3), we have

$$\begin{aligned} \varepsilon_1(K) &= O\left(\sum_{k=K+1}^{\infty} \lambda_k \right) = O\left(\int_K^{\infty} k^{-1-1/K^*} \mathrm{d}k \right) = O\left(\left[-K^* k^{-1/K^*} \right]_K^{\infty} \right) \\ &= O(K^{-1/K^*}). \end{aligned}$$

This proves Lemma B.1. Finally, we verify that g_* satisfies the condition of g in Theorem B.1. As g_* is continuous on compact set,

$$\sup_{\boldsymbol{t} \in \mathcal{Y}} \sup_{\boldsymbol{y} \in \mathcal{Y}} |g_*(\boldsymbol{t}, \boldsymbol{y})| < \infty \tag{B.4}$$

obviously holds, and the condition (C-2) implies α-Hölder continuity, and so

$$\sup_{t\in\mathcal{Y}} \sup_{\substack{y,y'\in\mathcal{Y}\\ y\neq y'}} \frac{|g_*(t,y)-g_*(t,y')|}{\|y-y'\|_2} < \infty. \tag{B.5}$$

Inequalities (B.4) and (B.5) lead to

$$\begin{aligned}\sup_{t\in\mathcal{Y}} \|g_*(t,\cdot)\|_{C^1} &\leq \max\left\{ \sup_{t\in\mathcal{Y}}\sup_{y\in\mathcal{Y}} |g_*(t,y)|, \sup_{t\in\mathcal{Y}} \sup_{\substack{y,y'\in\mathcal{Y}\\ y\neq y'}} \frac{|g_*(t,y)-g_*(t,y')|}{\|y-y'\|_2} \right\} \\ &< \infty.\end{aligned}$$

Thus g_* satisfies

$$\int_{\mathcal{Y}} \|g_*(t,\cdot)\|_{C^1} \mathrm{d}t \leq \sup_{t\in\mathcal{Y}} \|g_*(t,\cdot)\|_{C^1} \int_{\mathcal{Y}} \mathrm{d}t < \infty,$$

because compact set $\mathcal{Y} \subset \mathbb{R}^{K^*}$ is bounded and closed. □

B.2.2 Proof of Theorem 4.5 (Approximation error rate for IPS)

Applying Theorem 4.1 to a PD kernel $g_*^{(\mathrm{PD})}$, there exist eigenvalues $\{\lambda_k\}_{k=1}^{\infty}$, $\lambda_1 \geq \lambda_2 \geq \cdots$ and eigenfunctions $\{\phi_k(y)\}_{k=1}^{\infty}$ such that $\sum_{k=1}^{K} \lambda_k \phi_k(y)\phi_k(y')$ absolutely and uniformly converges to $g_*^{(\mathrm{PD})}(y,y')$ as $K \to \infty$. Here, we define two vector-valued functions

$$\begin{aligned}\eta_K(y) &:= (\lambda_1^{1/2}\phi_1(y), \lambda_2^{1/2}\phi_2(y), \ldots, \lambda_K^{1/2}\phi_K(y)), \\ \tilde{\phi}_K(x) &:= \eta_K(f_*(x)),\end{aligned}$$

so that $\langle \eta_K(f_*(x)), \eta_K(f_*(x'))\rangle = \langle \tilde{\phi}_K(x), \tilde{\phi}_K(x')\rangle = \sum_{k=1}^{K} \lambda_k \phi_k(f_*(x))\phi_k(f_*(x'))$. Using these functions, for any $f_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_f, K)$, we have

$$\begin{aligned}&\left| g_*^{(\mathrm{PD})}(f_*(x), f_*(x')) - \langle f_{\mathrm{NN}}(x), f_{\mathrm{NN}}(x')\rangle \right| \\ &\quad\leq \left| g_*(f_*(x), f_*(x')) - \langle \eta_K(f_*(x)), \eta_K(f_*(x'))\rangle \right| && \text{(B.6)} \\ &\qquad + \left| \langle \tilde{\phi}_K(x), \tilde{\phi}_K(x')\rangle - \langle \tilde{\phi}_K(x), f_{\mathrm{NN}}(x')\rangle \right| && \text{(B.7)} \\ &\qquad + \left| \langle \tilde{\phi}_K(x), f_{\mathrm{NN}}(x')\rangle - \langle f_{\mathrm{NN}}(x), f_{\mathrm{NN}}(x')\rangle \right|. && \text{(B.8)}\end{aligned}$$

These terms (B.6)–(B.8) can be evaluated in the following way.

- Regarding the term (B.6),

$$\begin{aligned}
&\sup_{x,x'\in\mathcal{X}}\left|g_*^{(\mathrm{PD})}(f_*(x),f_*(x'))-\langle\eta_K(f_*(x)),\eta_K(f_*(x'))\rangle\right| \\
&\quad\le \sup_{y,y'\in\mathcal{Y}}\left|g_*^{(\mathrm{PD})}(y,y')-\langle\eta_K(y),\eta_K(y')\rangle\right| \\
&\quad= \sup_{y,y'\in\mathcal{Y}}\left|g_*^{(\mathrm{PD})}(y,y')-\sum_{k=1}^{K}\lambda_k\phi_k(y)\phi_k(y')\right| = O(K^{-1/K^*}),
\end{aligned}$$

where the last formula follows by applying Lemma B.1 to $g_*^{(\mathrm{PD})}$. Thus, we have

$$\begin{aligned}
\inf_{f_{\mathrm{NN}}\in\mathfrak{S}_\alpha(W_f,K)}\sup_{x,x'\in\mathcal{X}}\Big|&g_*^{(\mathrm{PD})}(f_*(x),f_*(x')) \\
&-\langle\eta_K(f_*(x)),\eta_K(f_*(x'))\rangle\Big| = O(K^{-1/K^*}).
\end{aligned}\tag{B.9}$$

- Regarding the terms (B.7) and (B.8),

$$\begin{aligned}
&\sup_{x,x'\in\mathcal{X}}\left\{\left|\langle\tilde{\boldsymbol{\phi}}_K(x),\tilde{\boldsymbol{\phi}}_K(x')\rangle-\langle\tilde{\boldsymbol{\phi}}_K(x),f_{\mathrm{NN}}(x')\rangle\right|\right. \\
&\qquad\left.+\left|\langle\tilde{\boldsymbol{\phi}}_K(x),f_{\mathrm{NN}}(x')\rangle-\langle f_{\mathrm{NN}}(x),f_{\mathrm{NN}}(x')\rangle\right|\right\} \\
&\le \sup_{x,x'\in\mathcal{X}}\left\{\|\tilde{\boldsymbol{\phi}}_K(x)\|_2\|\tilde{\boldsymbol{\phi}}_K(x')-f_{\mathrm{NN}}(x')\|_2+\|f_{\mathrm{NN}}(x')\|_2\|\tilde{\boldsymbol{\phi}}_K(x)-f_{\mathrm{NN}}(x)\|_2\right\} \\
&\le \sup_{x,x'\in\mathcal{X}}\left\{\|\tilde{\boldsymbol{\phi}}_K(x)\|_2\|\tilde{\boldsymbol{\phi}}_K(x')-f_{\mathrm{NN}}(x')\|_2\right. \\
&\qquad\left.+(\|\tilde{\boldsymbol{\phi}}_K(x')\|_2+\|\tilde{\boldsymbol{\phi}}_K(x')-f_{\mathrm{NN}}(x')\|_2)\|\tilde{\boldsymbol{\phi}}_K(x)-f_{\mathrm{NN}}(x)\|_2\right\} \\
&= 2\sup_{x\in\mathcal{X}}\|\tilde{\boldsymbol{\phi}}_K(x)\|_2\sup_{x'\in\mathcal{X}}\|\tilde{\boldsymbol{\phi}}_K(x')-f_{\mathrm{NN}}(x')\|_2+\sup_{x\in\mathcal{X}}\|\tilde{\boldsymbol{\phi}}_K(x)-f_{\mathrm{NN}}(x)\|_2^2.
\end{aligned}$$

Here,

$$\begin{aligned}
\|\tilde{\boldsymbol{\phi}}_K(x)\|_2 &= \|\sum_{k=1}^{K}\lambda_k\phi_k(f_*(x))\phi_k(f_*(x))\|_2 \\
&\le \|\sum_{k=1}^{\infty}\lambda_k\phi_k(f_*(x))\phi_k(f_*(x))\|_2 \\
&= \|g_*^{(\mathrm{PD})}(f_*(x),f_*(x))\|_2
\end{aligned}$$

is bounded, because $g_*^{(\mathrm{PD})}(f_*(x),f_*(x))$ is continuous over the compact set $\mathcal{X}^2$. For applying Lemma 2.1 to $\tilde{\boldsymbol{\phi}}_K(x)$, we need to show that the

constant b exists. Noting

$$\|\partial\tilde{\phi}_k/\partial \boldsymbol{x}\|_2^2 = \sum_{i=1}^{p}(\partial\tilde{\phi}_k/\partial x_i)^2 \leq \sum_{i=1}^{p}\lambda_k\|\partial\phi_k/\partial \boldsymbol{y}\|_2^2\,\|\partial \boldsymbol{f}_*/\partial x_i\|_2^2, \tag{B.10}$$

we have

$$\begin{aligned}\sup_{k\in\mathbb{N}}\sup_{\boldsymbol{x}\in\mathcal{X}}\|\partial\tilde{\phi}_k/\partial \boldsymbol{x}\|_2^2 &\leq \sup_{k\in\mathbb{N}}\sup_{\boldsymbol{y}\in\mathcal{Y}}\lambda_k\|\partial\phi_k/\partial \boldsymbol{y}\|_2^2 \\ &\quad\times \sup_{\boldsymbol{x}\in\mathcal{X}}\sum_{i=1}^{p}\|\partial \boldsymbol{f}_*/\partial x_i\|_2^2 \\ &< \infty,\end{aligned} \tag{B.11}$$

where $\sup_{k\in\mathbb{N}}\sup_{\boldsymbol{y}\in\mathcal{Y}}\lambda_k\|\partial\phi_k/\partial \boldsymbol{y}\|_2^2 < \infty$ and $\sup_{\boldsymbol{x}\in\mathcal{X}}\sum_{i=1}^{p}\|\partial \boldsymbol{f}_*/\partial x_i\|_2^2 < \infty$ follow from (C-1) and (C-3), respectively. We can take b^2 as the upper bound of (B.11), and then Lemma 2.1 implies

$$\inf_{f_{\mathrm{NN}}\in\mathfrak{S}_\alpha(W_f,K)}\sup_{\boldsymbol{x}\in\mathcal{X}}\|\tilde{\boldsymbol{\phi}}_K(\boldsymbol{x}) - \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x})\|_2 = O(K^{\frac{1}{2}+\frac{1+\alpha}{p}}W_f^{-\frac{1+\alpha}{p}}) \tag{B.12}$$

so that the evaluation of (B.7) leads to

$$\begin{aligned}&\inf_{f_{\mathrm{NN}}\in\mathfrak{S}_\alpha(W_f,K)}\sup_{\boldsymbol{x},\boldsymbol{x}'\in\mathcal{X}}\left\{\left|\langle\tilde{\boldsymbol{\phi}}_K(\boldsymbol{x}),\tilde{\boldsymbol{\phi}}_K(\boldsymbol{x}')\rangle - \langle\tilde{\boldsymbol{\phi}}_K(\boldsymbol{x}),\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\rangle\right|\right. \\ &\qquad\left.+\left|\langle\tilde{\boldsymbol{\phi}}_K(\boldsymbol{x}),\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\rangle - \langle\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}),\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\rangle\right|\right\} \\ &= O(K^{\frac{1}{2}+\frac{1+\alpha}{p}}W_f^{-\frac{1+\alpha}{p}}).\end{aligned} \tag{B.13}$$

Considering (B.9) and (B.13), we finally obtain

$$\begin{aligned}&\inf_{f_{\mathrm{NN}}\in\mathfrak{S}_\sigma(W_f,K)}\sup_{\boldsymbol{x},\boldsymbol{x}'\in\mathcal{X}}\left|g_*^{(\mathrm{PD})}(\boldsymbol{f}_*(\boldsymbol{x}),\boldsymbol{f}_*(\boldsymbol{x}')) - \langle\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}),\boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}')\rangle\right| \\ &= O\Big(K^{-\frac{1}{K^*}} + K^{\frac{1}{2}+\frac{1+\alpha}{p}}W_f^{-\frac{1+\alpha}{p}}\Big).\end{aligned}$$

□

B.2.3 Proof of Theorem 4.6 (Approximation error rate for SIPS)

By recalling the decomposition (4.12) of the CPD kernel $g_*^{(\mathrm{CPD})}$, we first define a function

$$u_*(\boldsymbol{x}) := g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}),\boldsymbol{y}_0) - \frac{1}{2}g_*^{(\mathrm{CPD})}(\boldsymbol{y}_0,\boldsymbol{y}_0),$$

so that $g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) = g_0(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) + u_*(\boldsymbol{x}) + u_*(\boldsymbol{x}')$, and $u_*(\boldsymbol{x})$ is twice differentiable. Then, it holds that

$$\begin{aligned}
&\sup_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}} \left| g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) - (\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle + u_{\mathrm{NN}}(\boldsymbol{x}) + u_{\mathrm{NN}}(\boldsymbol{x}')) \right| \\
&\quad \leq \sup_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}} \left| g_0(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) - \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle \right| \\
&\qquad\qquad + 2 \sup_{\boldsymbol{x} \in \mathcal{X}} \left| u_*(\boldsymbol{x}) - u_{\mathrm{NN}}(\boldsymbol{x}) \right| \qquad \text{(B.14)}
\end{aligned}$$

We evaluate the two terms in (B.14). Since we have assumed that $g_*^{(\mathrm{CPD})}$ is C^1 (the condition C-2), g_0 and u_* are also C^1. Then, by applying Theorem 4.5 to the PD kernel g_0, the first term in (B.14) is evaluated as

$$\begin{aligned}
&\inf_{\boldsymbol{f}_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_f, K)} \sup_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}} \left| g_0(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) - \langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle \right| \\
&\qquad = O\Big(K^{-\frac{1}{K^*}} + K^{\frac{1}{2} + \frac{1+\alpha}{p}} W_f^{-\frac{1+\alpha}{p}} \Big). \qquad \text{(B.15)}
\end{aligned}$$

By applying UAT for real-valued NN (Theorem 2.1 in Chapter 2) to u_*, the second term in (B.14) is evaluated as

$$\inf_{u_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_u, 1)} \sup_{\boldsymbol{x} \in \mathcal{X}} \left| u_*(\boldsymbol{x}) - u_{\mathrm{NN}}(\boldsymbol{x}) \right| = O\Big(W_u^{-\frac{1+\alpha}{p}} \Big). \qquad \text{(B.16)}$$

Considering (B.14), (B.15) and (B.16), we obtain

$$\begin{aligned}
&\inf_{\substack{\boldsymbol{f}_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_f, K) \\ u_{\mathrm{NN}} \in \mathfrak{S}_\alpha(W_u, 1)}} \sup_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}} \Big| g_*^{(\mathrm{CPD})}(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}')) \\
&\qquad\qquad - (\langle \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}_{\mathrm{NN}}(\boldsymbol{x}') \rangle + u_{\mathrm{NN}}(\boldsymbol{x}) + u_{\mathrm{NN}}(\boldsymbol{x}')) \Big| \\
&\quad = O\Big(K^{-\frac{1}{K^*}} + K^{\frac{1}{2} + \frac{1+\alpha}{p}} W_f^{-\frac{1+\alpha}{p}} + W_u^{-\frac{1+\alpha}{p}} \Big).
\end{aligned}$$

□

B.3 Approximation Theorem for IPDS

B.3.1 Proof of Theorem 4.7 (AT for IPDS; Arbitrary)

By following Section 4.7.3, the similarity $h_*(\boldsymbol{x}, \boldsymbol{x}') := g_*(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}'))$ is decomposed as

$$h_*(\boldsymbol{x}, \boldsymbol{x}') = \lim_{K_+, K_-} \left\{ \langle \tilde{\boldsymbol{\Phi}}^+_{K_+}(\boldsymbol{x}), \tilde{\boldsymbol{\Phi}}^+_{K_+}(\boldsymbol{x}') \rangle - \langle \tilde{\boldsymbol{\Phi}}^-_{K_-}(\boldsymbol{x}), \tilde{\boldsymbol{\Phi}}^-_{K_-}(\boldsymbol{x}') \rangle \right\},$$

where $\tilde{\boldsymbol{\Phi}}^+_{K_+}, \tilde{\boldsymbol{\Phi}}^-_{K_-}$ are $\mathbb{R}^{K_+}, \mathbb{R}^{K_-}$-valued continuous functions, respectively, and the convergence is in the sense of L^2-norm. By virtue of the UAT for NN, the functions $\tilde{\boldsymbol{\Phi}}^+_{K_+}, \tilde{\boldsymbol{\Phi}}^-_{K_-}$ can be approximated by the NNs $\boldsymbol{f}^+_{\mathrm{NN}} : \mathbb{R}^p \to \mathbb{R}^{K_+}, \boldsymbol{f}^-_{\mathrm{NN}} : \mathbb{R}^p \to \mathbb{R}^{K_-}$ having m_+, m_- hidden units. They are parametrized by $\boldsymbol{\theta}^+, \boldsymbol{\theta}^-$, respectively.

Therefore, by following the same procedure as the proof of AT for IPS, for any $\varepsilon > 0$, there exist $K_{+0} = K_{+0}(\varepsilon), K_{-0} = K_{-0}(\varepsilon) \in \mathbb{N}$ such that

$$\begin{aligned}
\delta_1 &:= \left\| h_*(\boldsymbol{x}, \boldsymbol{x}') - \left(\langle \tilde{\boldsymbol{\Phi}}^+_{K_+}(\boldsymbol{x}), \tilde{\boldsymbol{\Phi}}^+_{K_+}(\boldsymbol{x}') \rangle \right.\right. \\
&\qquad \left.\left. - \langle \tilde{\boldsymbol{\Phi}}^-_{K_-}(\boldsymbol{x}), \tilde{\boldsymbol{\Phi}}^-_{K_-}(\boldsymbol{x}') \rangle \right) \right\|_2 < \varepsilon/3, \\
\delta_2 &:= \inf_{\boldsymbol{\theta}^+ \in \boldsymbol{\Theta}^+} \left\| \langle \tilde{\boldsymbol{\Phi}}^+_{K_+}(\boldsymbol{x}), \tilde{\boldsymbol{\Phi}}^+_{K_+}(\boldsymbol{x}') \rangle - \langle \boldsymbol{f}^+_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}^+_{\mathrm{NN}}(\boldsymbol{x}') \rangle \right\|_\infty < \varepsilon/3, \\
\delta_3 &:= \inf_{\boldsymbol{\theta}^- \in \boldsymbol{\Theta}^-} \left\| \langle \tilde{\boldsymbol{\Phi}}^-_{K_-}(\boldsymbol{x}), \tilde{\boldsymbol{\Phi}}^-_{K_-}(\boldsymbol{x}') \rangle - \langle \boldsymbol{f}^-_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}^-_{\mathrm{NN}}(\boldsymbol{x}') \rangle \right\|_\infty < \varepsilon/3,
\end{aligned}$$

with sufficiently large $m_+ = m_+(K_+, \varepsilon), m_- = m_-(K_-, \varepsilon) \in \mathbb{N}$, for all $K_+ \geq K_{+0}, K_- \geq K_{-0}$. Considering the above inequalities, we have

$$\begin{aligned}
&\inf_{\substack{\boldsymbol{\theta}^+ \in \boldsymbol{\Theta}+ \\ \boldsymbol{\theta}^- \in \boldsymbol{\Theta}^-}} \| h_*(\boldsymbol{x}, \boldsymbol{x}') - \left(\left\langle \boldsymbol{f}^+_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}^+_{\mathrm{NN}}(\boldsymbol{x}') \right\rangle - \left\langle \boldsymbol{f}^-_{\mathrm{NN}}(\boldsymbol{x}), \boldsymbol{f}^-_{\mathrm{NN}}(\boldsymbol{x}') \right\rangle \right) \|_2 \\
&\qquad \leq \delta_1 + \delta_2 + \delta_3 < \varepsilon/3 + \varepsilon/3 + \varepsilon/3 = \varepsilon.
\end{aligned}$$

Thus proving the assertion. □

C Appendix to Chapter 5

C.1 Synthetic Dataset for Figure 5.5

For generating $n = 200$ data vectors $\{\boldsymbol{x}_i\}_{i=1}^{200}$ of $p = 20$ dimensions, we first prepared centers of clusters $\boldsymbol{x}_k^{(0)} \in \mathbb{R}^5$, $k = 1, \ldots, 4$, and a linear transformation $\boldsymbol{A} \in \mathbb{R}^{20\times 5}$, where all the elements are generated from $N(0,1)$ independently. Then data vectors are generated by $\boldsymbol{x}_i \overset{\text{i.i.d.}}{\sim} N(\boldsymbol{A}\boldsymbol{x}_{k_i}^{(0)}, \boldsymbol{I}_{20})$ for $i = 1, 2, \ldots, 200$, where $k_i := \lceil i/50 \rceil$ represents cluster index to which $\boldsymbol{x}_i$ belongs. $\{\boldsymbol{x}_i\}_{i=1}^{50}$ are finally rescaled such that $\sum_{i=1}^{200} \|\boldsymbol{x}_i\|_2/200 = 4$. Then binary link weights $\{w_{ij}\}_{i,j=1}^{200}$ are generated from Bernoulli distribution. For $k_i = k_j$, $w_{ij} = w_{ji} \sim B(0.05)$ $(i \neq j)$ and $w_{ii} = 0$. For $k_i \neq k_j$, $w_{ij} = w_{ji} \sim B(\xi)$ with a specified parameter $\xi \in [0, 1]$.

C.2 Tensor Factorization as a BHLR

As explained in Section 5.4.3, tensor factorization (TF; Cichocki et al. (2009)) decomposes a given tensor $\boldsymbol{V} = (v_{\boldsymbol{j}}) \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_U}$ into matrices $\boldsymbol{\xi}^{(u)} = (\xi_{ik}^{(u)}) \in \mathbb{R}^{n_u\times K}$, by minimizing the BD between the entries of $\boldsymbol{V}$ and the tensor product $[\![\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}, \ldots, \boldsymbol{\xi}^{(U)}]\!]$ whose $\boldsymbol{j} = (j_1, j_2, \ldots, j_U)$-th entry is specified as $\sum_{k=1}^K \xi_{j_1k}^{(1)} \xi_{j_2k}^{(2)} \cdots \xi_{j_Uk}^{(U)}$. Namely, TF minimizes the BD

$$D_\varphi(\{v_{\boldsymbol{j}}\}_{\boldsymbol{j}\in[n_1]\times[n_2]\times\cdots\times[n_U]}, \{\langle \boldsymbol{\xi}_{j_1}^{(1)}, \boldsymbol{\xi}_{j_2}^{(2)}, \ldots, \boldsymbol{\xi}_{j_U}^{(U)}\rangle\}_{\boldsymbol{j}\in[n_1]\times[n_2]\times\cdots\times[n_U]}) \tag{C.1}$$

where $\langle \boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}'' \ldots\rangle := \sum_{k=1}^K y_k y_k' y_k'' \cdots$, and $\boldsymbol{\xi}_l^{(u)} = (\xi_{l1}^{(u)}, \xi_{l2}^{(u)}, \ldots, \xi_{lK}^{(u)})$ $(l \in [n_u])$ are column vectors of the matrix $\boldsymbol{\xi}^{(u)}$. Subsequently, we can expect that $v_{\boldsymbol{j}} \approx \langle \boldsymbol{\xi}_{j_1}^{(1)}, \boldsymbol{\xi}_{j_2}^{(2)}, \ldots, \boldsymbol{\xi}_{j_U}^{(U)}\rangle$ for all $\boldsymbol{j} \in [n_1] \times [n_2] \times \cdots \times [n_U]$.

For showing that BHLR includes TF ($U \geq 2$), we first briefly review the relation between BHLR and MF ($U = 2$), that is explained in Section 5.4.2. In the case of $U = 2$, factorizing the matrix $\boldsymbol{V}$ corresponds to BHLR using

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{O}_{n_1\times n_1} & \boldsymbol{V} \\ \boldsymbol{V}^\top & \boldsymbol{O}_{n_2\times n_2} \end{pmatrix}, \tag{C.2}$$

that is defined in eq. (5.19). The link weights (C.2) indicate $v_{\boldsymbol{j}} = v_{j_1,j_2} = w_{j_1,n_1+j_2} = w_{\boldsymbol{i}}$; indices of the matrix $\boldsymbol{V} = (v_{\boldsymbol{j}})$ are formally transformed into those of the matrix $\boldsymbol{W} = (w_{\boldsymbol{i}})$, by utilizing the conversion $\mathcal{F} : (j_1, j_2) \mapsto$

$(j_1, n_1 + j_2) =: (i_1, i_2)$. Although this conversion only considers the correspondence between $\boldsymbol{V}$ and the upper-right part of the matrix $\boldsymbol{W}$, the lower-left part is specified by the symmetry of $\boldsymbol{W}$. In the case of $U \geq 2$, we generalize the conversion as

$$\begin{aligned}\mathcal{F} : (j_1, j_2, \ldots, j_U) &\mapsto \left(j_1,\ n_1 + j_2,\ (n_1 + n_2) + j_3,\ \ldots,\ (\sum_{u=1}^{U-1} n_u) + j_U \right) \\ &=: (i_1, i_2, \ldots, i_U),\end{aligned}$$

whose inverse $\mathcal{F}^{-1}$ can be defined over a set

$$\begin{aligned}\mathcal{C}(n_1, n_2, \ldots, n_U) &:= \{\boldsymbol{i} \mid \boldsymbol{i} = \mathcal{F}(\boldsymbol{j}), \boldsymbol{j} \in [n_1] \times [n_2] \times \cdots \times [n_U]\} \\ &= \{\boldsymbol{i} = (i_1, i_2, \ldots, i_U) \mid i_1 = 1, 2, \ldots, n_1; \\ &\qquad i_2 = n_1 + 1, n_1 + 2, \ldots, n_1 + n_2; \cdots; \\ &\qquad i_U = \sum_{u=1}^{U-1} n_u + 1, \ldots, \sum_{u=1}^{U} n_u\},\end{aligned}$$

such that $\mathcal{F}^{-1} : \mathcal{C}(n_1, n_2, \ldots, n_U) \ni \boldsymbol{i} \mapsto \boldsymbol{j} \in [n_1] \times [n_2] \times \cdots \times [n_U]$. Since $\mathcal{F}^{-1}$ converts the indices of $\boldsymbol{W} = (w_{\boldsymbol{i}})$ to those of $\boldsymbol{V} = (v_j)$, we may specify the hyperlink weights as $w_{\boldsymbol{i}} := v_{\mathcal{F}^{-1}(\boldsymbol{i})}$ for all $\boldsymbol{i} \in \mathcal{C}(n_1, n_2, \ldots, n_U)$, similarly to $U = 2$.

Although the above specification is essentially sufficient for describing the relation between BHLR and TF, the hyperlink weights $\boldsymbol{W} = (w_{\boldsymbol{i}})$ are assumed to be symmetric. The symmetry can be realized by considering the non-decreasing order permutation $r(\boldsymbol{i})$ defined for any $\boldsymbol{i}$; a tensor $\boldsymbol{W} = (w_{\boldsymbol{i}}) \in \mathbb{R}^{N^U}$ ($N := \sum_{u=1}^{U} n_u$), whose entries are specified as

$$w_{\boldsymbol{i}} := \begin{cases} v_{\mathcal{F}^{-1}(r(\boldsymbol{i}))} & (r(\boldsymbol{i}) \in \mathcal{C}(n_1, n_2, \ldots, n_U)) \\ 0 & (\text{otherwise}) \end{cases} \quad (\forall \boldsymbol{i} \in [N]^U), \tag{C.3}$$

simultaneously satisfies the symmetry $w_{\boldsymbol{i}} = w_{\boldsymbol{i}'}$ for any $\boldsymbol{i}' \in [N]^U$ obtained by permutating the entries of $\boldsymbol{i} \in [N]^U$, and the above specification $w_{\boldsymbol{i}} = v_{\mathcal{F}^{-1}(\boldsymbol{i})}$ for any $\boldsymbol{i} \in \mathcal{C}(n_1, n_2, \ldots, n_U)$. Therefore, (C.3) generalizes (C.2) from the case of $U = 2$ to $U \geq 2$.

Using the weights (C.3), one-hot vector $\boldsymbol{x}_i \in \{0, 1\}^N$ whose i-th entry is 1 and 0 otherwise ($i \in [N]$) and the parameter $\boldsymbol{\theta} = (\boldsymbol{\xi}^{(1)\top}, \boldsymbol{\xi}^{(2)\top}, \ldots, \boldsymbol{\xi}^{(U)\top})^\top \in \mathbb{R}^{N \times K}$, we have

$$\begin{aligned}(C.1) = D_\varphi(&\{w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{C}(n_1, n_2, \ldots, n_U)}, \\ &\{\underbrace{\langle \boldsymbol{\theta}^\top \boldsymbol{x}_{i_1}, \boldsymbol{\theta}^\top \boldsymbol{x}_{i_2}, \ldots, \boldsymbol{\theta}^\top \boldsymbol{x}_{i_U} \rangle}_{=: \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_i)}\}_{\boldsymbol{i} \in \mathcal{C}(n_1, n_2, \ldots, n_U)}),\end{aligned} \tag{C.4}$$

generalizing eq. (5.20) from $U = 2$ to $U \geq 2$. Therefore, TF that minimizes $(C.1)$ is equivalent to BHLR minimizing $(C.4)$; TF is a special case of BHLR.

C.3 Proofs

In C.3.1, we first show and prove Theorem C.1, that is the law of large numbers for multiply-indexed partially-dependent random variables. In C.3.2, we prove Proposition 5.5.1 by applying Theorem C.1. In C.3.3, we prove Theorem 5.5.1, indicating that BHLR asymptotically recovers the underlying conditional expectation of link weights as $n \to \infty$. In C.3.4, we last prove Theorem 5.5.2, showing the asymptotics of the minibatch SGD using the proposed Algorithm 1, as $T \to \infty$.

C.3.1 Preliminary for Proofs

Theorem C.1: Law of large numbers (LLN)

Let $\boldsymbol{Z} := (Z_{\boldsymbol{i}})$ be an array of random variables $Z_{\boldsymbol{i}} \in \mathcal{Z}$, $\boldsymbol{i} \in \mathcal{I}_n^{(U)} = \mathcal{J}_n^{(U)} \overset{(5.21)}{:=} \{(i_1, i_2, \ldots, i_U) \mid 1 \leq i_1 < i_2 < \cdots < i_U \leq n\}$, and $h : \mathcal{Z} \to \mathbb{R}$ be a continuous function. We assume that $Z_{\boldsymbol{i}}$ is independent of $Z_{\boldsymbol{j}}$ if $\boldsymbol{j} \in \mathcal{R}_n^{(U)}(\boldsymbol{i}) := \{(j_1, j_2, \ldots, j_U) \in \mathcal{I}_n^{(U)} \mid j_1, j_2, \ldots, j_U \in \{1, \ldots, n\} \setminus \{i_1, i_2, \ldots, i_U\}\}$, and $\mathbb{E}_{\boldsymbol{Z}}(h(Z_{\boldsymbol{i}})^2) < \infty$, for all $\boldsymbol{i} \in \mathcal{I}_n^{(U)}$. Then the average of $h(Z_{\boldsymbol{i}})$ over $\mathcal{I}_n^{(U)}$ converges to the expectation in probability as $n \to \infty$; that is

$$\frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} h(Z_{\boldsymbol{i}}) = \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \mathbb{E}_{\boldsymbol{Z}}(h(Z_{\boldsymbol{i}})) + O_p(1/\sqrt{n}).$$

Proof of Theorem C.1. Regarding the variance of the average, we have

$$\begin{aligned}
&\mathbb{V}_{\boldsymbol{Z}} \left(\frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} h(Z_{\boldsymbol{i}}) \right) \\
&= \mathbb{E}_{\boldsymbol{Z}} \left(\left(\frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} h(Z_{\boldsymbol{i}}) \right)^2 \right) - \mathbb{E}_{\boldsymbol{Z}} \left(\frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} h(Z_{\boldsymbol{i}}) \right)^2 \\
&= \frac{1}{|\mathcal{I}_n^{(U)}|^2} \left(\sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \sum_{\boldsymbol{j} \in \mathcal{I}_n^{(U)}} \mathbb{E}_{\boldsymbol{Z}} \left(h(Z_{\boldsymbol{i}}) h(Z_{\boldsymbol{j}}) \right) - \left(\sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \mathbb{E}_{\boldsymbol{Z}} \left(h(Z_{\boldsymbol{i}}) \right) \right)^2 \right) \\
&= \frac{1}{|\mathcal{I}_n^{(U)}|^2} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \sum_{\boldsymbol{j} \in \mathcal{I}_n^{(U)} \setminus \mathcal{R}_n^{(U)}(\boldsymbol{i})} \left(\mathbb{E}_{\boldsymbol{Z}} \left(h(Z_{\boldsymbol{i}}) h(Z_{\boldsymbol{j}}) \right) - \mathbb{E}_{\boldsymbol{Z}}(h(Z_{\boldsymbol{i}})) \mathbb{E}_{\boldsymbol{Z}}(h(Z_{\boldsymbol{j}})) \right),
\end{aligned} \tag{C.5}$$

where $\mathbb{E}_{\boldsymbol{Z}}, \mathbb{V}_{\boldsymbol{Z}}$ represent expectation and variance with respect to $\boldsymbol{Z}$. Considering $\mathbb{E}_{\boldsymbol{Z}}(|h(Z_{\boldsymbol{i}})|) < \infty, \mathbb{E}_{\boldsymbol{Z}}(|h(Z_{\boldsymbol{i}})h(Z_{\boldsymbol{j}})|) < \infty$, $|\mathcal{I}_n^{(U)}| = O(n^U)$, and

$$\begin{aligned}
|\mathcal{I}_n^{(U)} \setminus \mathcal{R}_n^{(U)}(\boldsymbol{i})| &= \Big| \big\{ (j_1, j_2, \ldots, j_U) \in \mathcal{I}_n^{(U)} \big| \\
&\qquad\qquad \exists u \in \{1, 2, \ldots, U\} \text{ s.t. } j_u \in \{i_1, i_2, \ldots, i_U\} \big\} \Big| \\
&\leq \sum_{u=1}^{U} \left| \left\{ (j_1, j_2, \ldots, j_U) \in \mathcal{I}_n^{(U)} \middle| j_u \in \{i_1, i_2, \ldots, i_U\} \right\} \right| \\
&= \sum_{u=1}^{U} \sum_{l=1}^{U} \left| \left\{ (j_1, \ldots, j_{u-1}, i_l, j_{u+1}, \ldots, j_U) \in \mathcal{I}_n^{(U)} \right\} \right| \\
&= O(U^2 n^{U-1}) = O(n^{U-1})
\end{aligned}$$

for any fixed $\boldsymbol{i} = (i_1, i_2, \ldots, i_U) \in \mathcal{I}_n^{(U)}$, the formula (C.5) is of order $O(n^{-2U} \cdot n^U \cdot n^{U-1}) = O(n^{-1})$. Therefore,

$$\mathbb{V}_{\boldsymbol{Z}} \left(\frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} h(Z_{\boldsymbol{i}}) \right) = O(n^{-1}). \tag{C.6}$$

(C.6) and Chebyshev's inequality indicate the assertion. □

We note that the convergence rate is $O_p(n^{-1/2})$ but not $O_p(1/|\mathcal{I}_n^{(U)}|^{1/2}) = O_p(n^{-U/2})$, even though we leverage $|\mathcal{I}_n^{(U)}| = O(n^U)$ observations $\{Z_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}}$.

C.3.2 Proof of Proposition 5.5.1

By a simple calculation, we have

$$\begin{aligned}
L_{\varphi,n}(\boldsymbol{\theta}) &= \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \left\{ \varphi(w_{\boldsymbol{i}}) - \varphi(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) - \varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))(w_{\boldsymbol{i}} - \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \right\} \\
&= \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \underbrace{\left\{ \varphi(\mu_*(\boldsymbol{X}_{\boldsymbol{i}})) - \varphi(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) - \varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}) - \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \right\}}_{= d_\varphi(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))} \qquad \text{(C.7)} \\
&\quad + \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \left\{ \varphi(w_{\boldsymbol{i}}) - \varphi(\mu_*(\boldsymbol{X}_{\boldsymbol{i}})) \right\} \qquad \text{(C.8)} \\
&\quad + \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \left\{ \varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}) - w_{\boldsymbol{i}}) \right\}. \qquad \text{(C.9)}
\end{aligned}$$

Under the conditions (C-1)–(C-5), Theorem C.1 can be applied to the terms (C.7)–(C.9) as shown in the following:

specifying $Z_{\boldsymbol{i}} := \boldsymbol{X}_{\boldsymbol{i}}, h(Z_{\boldsymbol{i}}) := d_\varphi(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))$ leads to

$$\begin{aligned}(C.7) &\overset{\text{Theorem C.1}}{=} \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \mathbb{E}_{\mathcal{X}^U}(d_\varphi(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))) + O_p(1/\sqrt{n}) \\ &= \mathbb{E}_{\mathcal{X}^U}(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}))) + O_p(1/\sqrt{n}),\end{aligned}$$

specifying $Z_{\boldsymbol{i}} := (w_{\boldsymbol{i}}, \boldsymbol{X}_{\boldsymbol{i}}), h(Z_{\boldsymbol{i}}) := \varphi(w_{\boldsymbol{i}}) - \varphi(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}))$ leads to

$$\begin{aligned}(C.8) &\overset{\text{Theorem C.1}}{=} \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \mathbb{E}_{Z_{\boldsymbol{i}}}\left(\varphi(w_{\boldsymbol{i}}) - \varphi(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}))\right) + O_p(1/\sqrt{n}) \\ &= \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \mathbb{E}_{\mathcal{X}^U}\left(\mathbb{E}(\varphi(w_{\boldsymbol{i}}) \mid \boldsymbol{X}_{\boldsymbol{i}}) - \varphi(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}))\right) + O_p(1/\sqrt{n}) \\ &= \mathbb{E}_{\mathcal{X}^U}\left(\mathbb{E}(\varphi(w) \mid \boldsymbol{X}) - \varphi(\mu_*(\boldsymbol{X}))\right) + O_p(1/\sqrt{n}) \\ &= C_\varphi + O_p(1/\sqrt{n}),\end{aligned}$$

and specifying $Z_{\boldsymbol{i}} := (w_{\boldsymbol{i}}, \boldsymbol{X}_{\boldsymbol{i}}), h(Z_{\boldsymbol{i}}) := \varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}) - w_{\boldsymbol{i}})$ leads to

$$\begin{aligned}(C.9) &\overset{\text{Theorem C.1}}{=} \mathbb{E}_{Z_{\boldsymbol{i}}}\left(\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}) - w_{\boldsymbol{i}})\right) + O_p(1/\sqrt{n}) \\ &= \mathbb{E}_{\mathcal{X}^U}(\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))\underbrace{(\mu_*(\boldsymbol{X}_{\boldsymbol{i}}) - \underbrace{\mathbb{E}(w_{\boldsymbol{i}} \mid \boldsymbol{X}_{\boldsymbol{i}})}_{=\mu_*(\boldsymbol{X}_{\boldsymbol{i}})})}_{=0}) + O_p(1/\sqrt{n}) \\ &= O_p(1/\sqrt{n}).\end{aligned}$$

Thus proving the assertion

$$\begin{aligned}L_{\varphi,n}(\boldsymbol{\theta}) &= (C.7) + (C.8) + (C.9) \\ &= \mathbb{E}_{\mathcal{X}^U}(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}))) + C_\varphi + O_p(1/\sqrt{n}).\end{aligned}$$

□

C.3.3 Proof of Theorem 5.5.1

Definition of the estimator (5.9) leads to

$$L_{\varphi,n}(\boldsymbol{\theta}_*) - C_\varphi \geq \min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} L_{\varphi,n}(\boldsymbol{\theta}) - C_\varphi = L_{\varphi,n}(\hat{\boldsymbol{\theta}}_{\varphi,n}) - C_\varphi. \tag{C.10}$$

We evaluate both sides of the inequality (C.10), for proving the assertion.

- Regarding the left-hand side of the inequality (C.10), Proposition 5.5.1 indicates that

$$L_{\varphi,n}(\boldsymbol{\theta}_*) - C_\varphi = L_{\varphi,n}(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_*} - C_\varphi$$

$$\begin{aligned}
&\overset{\text{Proposition 5.5.1}}{=} \left(\mathbb{E}_{\mathcal{X}^U}(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X})))\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_*} \right. \\
&\qquad\qquad \left. + C_\varphi + \varepsilon_n^{(1)} \right) - C_\varphi \\
&= \underbrace{\mathbb{E}_{\mathcal{X}^U}(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}_*}(\boldsymbol{X})))}_{=0} + \varepsilon_n^{(1)} \qquad \text{(C.11)} \\
&= \varepsilon_n^{(1)} \qquad \text{(C.12)}
\end{aligned}$$

where $\varepsilon_n^{(1)} := L_{\varphi,n}(\boldsymbol{\theta}_*) - \left(\mathbb{E}_{\mathcal{X}^U}(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}_*}(\boldsymbol{X})) + C_\varphi\right) = O_p(1/\sqrt{n})$.

- We here consider the right-hand side of the inequality (C.10). Since the function φ is strongly convex, the definition indicates the existence of $M_\varphi > 0$ such that

$$d_\varphi(a, b) = \varphi(a) - (\varphi(b) + \varphi'(b)(a - b)) \overset{(\because \text{srtongly convex})}{\geq} M_\varphi \cdot (a - b)^2,$$

for all $a, b \in \mathrm{dom}(\varphi)$. This inequality indicates that the squared difference is bounded by the function d_φ. By substituting $\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X})$ into a, b, respectively, we have an inequality

$$d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X})) \geq M_\varphi \cdot (\mu_*(\boldsymbol{X}) - \mu_{\boldsymbol{\theta}}(\boldsymbol{X}))^2, \quad (\forall \boldsymbol{\theta} \in \boldsymbol{\Theta}). \qquad \text{(C.13)}$$

Using the above inequality (C.13), the right-hand side of the inequality (C.10) is evaluated as

$$\begin{aligned}
&L_{\varphi,n}(\hat{\boldsymbol{\theta}}_{\varphi,n}) - C_\varphi \\
&\quad = L_{\varphi,n}(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{\varphi,n}} - C_\varphi \\
&\quad \overset{\text{Proposition 5.5.1}}{=} \left(\mathbb{E}_{\mathcal{X}^U}(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}))) \right. \\
&\qquad\qquad \left. + C_\varphi + \varepsilon_n^{(2)}(\boldsymbol{\theta}) \right)\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{\varphi,n}} - C_\varphi \\
&\quad = \mathbb{E}_{\mathcal{X}^U}(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X})))\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{\varphi,n}} + \varepsilon_n^{(2)}(\hat{\boldsymbol{\theta}}_{\varphi,n}) \\
&\quad \overset{\text{Ineq. (C.13)}}{\geq} M_\varphi \cdot \mathbb{E}_{\mathcal{X}^U}((\mu_*(\boldsymbol{X}) - \mu_{\boldsymbol{\theta}}(\boldsymbol{X}))^2)\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{\varphi,n}} + \varepsilon_n^{(2)}(\hat{\boldsymbol{\theta}}_{\varphi,n}), \\
&\quad = M_\varphi \cdot \|\mu_* - \mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}\|^2 + \varepsilon_n^{(2)}(\hat{\boldsymbol{\theta}}_{\varphi,n}) \qquad \text{(C.14)}
\end{aligned}$$

where $\|f\| := \mathbb{E}_{\mathcal{X}^U}(f(\boldsymbol{X})^2)^{1/2}$ for functions $f : \mathcal{X}^U \to \mathbb{R}$ and $\varepsilon_n^{(2)}(\boldsymbol{\theta}) := L_{\varphi,n}(\boldsymbol{\theta}) - \left\{\mathbb{E}_{\mathcal{X}^U}\left(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X})\right) + C_\varphi\right\}$ represents the residual in Proposition 5.5.1 using the parameter $\boldsymbol{\theta}$, that satisfies $\varepsilon_n^{(2)}(\boldsymbol{\theta}) = O_p(1/\sqrt{n})$ for each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

By substituting (C.12) and (C.14) into (C.10), we have

$$\varepsilon_n^{(1)} \geq M_\varphi \cdot \|\mu_* - \mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}\|^2 + \varepsilon_n^{(2)}(\hat{\boldsymbol{\theta}}_{\varphi,n}),$$

indicating that

$$\varepsilon_n^{(1)} - \varepsilon_n^{(2)}(\hat{\boldsymbol{\theta}}_{\varphi,n}) \geq M_\varphi \cdot \|\mu_* - \mu_{\hat{\boldsymbol{\theta}}_{\varphi,n}}\|^2 \geq 0, \tag{C.15}$$

where $\varepsilon_n^{(1)} = O_p(1/\sqrt{n}) = o_p(1)$. The term $\varepsilon_n^{(2)}(\hat{\boldsymbol{\theta}}_{\varphi,n})$ is proved to be $o_p(1)$, as shown in the remaining of this proof; then, (C.15) immediately proves Theorem 5.5.1.

Hereinafter, we last prove $\varepsilon_n^{(2)}(\hat{\boldsymbol{\theta}}_{\varphi,n}) = o_p(1)$, by employing Newey (1991) Corollary 2.2, indicating that $\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} |\varepsilon_n^{(2)}(\boldsymbol{\theta})| = o_p(1)$ under the following assumptions: (i) $\boldsymbol{\Theta}$ is compact, (ii) $\varepsilon_n^{(2)}(\boldsymbol{\theta}) = o_p(1)$ for each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, and (iii) $\exists B_n = O_p(1)$ such that $|\varepsilon_n^{(2)}(\boldsymbol{\theta}) - \varepsilon_n^{(2)}(\boldsymbol{\theta}')| \leq B_n \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$ for all $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \boldsymbol{\Theta}$. Above assumptions (i), (ii) and (iii) correspond to assumptions 1, 2 and 3A, in Newey (1991). In our setting, the assumption (i) is assumed, (ii) is proved by Proposition 5.5.1. (iii) is obtained as follows: since the product of two bounded Lipschitz continuous (LC) functions is LC, C^1-function applied to LC function is LC, and the expectation of LC function is also LC, there exist $M_1, M_2 > 0$ such that

$$\begin{aligned}
|\varepsilon_n^{(2)}(\boldsymbol{\theta}) - \varepsilon_n^{(2)}(\boldsymbol{\theta}')| \leq{}& \left|L_{\varphi,n}(\boldsymbol{\theta}) - L_{\varphi,n}(\boldsymbol{\theta}')\right| + \Big|\mathbb{E}_{\mathcal{X}^U}(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}}(\boldsymbol{X}))) \\
&- \mathbb{E}_{\mathcal{X}^U}(d_\varphi(\mu_*(\boldsymbol{X}), \mu_{\boldsymbol{\theta}'}(\boldsymbol{X})))\Big| \\
\leq{}& \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}} \left|\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))w_{\boldsymbol{i}} - \varphi'(\mu_{\boldsymbol{\theta}'}(\boldsymbol{X}_{\boldsymbol{i}}))w_{\boldsymbol{i}}\right| \\
&+ \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}} \Big|\underbrace{\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}_{\text{(Lipschitz)}} - \varphi'(\mu_{\boldsymbol{\theta}'}(\boldsymbol{X}_{\boldsymbol{i}}))\mu_{\boldsymbol{\theta}'}(\boldsymbol{X}_{\boldsymbol{i}})\Big| \\
&+ \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}} \Big|\underbrace{\varphi(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))}_{\text{(Lipschitz)}} - \varphi(\mu_{\boldsymbol{\theta}'}(\boldsymbol{X}_{\boldsymbol{i}}))\Big| \\
&+ \Big|\underbrace{\mathbb{E}_{\mathcal{X}^U}\left(\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}))\mu_*(\boldsymbol{X})\right)}_{\text{(Lipschitz)}} - \mathbb{E}_{\mathcal{X}^U}\left(\varphi'(\mu_{\boldsymbol{\theta}'}(\boldsymbol{X}))\mu_*(\boldsymbol{X})\right)\Big| \\
&+ \Big|\underbrace{\mathbb{E}_{\mathcal{X}^U}\left(\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}))\mu_{\boldsymbol{\theta}}(\boldsymbol{X})\right)}_{\text{(Lipschitz)}} - \mathbb{E}_{\mathcal{X}^U}\left(\varphi'(\mu_{\boldsymbol{\theta}'}(\boldsymbol{X}))\mu_{\boldsymbol{\theta}'}(\boldsymbol{X})\right)\Big| \\
&+ \Big|\underbrace{\mathbb{E}_{\mathcal{X}^U}\left(\varphi(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}))\right)}_{\text{(Lipschitz)}} - \mathbb{E}_{\mathcal{X}^U}\left(\varphi(\mu_{\boldsymbol{\theta}'}(\boldsymbol{X}))\right)\Big|
\end{aligned}$$

$$\leq \frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} |w_{\boldsymbol{i}}| \Big| \underbrace{\varphi'(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}))}_{\text{(Lipschitz)}} - \varphi'(\mu_{\boldsymbol{\theta}'}(\boldsymbol{X}_{\boldsymbol{i}})) \Big| + M_2 \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$$

$$\leq M_1 \left(\frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} |w_{\boldsymbol{i}}| \right) \cdot \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2 + M_2 \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2.$$

Denoting by $B_n := M_1 \left(\frac{1}{|\mathcal{I}_n^{(U)}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} |w_{\boldsymbol{i}}| \right) + M_2$, Proposition 5.5.1 indicates $B_n = O_p(1)$. Therefore the condition (iii) holds; Newey (1991) Corollary 2.2 proves

$$|\varepsilon_n^{(2)}(\hat{\boldsymbol{\theta}}_{\varphi,n})| \leq \sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} |\varepsilon_n^{(2)}(\boldsymbol{\theta})| \overset{\text{Newey (1991) Corollary 2.2}}{=} o_p(1), \tag{C.16}$$

indicating that $\varepsilon_n^{(2)}(\hat{\boldsymbol{\theta}}_{\varphi,n}) = o_p(1)$. □

C.3.4 Proof of Theorem 5.5.2

Proof is two-folded. In the following, we first verify that (i) $\mathbb{E}_{\mathcal{M}^{(t)}}(\tilde{g}_\eta^{(t)}(\boldsymbol{\theta})) = \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta})$, where

$$Q_\eta(\boldsymbol{\theta}) := D_\varphi(\{\eta w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}}, \{\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}}), \quad \alpha := \begin{cases} |\mathcal{I}_n^{(U)}|/|\mathcal{K}_{\boldsymbol{u}}| & (v = 1) \\ |\mathcal{I}_n^{(U)}| & (v = 0) \end{cases},$$

and we next prove (ii) $\mathbb{E}_\tau \left(\mathbb{E}_{\{\mathcal{M}^{(t)}\}_{t \in [\tau]}} (\| \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\tilde{\boldsymbol{\theta}}^{(\tau)}) \|_2^2) \right) = O(1/\log T)$ by referring to (i) and Ghadimi and Lan (2013) Theorem 2.1 (a). Then, the assertion is proved.

(i) We first verify that $\mathbb{E}_{\mathcal{M}^{(t)}}(\tilde{g}_\eta^{(t)}(\boldsymbol{\theta})) = \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta})$. Here, we first consider the case $U \geq 2, v \geq 1$. A vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_v)$ representing which of the entries in the index $\boldsymbol{i} = (i_1, i_2, \ldots, i_U)$ is fixed, is preliminary specified from the set $\{\boldsymbol{u} = (u_1, u_2, \ldots, u_v) \in [U]^v \mid u_1 < u_2 < \cdots < u_v\}$ by users. Then, considering a set $\mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j}) := \{\boldsymbol{i} := (i_1, i_2, \ldots, i_U) \mid \boldsymbol{i} \in \mathcal{I}_n^{(U)}, i_{u_1} = j_1, \ldots, i_{u_v} = j_v\}$ for $\boldsymbol{j} \in [n]^v$, Algorithm 1 that defines $\mathcal{M}^{(t)} = (\tilde{\mathcal{P}}_{\text{mini}}^{(t)}, \tilde{\mathcal{I}}_{\text{mini}}^{(t)}, s_+^{(t)}, s_-^{(t)})$ consists of the following two-steps. At iteration t,

step 1. $\boldsymbol{j}$ is randomly selected from a set $\mathcal{K}_{\boldsymbol{u}} := \{\boldsymbol{j} \in [n]^v \mid \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j}) \neq \emptyset\}$ with the probability $p_{\boldsymbol{j}}$ (in Theorem 5.5.2, $p_{\boldsymbol{j}}$ is assumed to be $1/|\mathcal{K}_{\boldsymbol{u}}|$),

step 2. m_-, m_+ entries are uniformly randomly selected from sets $\tilde{\mathcal{I}}_n^{(U)} = \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j})$ and $\tilde{\mathcal{P}}_n^{(U)} = \mathcal{P}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j}) := \{\boldsymbol{i}' \mid \boldsymbol{i}' \in \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j}), w_{\boldsymbol{i}'} \neq 0\}$, and denote the sets as $\tilde{\mathcal{I}}_{\text{mini}}^{(t)}, \tilde{\mathcal{P}}_{\text{mini}}^{(t)}$. Coefficients $s_+^{(t)} := |\tilde{\mathcal{P}}_n^{(U)}|/m_+$ and $s_-^{(t)} := |\tilde{\mathcal{I}}_n^{(U)}|/m_-$ are also defined.

Therefore, the expectation of the stochastic gradient $\tilde{g}_{\eta}^{(t)}(\boldsymbol{\theta})$ with respect to sampling the minibatch $\mathcal{M}^{(t)}$ is,

$$
\begin{aligned}
&\mathbb{E}_{\mathcal{M}^{(t)}}(\tilde{g}_{\eta}^{(t)}(\boldsymbol{\theta})) \\
&\quad= \mathbb{E}_{\mathcal{M}^{(t)}}\Bigg(s_{-}^{(t)} \sum_{i \in \tilde{\mathcal{I}}_{\text{mini}}^{(t)}} \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})}{\partial \boldsymbol{\theta}} \\
&\qquad\quad -\eta \cdot s_{+}^{(t)} \sum_{i \in \tilde{\mathcal{P}}_{\text{mini}}^{(t)}} w_i \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})}{\partial \boldsymbol{\theta}}\Bigg) \quad (\because \text{ the definition (5.29)}) \\
&\quad= \underbrace{\mathbb{E}_{\mathcal{M}^{(t)}}\left(s_{-}^{(t)} \sum_{i \in \tilde{\mathcal{I}}_{\text{mini}}^{(t)}} \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})}{\partial \boldsymbol{\theta}}\right)}_{(\star 1)} \\
&\qquad\quad -\eta \cdot \underbrace{\mathbb{E}_{\mathcal{M}^{(t)}}\left(s_{+}^{(t)} \sum_{i \in \tilde{\mathcal{P}}_{\text{mini}}^{(t)}} w_i \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})}{\partial \boldsymbol{\theta}}\right)}_{(\star 2)}, \qquad \text{(C.17)}
\end{aligned}
$$

where the term $(\star 1)$ is evaluated by taking expectation with respect to the two steps in Algorithm 1 as

$$
\begin{aligned}
(\star 1) &= \underbrace{\mathbb{E}_{\boldsymbol{j}}\left(s_{-}^{(t)} \underbrace{\mathbb{E}_{\tilde{\mathcal{I}}_{\text{mini}}^{(t)}}\left(\sum_{i \in \tilde{\mathcal{I}}_{\text{mini}}^{(t)}} \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})}{\partial \boldsymbol{\theta}} \mid \boldsymbol{j}\right)}_{\text{(expectation w.r.t. step 2)}}\right)}_{\text{(expectation w.r.t. step 1)}} \\
&= \mathbb{E}_{\boldsymbol{j}}\left(s_{-}^{(t)} \frac{m_{-}}{|\mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j})|} \sum_{i \in \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j})} \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})}{\partial \boldsymbol{\theta}}\right) \\
&= \mathbb{E}_{\boldsymbol{j}}\left(\sum_{i \in \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j})} \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})}{\partial \boldsymbol{\theta}}\right) \\
&\qquad\qquad \left(\because s_{-}^{(t)} = \frac{|\mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j})|}{m_{-}}\right) \\
&= \frac{1}{|\mathcal{K}_{\boldsymbol{u}}|} \sum_{\boldsymbol{j} \in \mathcal{K}_{\boldsymbol{u}}} \sum_{i \in \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j})} \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X_i})}{\partial \boldsymbol{\theta}} \\
&\qquad\qquad \left(\because p_{\boldsymbol{j}} = \frac{1}{|\mathcal{K}_{\boldsymbol{u}}|} \, (\forall \boldsymbol{j} \in \mathcal{K}_{\boldsymbol{u}})\right)
\end{aligned}
$$

$$= \frac{1}{|\mathcal{K}_{\boldsymbol{u}}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}}$$

$$\left(\because \bigcup_{\boldsymbol{j} \in \mathcal{K}_{\boldsymbol{u}}} \mathcal{I}_{n,\boldsymbol{u}}^{(U)}(\boldsymbol{j}) = \mathcal{I}_n^{(U)} \right), \quad \text{(C.18)}$$

and similarly,

$$(\star 2) = \frac{1}{|\mathcal{K}_{\boldsymbol{u}}|} \sum_{\boldsymbol{i} \in \mathcal{P}_n^{(U)}} w_{\boldsymbol{i}} \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}}. \quad \text{(C.19)}$$

Substituting (C.18) and (C.19) into (C.17) leads to

$$\begin{aligned}
\mathbb{E}_{\mathcal{M}^{(t)}}(\tilde{g}_\eta^{(t)}(\boldsymbol{\theta})) &= \frac{1}{|\mathcal{K}_{\boldsymbol{u}}|} \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}} \\
&\qquad - \eta \cdot \frac{1}{|\mathcal{K}_{\boldsymbol{u}}|} \sum_{\boldsymbol{i} \in \mathcal{P}_n^{(U)}} w_{\boldsymbol{i}} \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}} \\
&= \frac{|\mathcal{I}_n^{(U)}|}{|\mathcal{K}_{\boldsymbol{u}}|} \frac{1}{|\mathcal{I}_n^{(U)}|} \Bigg\{ \sum_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}} \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}}) \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}} \\
&\qquad - \sum_{\boldsymbol{i} \in \mathcal{P}_n^{(U)}} \eta w_{\boldsymbol{i}} \varphi''(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})) \frac{\partial \mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})}{\partial \boldsymbol{\theta}} \Bigg\} \\
&= \underbrace{\frac{|\mathcal{I}_n^{(U)}|}{|\mathcal{K}_{\boldsymbol{u}}|}}_{=\alpha} \underbrace{\frac{\partial}{\partial \boldsymbol{\theta}} D_\varphi(\{\eta w_{\boldsymbol{i}}\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}}, \{\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})\}_{\boldsymbol{i} \in \mathcal{I}_n^{(U)}})}_{=\frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta})} \\
&= \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta}).
\end{aligned}$$

Thus (i) is proved for the case $U \geq 2, v \geq 1$. Here, we also consider the case $U \in \mathbb{N}, v = 0$. As $v = 0$ indicates that there is no fixed entry in the index $\boldsymbol{i}$, meaning that the step 1 in the above explanation is skipped, Algorithm 1 consists of only the step 2. Thus, by noticing that $\tilde{\mathcal{P}}_n^{(U)} = \mathcal{P}_n^{(U)}, \tilde{\mathcal{I}}_n^{(U)} = \mathcal{I}_n^{(U)}$, following the same calculation leads to the equation $\mathbb{E}_{\mathcal{M}^{(t)}}(\tilde{g}_\eta^{(t)}(\boldsymbol{\theta})) = \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta})$, which is the same as the case of $U \geq 2, v \geq 1$.

Since v is limited to take value in $\{0, 1, 2, \ldots, U-1\}$, (i) is hereby proved for all the possible (U, v).

(ii) We next prove that $\mathbb{E}_\tau \left(\mathbb{E}_{\{\mathcal{M}^{(t)}\}_{t \in [\tau]}} (\| \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\tilde{\boldsymbol{\theta}}^{(\tau)}) \|_2^2) \right) = O(1/\log T)$ by referring to (i) and Ghadimi and Lan (2013) Theorem 2.1 (a). The following explanations are based on Ghadimi and Lan (2013), with corresponding symbols $k \Leftrightarrow t$, $R \Leftrightarrow \tau$, $N \Leftrightarrow T$, $\gamma_k \Leftrightarrow \gamma^{(t)}$, $x_k \Leftrightarrow \tilde{\boldsymbol{\theta}}^{(t)}$,

$f(x) \Leftrightarrow \alpha Q_\eta(\boldsymbol{\theta})$, $G(\cdot, \xi_k) \Leftrightarrow \tilde{g}_\eta^{(t)}(\cdot)$, $L \Leftrightarrow H$, $D_f \Leftrightarrow D$, $\nabla \Leftrightarrow \frac{\partial}{\partial \boldsymbol{\theta}}$.
Ghadimi and Lan (2013) Theorem 2.1 (a) shows that, the iterative update

$$\tilde{\boldsymbol{\theta}}^{(t+1)} = \tilde{\boldsymbol{\theta}}^{(t)} - \gamma^{(t)} \tilde{g}_\eta^{(t)}(\tilde{\boldsymbol{\theta}}^{(t)}) \tag{C.20}$$

satisfies

$$\mathbb{E}_\tau \left(\mathbb{E}_{\{\mathcal{M}^{(t)}\}_{t\in[\tau]}} \left(\left\| \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\tilde{\boldsymbol{\theta}}^{(\tau)}) \right\|_2^2 \right) \right) \leq H \cdot \frac{D^2 + \sigma^2 \sum_{t=1}^T \gamma^{(t)2}}{\sum_{t=1}^T (2\gamma^{(t)} - H\gamma^{(t)2})}, \tag{C.21}$$

where $D := \sqrt{\frac{2}{H}\left(Q_\eta(\tilde{\boldsymbol{\theta}}^{(1)}) - \inf_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} Q_\eta(\boldsymbol{\theta}) \right)}$, $H > 0$ is the Lipschitz constant of $\alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta})$, $\gamma^{(t)}$ represents the step size satisfying $\gamma^{(t)} < 2/H$, and the number of iterations τ is chosen from $\{1, 2, \ldots, T\}$ with the probability $\mathbb{P}(\tau = t) = \frac{2\gamma^{(t)} - H\gamma^{(t)2}}{\sum_{t=1}^T (2\gamma^{(t)} - H\gamma^{(t)2})}$, if (C-1) $\mathbb{E}_{\mathcal{M}^{(t)}}(\tilde{g}_\eta^{(t)}(\boldsymbol{\theta})) = \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta})$ and (C-2) $\mathbb{E}_{\mathcal{M}^{(t)}}(\|\tilde{g}_\eta^{(t)}(\boldsymbol{\theta}) - \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta})\|_2^2) < \sigma^2$ for some $\sigma \in (0, \infty)$, $(\forall \boldsymbol{\theta} \in \boldsymbol{\Theta})$ hold. These assumptions (C-1) and (C-2) correspond to eq. (1.2) and eq. (1.3) in Ghadimi and Lan (2013), respectively.

In the case of Theorem 5.5.2, the minibatch SGD (5.25) reduces to (C.20) due to the assumption $\boldsymbol{\Theta} = \mathbb{R}^q$, the step size satisfies $\gamma^{(t)} = \gamma t^{-1} \leq \gamma \overset{\text{(assumption)}}{<} 2/H$, (C-1) is proved by the above calculation (i), and (C-2) is proved by

$$\begin{aligned}
&\mathbb{E}_{\mathcal{M}^{(t)}} \left(\left\| \tilde{g}_\eta^{(t)}(\boldsymbol{\theta}) - \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta}) \right\|_2^2 \right) \\
&\quad = \mathbb{E}_{\mathcal{M}^{(t)}} \left(\sum_{\alpha=1}^p \left(\tilde{g}_\eta^{(t)}(\boldsymbol{\theta}) - \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta}) \right)_\alpha^2 \right) \\
&\quad = \sum_{\alpha=1}^p \mathbb{E}_{\mathcal{M}^{(t)}} \left(\left(\tilde{g}_\eta^{(t)}(\boldsymbol{\theta}) - \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta}) \right)_\alpha^2 \right) \\
&\quad = \mathrm{tr} \mathbb{E}_{\mathcal{M}^{(t)}} \left(\left(\tilde{g}_\eta^{(t)}(\boldsymbol{\theta}) - \alpha \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta}) \right)^{\otimes 2} \right) \\
&\quad = \mathrm{tr} \mathbb{V}_{\mathcal{M}^{(t)}}(\tilde{g}_\eta^{(t)}(\boldsymbol{\theta})) \\
&\quad \leq \sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathrm{tr} \mathbb{V}_{\mathcal{M}^{(1)}}(\tilde{g}_\eta^{(1)}(\boldsymbol{\theta})) =: \sigma^2 \overset{\text{(assumption)}}{<} \infty,
\end{aligned}$$

where $(\boldsymbol{z})_\alpha$ represents the α-th entry of the vector $\boldsymbol{z} = (z_1, z_2, \ldots, z_p)$, $\boldsymbol{z}^{\otimes 2} := \boldsymbol{z}\boldsymbol{z}^\top$, and $\mathrm{tr}\boldsymbol{Z}$ represents the trace of the matrix $\boldsymbol{Z} = (z_{ij})$, i.e., $\mathrm{tr}\boldsymbol{Z} = \sum_{\alpha=1}^p z_{\alpha\alpha}$. Thus (C.21) holds; we last evaluate the right hand side of (C.21) in the following.

Obviously, we have $H = O(1)$ and $\sigma^2 = O(1)$ due to the assumptions, and $D = O(1)$ since the Lipschitz continuity of $\frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta})$ proves that

$Q_\eta(\tilde{\boldsymbol{\theta}}^{(1)})$ is finite with any fixed $\tilde{\boldsymbol{\theta}}^{(1)} \in \boldsymbol{\Theta}$. Then, it holds for $\gamma^{(t)} = \gamma t^{-1}$ that

$$\begin{aligned} H \cdot \frac{D^2 + \sigma^2 \sum_{t=1}^T \gamma^{(t)2}}{\sum_{t=1}^T (2\gamma^{(t)} - H\gamma^{(t)2})} &= H \cdot \frac{D^2 + \sigma^2\gamma^2 \sum_{t=1}^T t^{-2}}{2\gamma \sum_{t=1}^T t^{-1} - \gamma^2 H \sum_{t=1}^T t^{-2}} \\ &\overset{(\star)}{\leq} H \cdot \frac{D^2 + \sigma^2\gamma^2\pi^2/6}{2\gamma \log T - \gamma^2 H\pi^2/6} \\ &= O(1/\log T). \end{aligned} \tag{C.22}$$

$$\left(\because\ H = O(1), \sigma^2 = O(1), D = O(1), \gamma = O(1) \right)$$

The inequality $(\star)$ holds as $\sum_{t=1}^T t^{-1} \geq \int_{t=1}^T t^{-1} \mathrm{d}t = \log T\ (\geq 0)$ and $\sum_{t=1}^T t^{-2} \leq \sum_{t=1}^\infty t^{-2} = \frac{\pi^2}{6}$. See, e.g., Hofbauer (2002) for the series expansion of $\sum_{t=1}^\infty t^{-2}$. Thus, substituting $\alpha = O(1)$ and (C.22) into (C.21) leads to

$$\mathbb{E}_\tau \left(\mathbb{E}_{\{\mathcal{M}^{(t)}\}_{t\in[\tau]}} \left(\left\| \frac{\partial}{\partial \boldsymbol{\theta}} Q_\eta(\boldsymbol{\theta}) \right\|_2^2 \Big|_{\boldsymbol{\theta}=\tilde{\boldsymbol{\theta}}^{(\tau)}} \right) \right) = O(1/\log T) \to 0, \quad (T \to \infty).$$

By noticing that $Q_\eta(\boldsymbol{\theta}) = D_\varphi(\{\eta w_{\boldsymbol{i}}\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}}, \{\mu_{\boldsymbol{\theta}}(\boldsymbol{X}_{\boldsymbol{i}})\}_{\boldsymbol{i}\in\mathcal{I}_n^{(U)}})$, Theorem 5.5.2 is proved. □

Bibliography

Akaho, Shotaro (2001). "A kernel method for canonical correlation analysis". In: *Proceedings of the International Meeting of the Psychometric Society*. Tokyo, Japan: Springer-Verlag.

Anderson, Theodore Wilbur (2003). *An Introduction to Multivariate Statistical Analysis*. Wiley Series in Probability and Statistics. Wiley. ISBN: 9780471360919.

Andrew, Galen et al. (2013). "Deep Canonical Correlation Analysis". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, pp. 1247–1255.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein generative adversarial networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 214–223.

Aronszajn, Nachman (1950). "Theory of reproducing kernels". In: *Transactions of the American mathematical society* 68.3, pp. 337–404.

Athreya, Avanti et al. (2018). "Statistical Inference on Random Dot Product Graphs: a Survey". In: *Journal of Machine Learning Research* 18.226, pp. 1–92.

Banerjee, Arindam et al. (2005). "Clustering with Bregman divergences". In: *Journal of Machine Learning Research* 6.Oct, pp. 1705–1749.

Basu, Ayanendranath et al. (1998). "Robust and Efficient Estimation by Minimising a Density Power Divergence". In: *Biometrika* 85.3, pp. 549–559.

Belkin, Mikhail and Partha Niyogi (2001). "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering". In: *Advances in Neural Information Processing Systems*. Vol. 14, pp. 585–591.

— (2003). "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural Computation* 15.6, pp. 1373–1396.

Benton, Adrian et al. (2019). "Deep Generalized Canonical Correlation Analysis". In: *Proceedings of the 4th Workshop on Representation Learning for NLP*. Florence, Italy: Association for Computational Linguistics, pp. 1–6. DOI: 10.18653/v1/W19-4301. URL: https://www.aclweb.org/anthology/W19-4301.

Berg, Christian, Jens Peter Reus Christensen, and Peter Ressel (1984). *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Graduate Texts in Mathematics. Springer New York. ISBN: 9781461211280.

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

Boissonnat, Jean-Daniel, Frank Nielsen, and Richard Nock (2010). "Bregman Voronoi Diagrams". In: *Discrete & Computational Geometry* 44.2, pp. 281–307.

Bojchevski, Aleksandar and Stephan Günnemann (2018). "Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking". In: *Proceedings of the International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1ZdKJ-0W.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge University Press. DOI: 10.1017/CBO9780511804441.

Bradley, Andrew P. (July 1997). "The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms". In: *Pattern Recognition* 30.7, pp. 1145–1159. ISSN: 0031-3203. DOI: 10.1016/S0031-3203(96)00142-2. URL: http://dx.doi.org/10.1016/S0031-3203(96)00142-2.

Bregman, Lev M (1967). "The Relaxation Method of Finding the Common Point of Convex Sets and Its Application to the Solution of Problems in Convex Programming". In: *USSR Computational Mathematics and Mathematical Physics* 7.3, pp. 200–217.

Bro, Rasmus (1997). "PARAFAC. Tutorial and applications". In: *Chemometrics and intelligent laboratory systems* 38.2, pp. 149–171.

Bromley, Jane et al. (1994). "Signature Verification using a "Siamese" Time Delay Neural Network". In: *Advances in Neural Information Processing Systems*, pp. 737–744.

Cameron, A Colin and Pravin K Trivedi (2007). "Essentials of Count Data Regression". In: *A Companion to Theoretical Econometrics*. 1st ed. Blackwell Oxford. Chap. 15, pp. 331–348.

— (2013). *Regression analysis of count data*. Vol. 53. Cambridge university press.

Censor, Yair, Stavros Andrea Zenios, et al. (1997). *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press on Demand.

Chung, Fan R. K. (1997). *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics 92. American Mathematical Society.

Cichocki, Andrzej and Rafal Zdunek (2006). *NTFLAB for Signal Processing*. Tech. rep. BSI, RIKEN.

Cichocki, Andrzej et al. (2009). *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. John Wiley & Sons.

Clauset, Aaron, Cristopher Moore, and Mark E. J. Newman (2008). "Hierarchical structure and the prediction of missing links in networks". In: *Nature* 453.7191, pp. 98–101.

Cobos, Fernando and Thomas Kühn (1990). "Eigenvalues of integral operators with positive definite kernels satisfying integrated Hölder conditions over metric compacta". In: *Journal of Approximation Theory* 63.1, pp. 39–55.

Cong, Fengyu et al. (2015). "Tensor decomposition of EEG signals: A brief review". In: *Journal of neuroscience methods* 248, pp. 59–69.

Cybenko, George (1989). "Approximation by Superpositions of a Sigmoidal Function". In: *Mathematics of Control, Signals, and Systems* 2.4, pp. 303–314.

Dai, Quanyu et al. (2018). "Adversarial Network Embedding". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. URL: https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16498.

De Maesschalck, Roy, Delphine Jouan-Rimbaud, and Désiré L Massart (2000). "The Mahalanobis distance". In: *Chemometrics and intelligent laboratory systems* 50.1, pp. 1–18.

Desmier, Elise et al. (2012). "Cohesive Co-evolution Patterns in Dynamic Attributed Graphs". In: *International Conference on Discovery Science*. Springer, pp. 110–124.

Donahue, Jeff et al. (2014). "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Bejing, China: PMLR, pp. 647–655.

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12, pp. 2121–2159.

Dunn, Joseph C (1981). "Global and asymptotic convergence rate estimates for a class of projected gradient processes". In: *SIAM Journal on Control and Optimization* 19.3, pp. 368–400.

Fan, Rong-En et al. (2008). "LIBLINEAR: A library for large linear classification". In: *Journal of Machine Learning Research* 9, pp. 1871–1874.

Faraut, Jacques and Khelifa Harzallah (1974). "Distances hilbertiennes invariantes sur un espace homogène". In: *Annales de l'institut Fourier*. Vol. 24. 3. Association des Annales de l'Institut Fourier, pp. 171–217.

Fukui, Kazuki, Akifumi Okuno, and Hidetoshi Shimodaira (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*, pp. 221–225.

Funahashi, Ken-Ichi (1989). "On the approximate realization of continuous mappings by neural networks". In: *Neural Networks* 2.3, pp. 183–192.

Gardner, Andrew et al. (2017). "On the Definiteness of Earth Mover's Distance and Its Relation to Set Intersection". In: *IEEE Transactions on Cybernetics* 48.11, pp. 3184–3196.

Ghadimi, Saeed and Guanghui Lan (2013). "Stochastic First- and Zeroth-order Methods for Nonconvex Stochastic Programming". In: *SIAM Journal on Optimization* 23.4, pp. 2341–2368.

Ghosh, Abhik, Ayanendranath Basu, et al. (2013). "Robust estimation for independent non-homogeneous observations using density power divergence with applications to linear regression". In: *Electronic Journal of statistics* 7, pp. 2420–2456.

Goldberger, Jacob et al. (2005). "Neighbourhood Components Analysis". In: *Advances in Neural Information Processing Systems*, pp. 513–520.

Golub, G and W Kahan (1965). "Calculating the Singular Values and Pseudo-Inverse of a Matrix". In: *J. SIAM. Numer. Anal., Ser. B* 2, pp. 205–224. DOI: 10.1137/0702016. URL: https://ci.nii.ac.jp/naid/30006817782/.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Gori, Marco, Gabriele Monfardini, and Franco Scarselli (2005). "A new model for learning in graph domains". In: *Proceedings of the IEEE International Joint Conference on Neural Networks*. Vol. 2. IEEE, pp. 729–734.

Grover, Aditya and Jure Leskovec (2016). "node2vec: Scalable Feature Learning for Networks". In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data mining*. ACM, pp. 855–864.

Guo, Yi, Junbin Gao, and Paul W. H. Kwan (2006). "Kernel Laplacian Eigenmaps for Visualization of Non-vectorial Data". In: *AI 2006: Advances in Artificial Intelligence*. Ed. by Abdul Sattar and Byeong-ho Kang. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1179–1183. ISBN: 978-3-540-49788-2.

Hamilton, William L., Zhitao Ying, and Jure Leskovec (2017). "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems*, pp. 1025–1035.

Harshman, Richard A, Sungjin Hong, and Margaret E Lundy (2003). "Shifted factor analysis—Part I: Models and properties". In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 17.7, pp. 363–378.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (Feb. 2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer. ISBN: 9780387848570.

He, Kaiming et al. (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In: *Proceedings of the 2015 IEEE International Conference on Computer Vision*. USA: IEEE Computer Society, 1026–1034.

He, Xiaofei and Partha Niyogi (2004). "Locality Preserving Projections". In: *Advances in Neural Information Processing Systems*, pp. 153–160.

Hernandez, Vicente, Jose E Roman, and Vicente Vidal (2005). "SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems". In: *ACM Transactions on Mathematical Software* 31.3, pp. 351–362.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks". In: *science* 313.5786, pp. 504–507.

Hofbauer, Josef (2002). "A Simple Proof of $1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots = \frac{\pi^2}{6}$ and Related Identities". In: *The American mathematical monthly* 109.2, pp. 196–200.

Holland, Paul W, Kathryn Blackmond Laskey, and Samuel Leinhardt (1983). "Stochastic blockmodels: First steps". In: *Social Networks* 5.2, pp. 109–137.

Horst, Paul (1961). "Generalized canonical correlations and their applications to experimental data". In: *Journal of Clinical Psychology* 17.4, pp. 331–347.

Hotelling, Harold (1936). "Relations between two sets of variates". In: *Biometrika* 28.3/4, pp. 321–377.

Huang, Zhiwu et al. (2012). "Cross-view Graph Embedding". In: *Proceedings of the Asian Conference on Computer Vision*, pp. 770–781.

Jeffrey, Johnson (2013). *Hypernetworks in the science of complex systems*. Vol. 3. World Scientific.

Jiang, Junyang et al. (2019). "Convolutional Gaussian Embeddings for Personalized Recommendation with Uncertainty". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International

Joint Conferences on Artificial Intelligence Organization, pp. 2642–2648. DOI: 10.24963/ijcai.2019/367. URL: https://doi.org/10.24963/ijcai.2019/367.

Jin, Chi et al. (2017). "How to Escape Saddle Points Efficiently". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 1724–1732.

Kanamori, Takafumi and Hironori Fujisawa (2015). "Robust Estimation under Heavy Contamination using Unnormalized Models". In: *Biometrika* 102.3, pp. 559–572.

Kawashima, Takayuki and Hironori Fujisawa (2019). "Robust and sparse regression in generalized linear model by stochastic optimization". In: *Japanese Journal of Statistics and Data Science*. ISSN: 2520-8764. DOI: 10.1007/s42081-019-00049-9. URL: https://doi.org/10.1007/s42081-019-00049-9.

Kettenring, Jon R (1971). "Canonical Analysis of Several Sets of Variables". In: *Biometrika* 58.3, pp. 433–451.

Kim, Geewook et al. (2019). "Representation Learning with Weighted Inner Product for Universal Approximation of General Similarities". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, pp. 5031–5038.

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

Kipf, Thomas N and Max Welling (2016). "Variational Graph Auto-Encoders". In: *NIPS Workshop on Bayesian Deep Learning*. arXiv:1611.07308.

— (2017). "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

Knapp, Anthony W. (2005). "Compact Self-Adjoint Operators". In: *Advanced Real Analysis: Along with a companion volume Basic Real Analysis*. Boston, MA: Birkhäuser Boston, pp. 34–53. ISBN: 978-0-8176-4442-0. DOI: 10.1007/0-8176-4442-3_2. URL: https://doi.org/10.1007/0-8176-4442-3_2.

Kolda, Tamara G and Brett W Bader (2009). "Tensor Decompositions and Applications". In: *SIAM review* 51.3, pp. 455–500.

Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix Factorization Techniques for Recommender Systems". In: *Computer* 42.8, pp. 30–37.

Kruskal, Joseph B (1964). "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis". In: *Psychometrika* 29.1, pp. 1–27.

Kullback, Solomon and Richard A Leibler (1951). "On information and sufficiency". In: *The annals of mathematical statistics* 22.1, pp. 79–86.

Kung, Sun Yuan (2014). *Kernel Methods and Machine Learning*. Cambridge University Press.

Lai, Pei Ling and Colin Fyfe (2000). "Kernel and nonlinear canonical correlation analysis". In: *International Journal of Neural Systems* 10.05, pp. 365–377.

Lampert, Christoph H, Hannes Nickisch, and Stefan Harmeling (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*. IEEE, pp. 951–958.

Lanczos, C. (1950). "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators". In: *Journal of Research of the National Bureau of Standard* 45.4, pp. 255–282. URL: `https://ci.nii.ac.jp/naid/10022240723/`.

Lee, Justin (1990). *U-statistics: Theory and Practice*. CRC Press.

Liben-Nowell, David and Jon Kleinberg (2007). "The Link-Prediction Problem for Social Networks". In: *Journal of the American society for Information Science and Technology* 58.7, pp. 1019–1031.

Lü, Linyuan and Tao Zhou (2011). "Link prediction in complex networks: A survey". In: *Physica A: statistical mechanics and its applications* 390.6, pp. 1150–1170.

Mantziou, Eleni, Symeon Papadopoulos, and Yiannis Kompatsiaris (2013). "Scalable training with approximate incremental laplacian eigenmaps and pca". In: *Proceedings of the 21st ACM International Conference on Multimedia*, pp. 381–384.

Mary, Xavier (2003). "Hilbertian subspaces, subdualities and applications". PhD thesis. PhD thesis, INSA Rouen.

Mikolov, Tomas et al. (2013). "Distributed Representations of Words and Phrases and Their Compositionality". In: *Advances in Neural Information Processing Systems*, pp. 3111–3119.

Miller, George A (1995). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41.

Minami, Mihoko and Shinto Eguchi (2002). "Robust blind source separation by beta divergence". In: *Neural computation* 14.8, pp. 1859–1886.

Minh, Ha Quang, Partha Niyogi, and Yuan Yao (2006). "Mercer's Theorem, Feature Maps, and Smoothing". In: *International Conference on Computational Learning Theory*. Springer, pp. 154–168.

Mørup, Morten and Mikkel N Schmidt (2006). *Sparse non-negative tensor 2D deconvolution (SNTF2D) for multi channel time-frequency analysis*. Tech. rep. Technical University of Denmark.

Newey, Whitney K (1991). "Uniform Convergence in Probability and Stochastic Equicontinuity". In: *Econometrica: Journal of the Econometric Society* 59.4, pp. 1161–1167.

Nickel, Maximillian and Douwe Kiela (2017). "Poincaré Embeddings for Learning Hierarchical Representations". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 6338–6347. URL: `http://papers.nips.cc/paper/7213-poincare-embeddings-for-learning-hierarchical-representations.pdf`.

— (2018). "Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3779–3788.

Nori, Nozomi, Danushka Bollegala, and Hisashi Kashima (2012). "Multinomial Relation Prediction in Social Data: A Dimension Reduction Approach." In: *Proceedings of the AAAI conference on Artificial Intelligence*. Vol. 12, pp. 115–121.

Oglic, Dino and Thomas Gaertner (2018). "Learning in Reproducing Kernel Krein Spaces". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3859–3867.

Okuno, Akifumi, Tetsuya Hada, and Hidetoshi Shimodaira (2018). "A probabilistic framework for multi-view feature learning with many-to-many associations via neural networks". In: *Proceedings of the International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3888–3897.

Okuno, Akifumi, Geewook Kim, and Hidetoshi Shimodaira (2019). "Graph Embedding With Shifted Inner Product Similarity and Its Improved Approximation Capability". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 644–653.

Okuno, Akifumi and Hidetoshi Shimodaira (2018). "On representation power of neural network-based graph embedding and beyond". In: *ICML workshop on Theoretical Foundations and Applications of Deep Generative Models*. arXiv:1805.12332. URL: `https://sites.google.com/view/tadgm/accepted-papers`.

— (2019). "Robust Graph Embedding with Noisy Link Weights". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 664–673.

— (2020). "Hyperlink Regression via Bregman Divergence". In: *Neural Networks* 126, pp. 362–383.

Ong, Cheng Soon et al. (2004). "Learning with non-positive kernels". In: *Proceedings of the 21st International Conference on Machine Learning*.

Oshikiri, Takamasa, Kazuki Fukui, and Hidetoshi Shimodaira (Aug. 2016). "Cross-Lingual Word Representations via Spectral Graph Embeddings". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Vol. 2, pp. 493–498.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543.

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena (2014). "DeepWalk: Online Learning of Social Representations". In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data mining*. ACM, pp. 701–710.

Prado, Adriana et al. (2013). "Mining graph topological patterns: Finding covariations among vertex descriptors". In: *IEEE Transactions on Knowledge and Data Engineering* 25.9, pp. 2090–2104.

Ravi, Sujith and Qiming Diao (May 2016). "Large Scale Distributed Semi-Supervised Learning Using Streaming Approximation". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Ed. by

Arthur Gretton and Christian C. Robert. Vol. 51. Proceedings of Machine Learning Research. Cadiz, Spain: PMLR, pp. 519–528.

Riesz, F. and B.S. Nagy (2012). *Functional Analysis*. Dover Books on Mathematics. Dover Publications. ISBN: 9780486162140.

Roweis, Sam T and Lawrence K Saul (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding". In: *Science* 290.5500, pp. 2323–2326.

Ruder, Sebastian (2016). "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747*.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1987). "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. Vol. 1. Cambridge, MA, USA: MIT Press. Chap. 8, pp. 318–362.

Sarkar, Rik (2011). "Low Distortion Delaunay Embedding of Trees in Hyperbolic Plane". In: *International Symposium on Graph Drawing*. Springer, pp. 355–366.

Sato, Ryoma (2020). "A Survey on The Expressive Power of Graph Neural Networks". In: *arXiv preprint arXiv:2003.04078*.

Scarselli, Franco et al. (2009). "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1, pp. 61–80.

Schölkopf, Bernhard (2001). "The kernel trick for distances". In: *Advances in Neural Information Processing Systems*, pp. 301–307.

Sen, Prithviraj et al. (2008). "Collective Classification in Network Data". In: *AI magazine* 29.3, p. 93.

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

Sra, Suvrit and Inderjit S Dhillon (2006). "Generalized Nonnegative Matrix Approximations with Bregman Divergences". In: *Advances in Neural Information Processing Systems*, pp. 283–290.

Sun, Shiliang (2013). "A survey of multi-view machine learning". In: *Neural Computing and Applications* 23.7-8, pp. 2031–2038.

Tan, Vincent YF and Cédric Févotte (2012). "Automatic Relevance Determination in Nonnegative Matrix Factorization with the/spl beta/-Divergence". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.7, pp. 1592–1605.

Tang, Jian et al. (2015). "LINE: Large-scale Information Network Embedding". In: *Proceedings of the International Conference on World Wide Web*, pp. 1067–1077.

Tang, Minh, Daniel L. Sussman, and Carey E. Priebe (June 2013). "Universally consistent vertex classification for latent positions graphs". In: *Ann. Statist.* 41.3, pp. 1406–1430. DOI: 10.1214/13-AOS1112. URL: https://doi.org/10.1214/13-AOS1112.

Telgarsky, Matus (2017). "Neural Networks and Rational Functions". In: *Proceedings of the International Conference on Machine Learning*. Ed. by Doina

Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 3387–3393.

Tenenbaum, Joshua B, Vin De Silva, and John C Langford (2000). "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500, pp. 2319–2323.

Tolstikhin, Ilya et al. (2018). "Wasserstein Auto-Encoders". In: *Proceedings of the International Conference on Representation Learning*. URL: https://openreview.net/forum?id=HkL7n1-0b.

Veitch, Victor et al. (2019). "Empirical Risk Minimization and Stochastic Gradient Descent for Relational Data". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 1733–1742.

Vilnis, Luke and Andrew McCallum (2015). "Word Representations via Gaussian Embedding". In: *Proceedings of the International Conference on Learning Representations*. URL: http://arxiv.org/abs/1412.6623.

Wang, Daixin, Peng Cui, and Wenwu Zhu (2016). "Structural Deep Network Embedding". In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, pp. 1225–1234.

Wang, Weiran et al. (2016). "Deep Variational Canonical Correlation Analysis". In: *arXiv preprint arXiv:1610.03454*.

Weisfeiler, Boris and Andrei A Lehman (1968). "A reduction of a graph to a canonical form and an algebra arising during this reduction". In: *Nauchno-Technicheskaya Informatsia* 2.9, pp. 12–16.

Xu, Keyulu et al. (2019). "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=ryGs6iA5Km.

Yan, Fei and Krystian Mikolajczyk (2015). "Deep Correlation for Matching Images and Text". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3441–3450.

Yan, Shuicheng et al. (2007). "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction". In: *IEEE transactions on Pattern Analysis and Machine Intelligence* 29.1, pp. 40–51.

Yarotsky, Dmitry (2017). "Error bounds for approximations with deep ReLU networks". In: *Neural Networks* 94, pp. 103–114.

— (2018). "Optimal approximation of continuous functions by very deep ReLU networks". In: *Proceedings of Machine Learning Research* 75. The 31st Annual Conference on Learning Theory, pp. 639–649.

Young, Stephen J and Edward R Scheinerman (2007). "Random dot product graph models for social networks". In: *International Workshop on Algorithms and Models for the Web-Graph*. Springer, pp. 138–149.

Zaheer, Manzil et al. (2017). "Deep Sets". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 3391–3401. URL: http://papers.nips.cc/paper/6931-deep-sets.pdf.

Zhang, Chunming, Yuan Jiang, and Yi Chai (2010). "Penalized Bregman divergence for large-dimensional regression and classification". In: *Biometrika* 97.3, pp. 551–566.

Zhang, Chunming, Yuan Jiang, and Zuofeng Shang (2009). "New aspects of Bregman divergence in regression and classification with parametric and nonparametric estimation". In: *Canadian Journal of Statistics* 37.1, pp. 119–139.

Zhang, Muhan et al. (2018). "Beyond Link Prediction: Predicting Hyperlinks in Adjacency Space". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. URL: https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17136.

Zhao, Jing et al. (2017). "Multi-view learning overview: Recent progress and new challenges". In: *Information Fusion* 38, pp. 43–54.

Zheng, Wenming et al. (2006). "Facial expression recognition using kernel canonical correlation analysis (KCCA)". In: *IEEE transactions on Neural Networks* 17.1, pp. 233–238.