Graph Embedding with Shifted Inner Product Similarity and Its Improved Approximation Capability

(Slides ▷ http://okuno.co/fimi.pdf)

to appear in AISTATS2019 (arXiv:1810.03463)

Joint work with H. Shimodaira and G. Kim

Akifumi Okuno

Kyoto University and RIKEN AIP

Purpose of this study

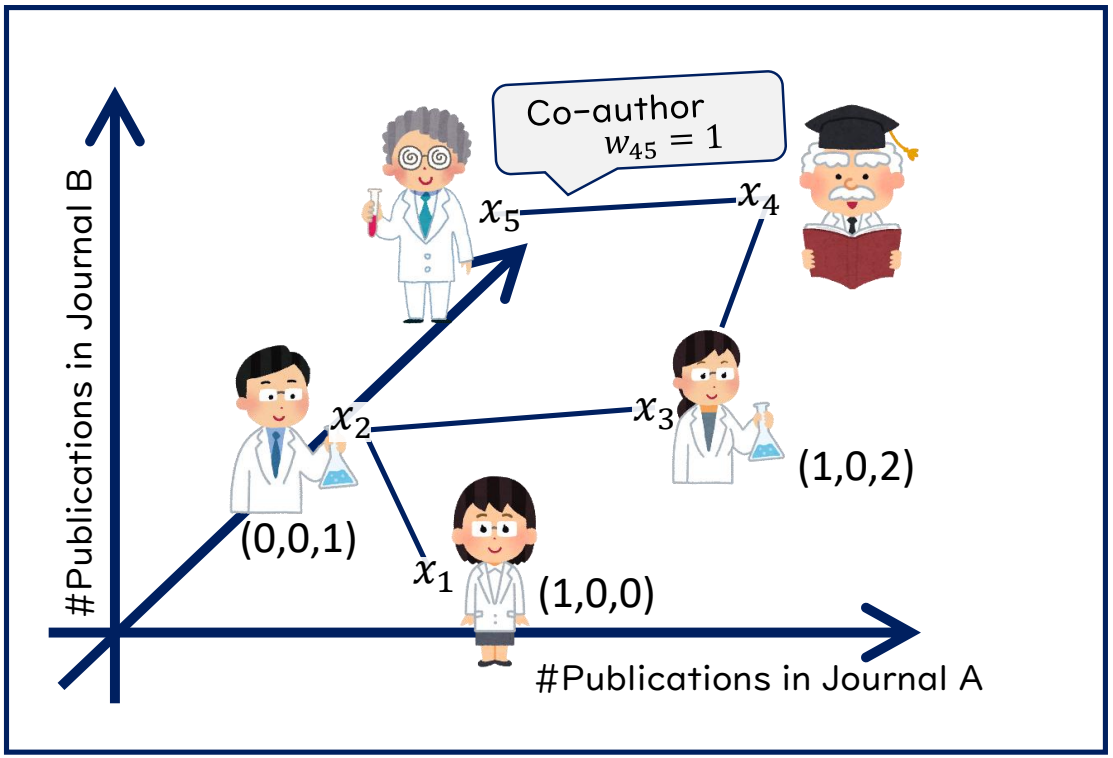
Purpose of this study is clarifying

expressive power

of neural network-based graph embedding (GE), and propose

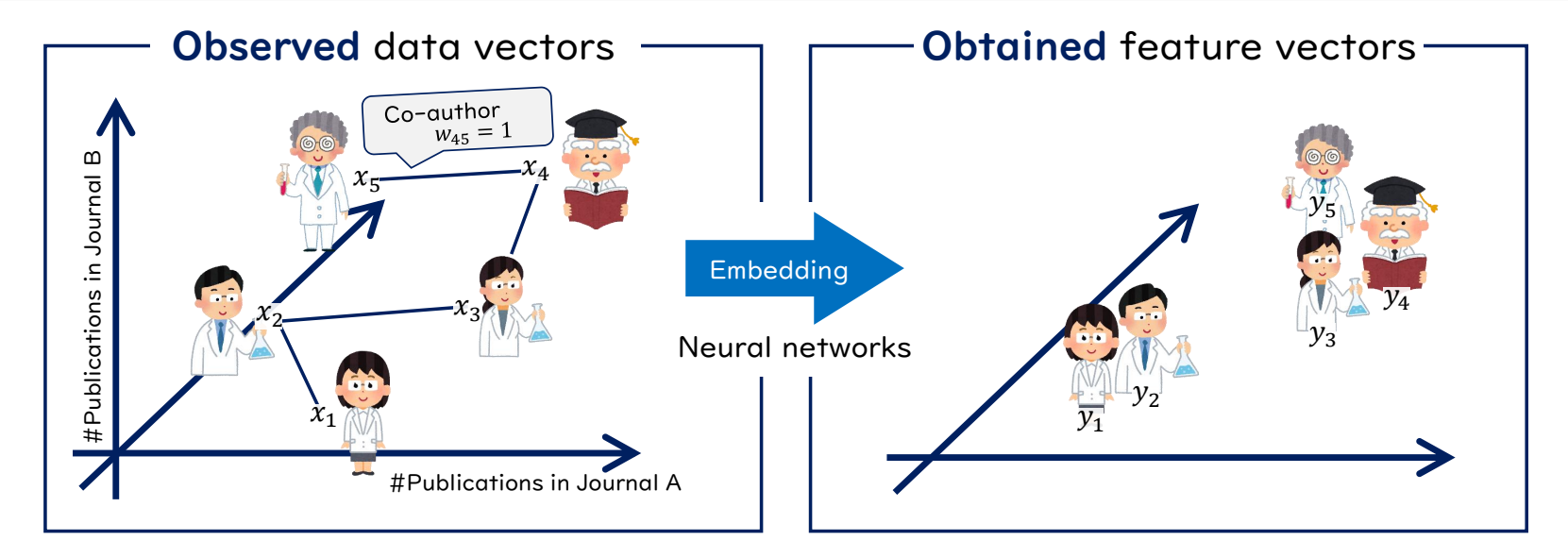
more expressive GE.

Given dataset



- Link weights $w_{ij} \geq 0$: co-authorship relation
- Data vectors $\boldsymbol{x}_i$: number of publications in each journal.

Summary of this study



Graph embedding learns a transformation $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$, so that

$$w_{ij} \approx \text{Sigmoid}(\underbrace{g(\boldsymbol{y}_i, \boldsymbol{y}_j)}_{\text{“Similarity”}}), \quad \boldsymbol{y}_i := \boldsymbol{f}(\boldsymbol{x}_i).$$

g is specified as inner-product (Tang et al., 2015), Poincaré-dist. (Nickel and Kiela, 2017), etc..

Question ▷ What kind of kernel g is good (in terms of the expressive power?)

Answer ▷ Proposed **shifted inner-product similarity (SIPS)**.

SIPS can approximate many kernels, e.g., Poincaré-distance!

Related publications

Today's talk is based on

- Okuno, Kim, and Shimodaira. "Graph Embedding with Shifted Inner Product Similarity and Its Improved Approximation Capability," arXiv:1810.03463, *to appear* in AISTATS2019,

that is an extended version of

- Okuno and Shimodaira. "On representation power of neural network-based graph embedding and beyond," arXiv:1805.12332, ICML2018 workshop on TADGM.

We recently submitted a preprint that proposes a new similarity.

- Kim, Okuno, Fukui, and Shimodaira. "Representation Learning with Weighted Inner Product for Universal Approximation of General Similarities," arXiv:1902.10409 , *submitted*.

Table of contents

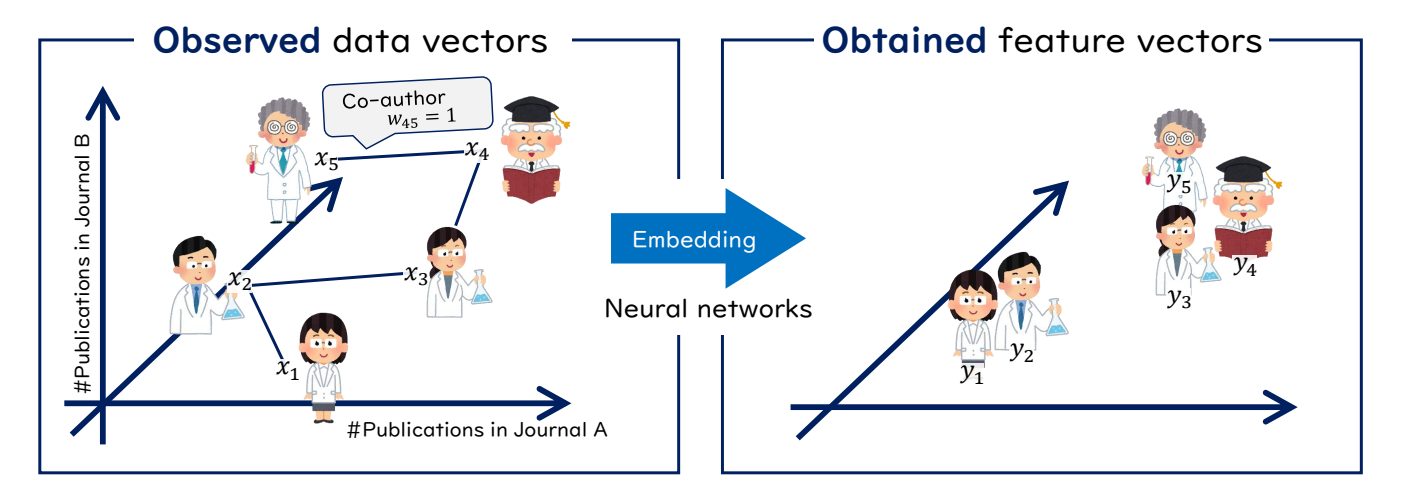
Table of contents

Table of contents

Table of contents

Introduction: Graph Embedding with Data Vectors

Formulation: Graph Embedding with Data Vectors



Given

- link weights $w_{ij} = w_{ji} \geq 0, \quad 1 \leq i < j \leq n,$
- data vectors $\boldsymbol{x}_i \in \mathcal{X} (\subset \mathbb{R}^p), \quad 1 \leq i \leq n,$

graph embedding (with data vectors) learns a transformation $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$, so that

$$w_{ij} \approx \exp(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle), \quad \boldsymbol{y}_i := \boldsymbol{f}(\boldsymbol{x}_i).$$

$\exp(\cdot)$ can be replaced with Sigmoid, or some other functions.

Existing Graph Embedding Methods

There are (mainly) 3 different types of graph embedding methods.

Spectral

Chung (1997),
Belkin and Niyogi (2003),
He and Niyogi (2004),...

☺ eigen-decomposition
☹ computationally hard
☹ limited to linear

Probabilistic

Mikolov et al. (2013),
Tang et al. (2015),...

☺ computationally tractable
☺ non-linear
☺ inductive

We employ this model.

Convolution

Kipf and Welling (2016),
Kipf and Welling (2017),
Hamilton et al. (2017),...

☺ computationally tractable
☺ non-linear
☺ graph convolution
☹ not inductive

Existing Graph Embedding Methods (Spectral)

We **do not consider this model** in this study.

Locality Preserving Projections (He and Niyogi, 2004, LPP) learns a linear-transformation

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

so that the obtained feature vectors $\{\boldsymbol{y}_i\}_{i=1}^n$ maximize

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\langle \boldsymbol{y}_i, \boldsymbol{y}_j\rangle,$$

under a quadratic constraint $\sum_{i=1}^n \sum_{j=1}^n w_{ij}\boldsymbol{y}_i\boldsymbol{y}_i^\top = \boldsymbol{I}$. LPP reduces to Spectral Graph Embedding (Chung, 1997) if $\{\boldsymbol{x}_i\}$ is specified as 1-hot vectors.

- ☺ can be **solved by eigen-decomposition**.
- ☹ **computationally hard**.
- ☹ limited to **linear** setting.

Existing Graph Embedding Methods (Probabilistic)

We employ this model in this study.

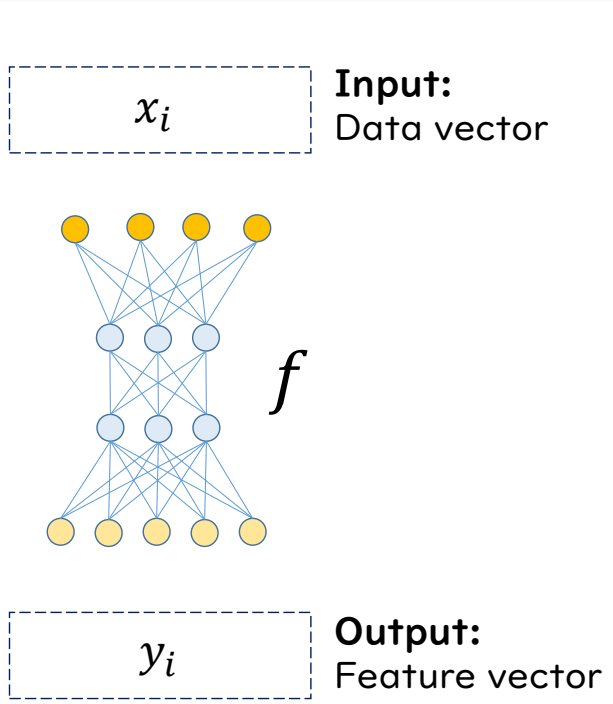
Given binary weights $w_{ij} \in \{0, 1\}$, Large-scale Information Network Embedding (Tang et al., 2015, LINE) learns a neural network

$$\boldsymbol{y}_i := \boldsymbol{f}(\boldsymbol{x}_i),$$

by assuming a probabilistic model[1]

$$w_{ij} \mid \boldsymbol{x}_i, \boldsymbol{x}_j \overset{\text{indep.}}{\sim} \text{Bernoulli}\left(\sigma(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle)\right),$$

where $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ is a neural network (NN). **NN can be trained by maximizing the log-likelihood** for $(\{w_{ij}\}_{1 \leq i < j \leq n} \mid \{\boldsymbol{x}_i\}_{i=1}^n)$ **without any constraint**.



$$\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$$

The log-likelihood can be maximized by stochastic algorithms.

- ☺ **computationally tractable**.
- ☺ **non-linear**.
- ☺ **inductive**, *i.e.*, unseen data vector $\boldsymbol{x}_{n+1}$ can be transformed.

[1] This model can be replaced, *e.g.*, Poisson distribution (Okuno et al., 2018).

Existing Graph Embedding Methods (Convolution)

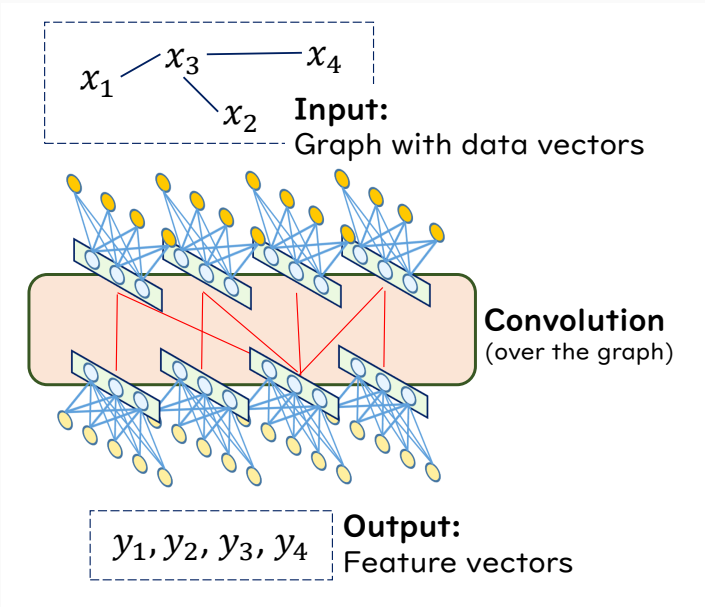
We **do not consider this model** in this study.

For binary weights $w_{ij} \in \{0, 1\}$,
Graph Autoencoder (Kipf and Welling, 2016, GAE) learns a graph convolutional network (Kipf and Welling, 2017, GCN)

$$(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n) = \mathrm{GCN}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n; \boldsymbol{W}),$$

so that

$$w_{ij} \approx \sigma(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle).$$



GCN: $\mathcal{X}^n \times \mathcal{W} \to \mathcal{Y}^n$

☺ **computationally tractable**.
☺ **non-linear**.
☺ **aggregate the information** by considering the graph structure.

☹ **not** inductive, *i.e.*, unseen data vector $\boldsymbol{x}_{n+1}$ **cannot** be transformed.[2]

[2] GraphSAGE Hamilton et al. (2017) and some recent studies try to make GCN inductive, by utilizing associations between the unseen data vector $\boldsymbol{x}_{n+1}$ and the learned vectors $\{\boldsymbol{x}_i\}_{i=1}^n$.

Existing Graph Embedding Methods

There are (mainly) 3 different types of graph embedding methods.

Spectral

Chung (1997),
Belkin and Niyogi (2003),
He and Niyogi (2004),...

☺ eigen-decomposition
☹ computationally hard
☹ limited to linear

Probabilistic

Mikolov et al. (2013),
Tang et al. (2015),...

☺ computationally tractable
☺ non-linear
☺ inductive

We employ this model.

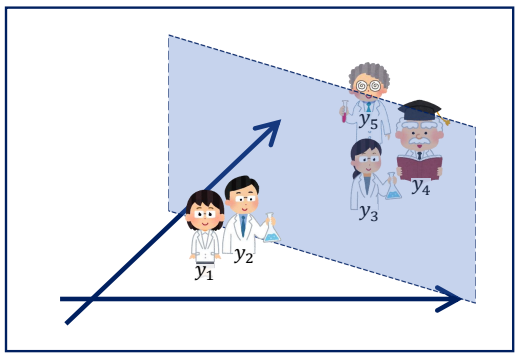
Convolution

Kipf and Welling (2016),
Kipf and Welling (2017),
Hamilton et al. (2017),...

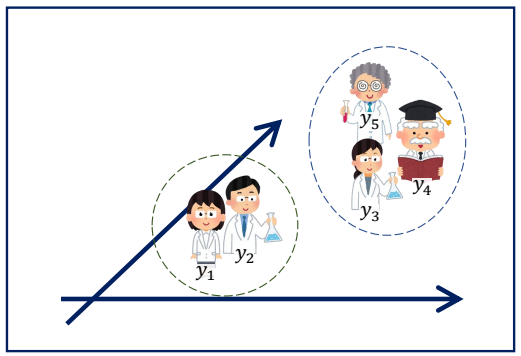
☺ computationally tractable
☺ non-linear
☺ graph convolution
☹ not inductive

Further Statistical Analyses

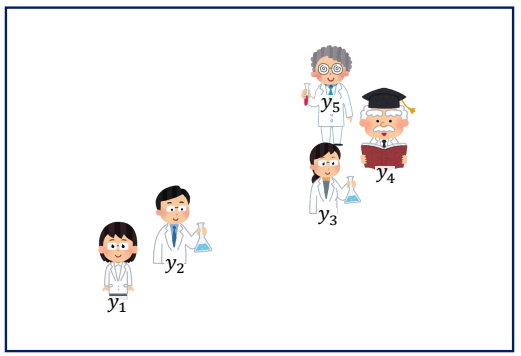
Further statistical analyses can be conducted on the vectors $\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathcal{Y}$.



(a) Discriminant analysis



(b) Clustering



(c) Visualization

Using obtained feature vectors have two advantages

- ☺ **computationally tractable** (usually, $\dim\mathcal{Y} \leq \dim\mathcal{X}$),
- ☺ expected to **improve the performance** of further analyses.

Obtained feature vectors $\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathcal{Y}(\subset \mathbb{R}^K)$ simultaneously preserves

1. the locality of $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathcal{X}(\subset \mathbb{R}^p)$, and
2. the association structure $\{w_{ij}\}_{1\leq i<j\leq n}$.

Examples: Graph with Data Vectors

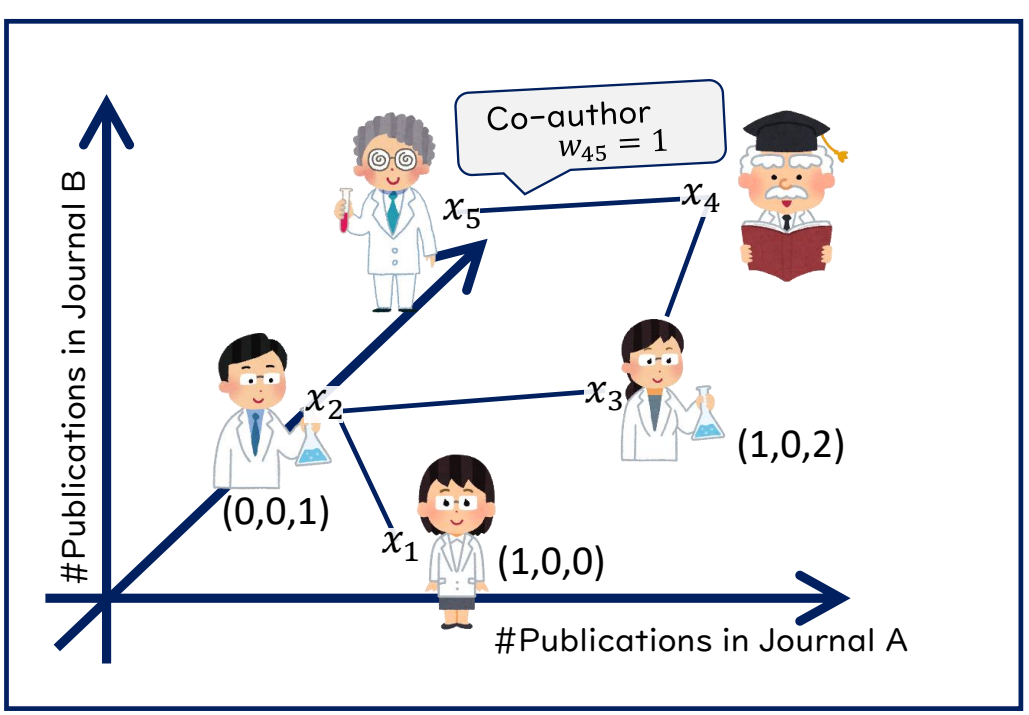


Figure: Co-authorship network

- $w_{ij} \in \{0, 1, 2, \ldots\}$: number of co-authored papers
- $\boldsymbol{x}_i \in \mathbb{N}_0^p$: number of publications in each of p conferences

$$\boldsymbol{x}_i := (\underset{\text{conf. A}}{2}, \underset{\text{conf. B}}{0}, \underset{\text{journal A}}{1}, \underset{\text{journal B}}{0}, \underset{\text{journal C}}{3}, \ldots) \in \mathbb{N}_0^p$$

Examples: Graph with Data Vectors

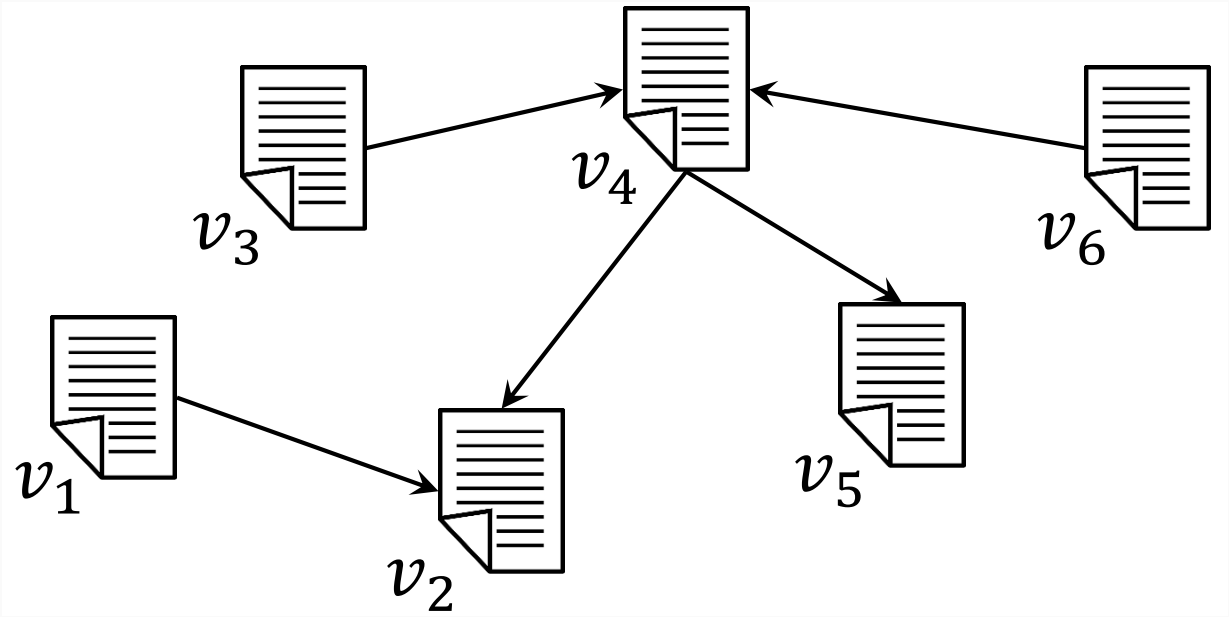


Figure: Citation network

- $w_{ij} \in \{0, 1\}$: whether *i*-th paper cites *j*-th paper [3]
- $\boldsymbol{x}_i \in \mathbb{N}_0^p$: bag-of-words (i.e., num. of each word) for each paper

$$\boldsymbol{x}_i = (\overset{\text{"the"}}{10}, \overset{\text{"he"}}{0}, \overset{\text{"she"}}{0}, \overset{\text{"animal"}}{2}, \overset{\text{"fish"}}{1}, \overset{\text{"dog"}}{0}, \ldots) \in \mathbb{N}_0^{\sharp\text{dictionary}}$$

[3] directed relation can be transformed to undirected relation.

Examples: Graph with Data Vectors

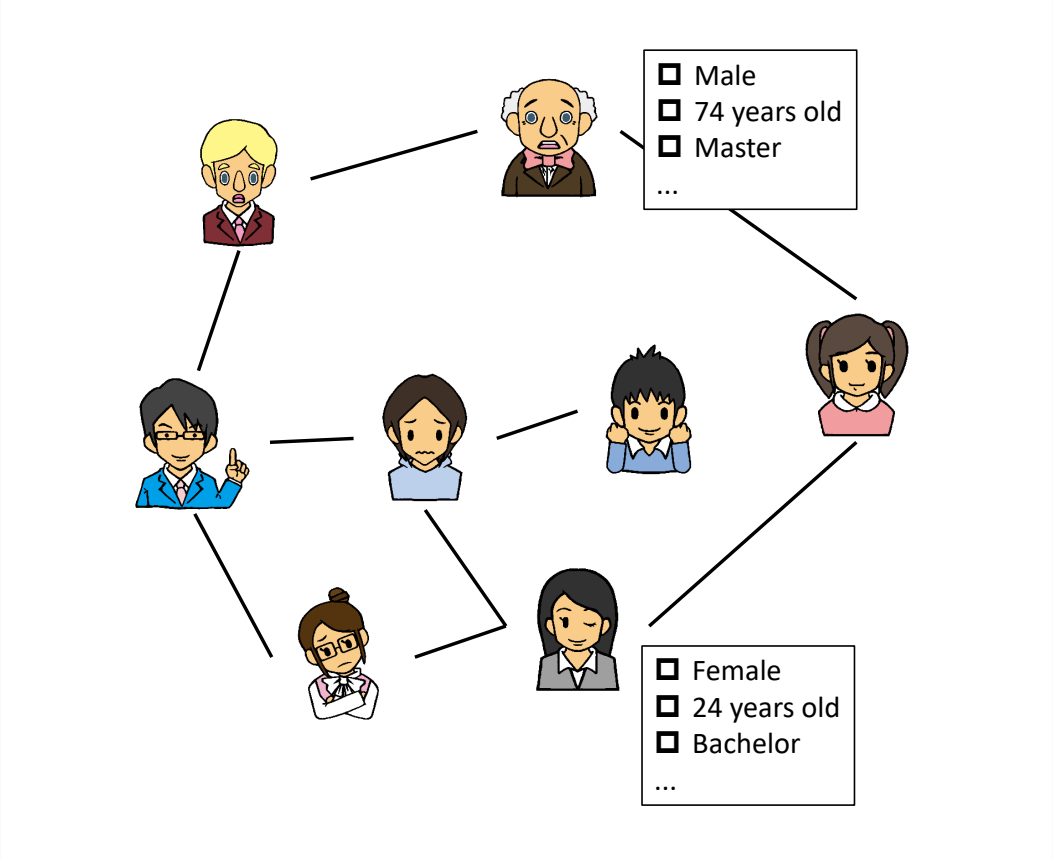


Figure: Friend network

- $w_{ij} \in \{0, 1\}$: friendship
- $\boldsymbol{x}_i \in \mathbb{R}^p$: property vector (e.g., age, born, degree, etc..)

Examples: Graph with Data Vectors

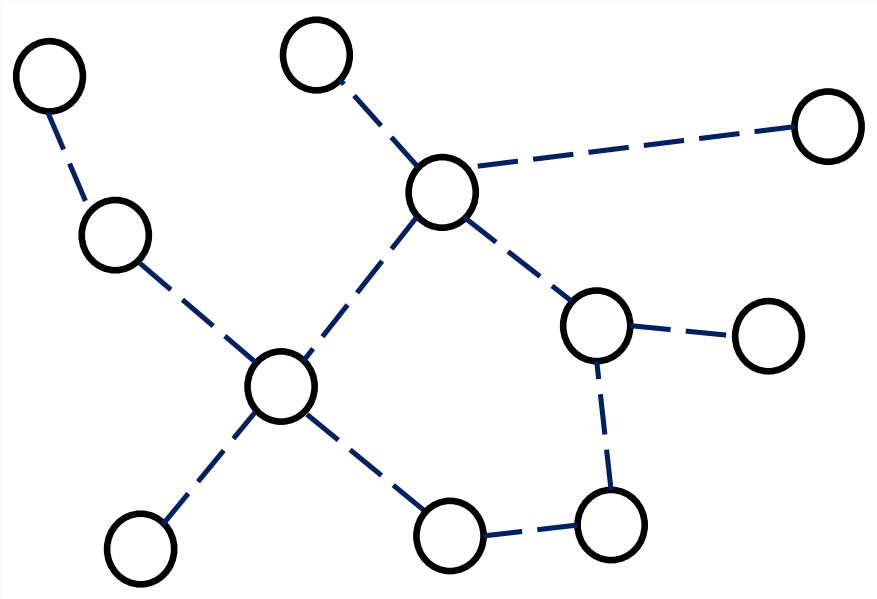


Figure: A general graph

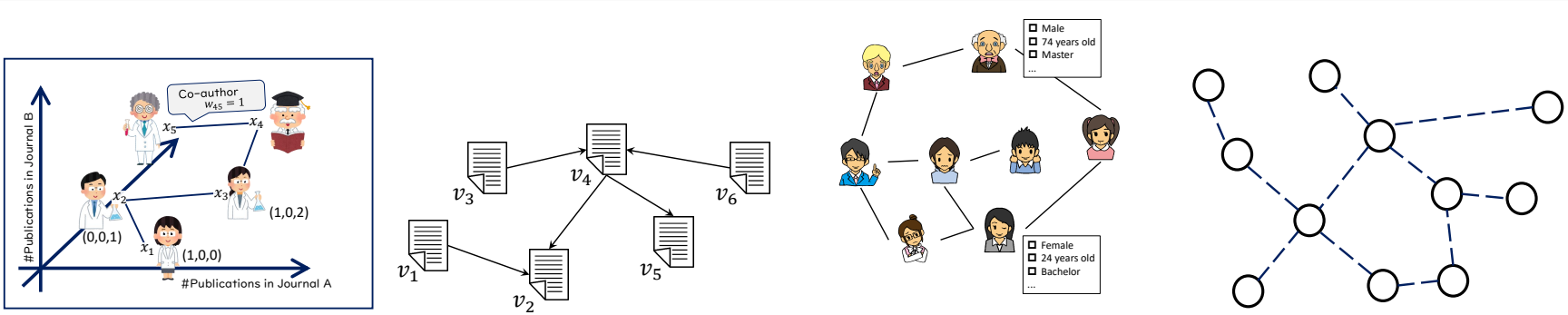
- $w_{ij} \geq 0$: Link weights
- no data vector: $\triangleright$ 1-hot vector

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

can be used instead. (or, $\boldsymbol{x}_i = \boldsymbol{w}_i := (w_{i1}, w_{i2}, \ldots, w_{in})$ is used in recommendation systems)

Summary so far

Given link weights $\{w_{ij}\}_{1 \le i < j \le n}$ and data vectors $\{\boldsymbol{x}_i\}_{i=1}^n$, *e.g.*,



(a) Co-author network (b) Citation network (c) Friend network (d) General graph

graph embedding learns a neural network $\boldsymbol{f} : \mathcal{X} \ni \boldsymbol{x}_i \mapsto \boldsymbol{y}_i \in \mathcal{Y}$, by maximizing the log-likelihood with a probabilistic model

$$w_{ij} \mid \boldsymbol{x}_i, \boldsymbol{x}_j \overset{\text{indep.}}{\sim} \text{Bernoulli}\left(\sigma\left(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle\right)\right).$$

(▷ inner-product similarity has been used,
cf., Spectral: $\sum_{i,j=1}^n w_{ij} \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle$, and Convlution: $w_{ij} \approx \sigma(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle)$.)

PD and CPD similarities (AISTATS2019)

Poincaré Embedding

Whereas many of graph embedding trains a neural network $\boldsymbol{f} : \mathcal{X} \ni \boldsymbol{x}_i \mapsto \boldsymbol{y}_i \in \mathcal{Y}$, by maximizing the log-likelihood with a probabilistic model

$$w_{ij} \mid \boldsymbol{x}_i, \boldsymbol{x}_j \overset{\text{indep.}}{\sim} \text{Bernoulli}\left(\sigma\left(\langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle\right)\right),$$

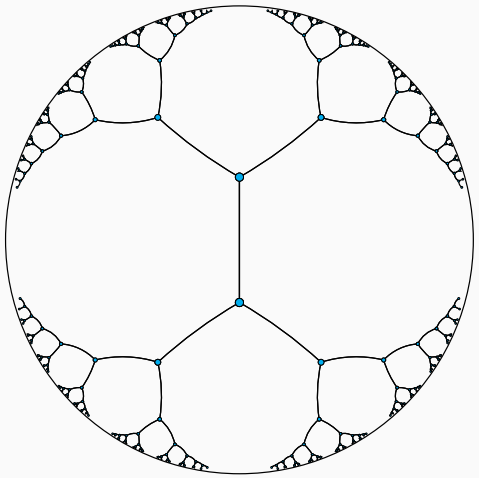
Poincaré embedding (Nickel and Kiela, 2017)[4] considers

$$w_{ij} \mid \boldsymbol{x}_i, \boldsymbol{x}_j \overset{\text{indep.}}{\sim} \text{Bernoulli}\left(\sigma\left(-d_{\text{Poincare}}(\boldsymbol{y}_i, \boldsymbol{y}_j)\right)\right),$$

where $\mathcal{Y} := \{\boldsymbol{y} \in \mathbb{R}^p \mid \|\boldsymbol{y}\|_2 < 1\}$,

$$d_{\text{Poincare}}(\boldsymbol{y}_i, \boldsymbol{y}_j) := \cosh^{-1}\left(1 + 2\frac{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2}{(1 - \|\boldsymbol{y}_i\|_2^2)(1 - \|\boldsymbol{y}_j\|_2^2)}\right),$$

$$\cosh^{-1}(z) := \log(z + \sqrt{z^2 - 1}).$$



Tree (=graph) can be efficiently embedded with Poincaré distance. (Nickel and Kiela, 2017, Fig. 1(b))

[4] Lorentz embedding (Nickel and Kiela, 2018) utilizes Lorentz model instead, for efficient computation.

General Graph Embedding Framework

(Probabilistic) graph embedding can be generalized as follows:

General graph embedding

General graph embedding can be obtained by maximizing the log-likelihood of a model

$$w_{ij} \mid \boldsymbol{x}_i, \boldsymbol{x}_j \overset{\text{indep.}}{\sim} Q(\nu(\underbrace{g(\boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j))}_{\text{(similarity)}})), \quad \boldsymbol{y}_i := \boldsymbol{f}(\boldsymbol{x}_i),$$

where

- $Q(\nu)$ is a **user-specified** distribution whose expectation is $\nu \in \mathbb{R}$,
- $\nu : \mathbb{R} \to \mathbb{R}$ is a **user-specified** function,
- $g : \mathcal{Y}^2 \to \mathbb{R}$ is a **user-specified** kernel (*i.e.*, symmetric and continuous function), and
- $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ is a neural network **to be estimated**.

- Tang et al. (2015) Q: Bernoulli, ν: Sigmoid, g: inner-product.
- Nickel and Kiela (2017) Q: Bernoulli, ν: Sigmoid, g: negative Poincaré.
- Okuno et al. (2018) Q: Poisson, ν: Exponential, g: inner-product. etc...

“Expressive power" of graph embedding

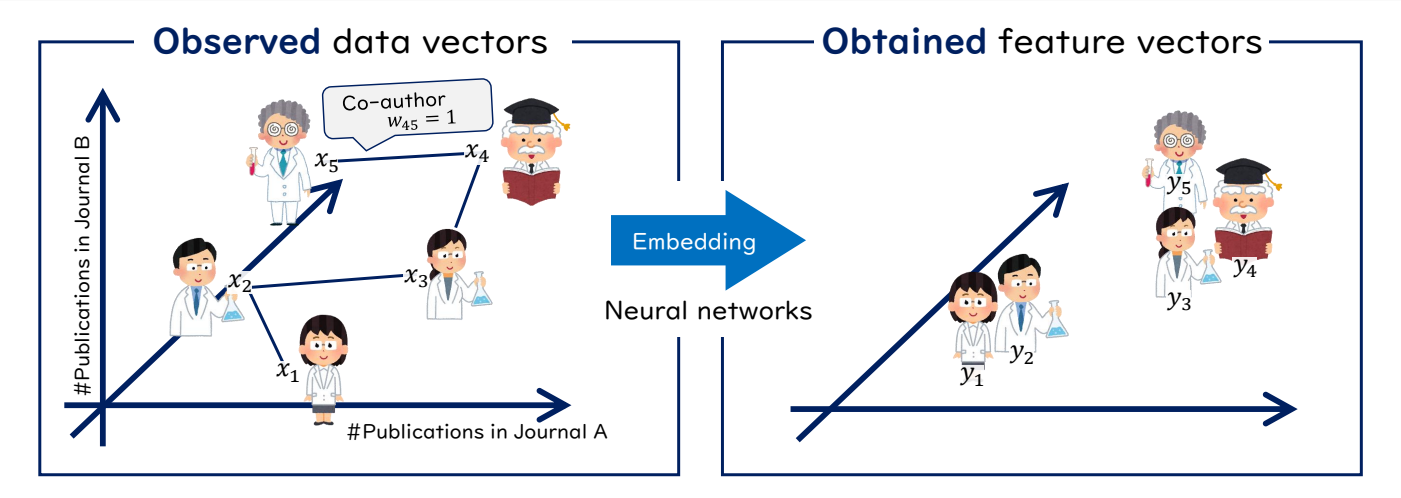


Figure: Graph embedding revisited.

Expressive power of graph embedding

$\Leftrightarrow$ Expressive power of the **similarity function**[5]

$$h(\boldsymbol{x}_i, \boldsymbol{x}_j) := g(\boldsymbol{y}_i, \boldsymbol{y}_j) = g(\boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j)),$$

equipped with the neural network $\boldsymbol{f}$ and the user-specified kernel g.

[5] Similarity between two NNs $g(\boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j))$ is also known as *siamese network* (Bromley et al., 1994).

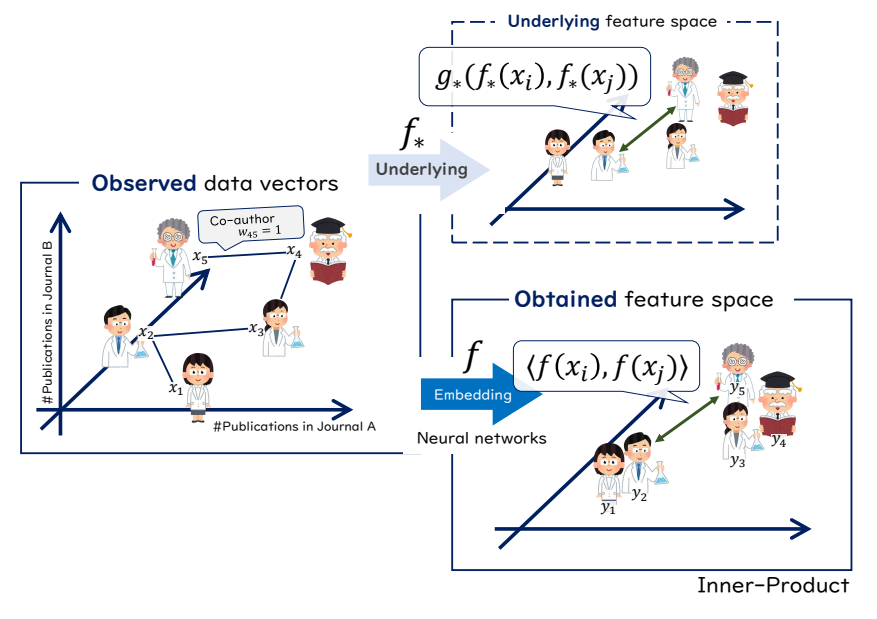
Inner-Product Similarity (IPS)

Okuno et al. (2018) Theorem 5.1, informal

$g_* : \mathcal{Y}_*^2 \to \mathbb{R}$ is a **PD** kernel [6] and $\boldsymbol{f}_* : \mathcal{X} \to \mathcal{Y}_*$ is a continuous function. Then,

$$g_*(\boldsymbol{f}_*(\boldsymbol{x}_i), \boldsymbol{f}_*(\boldsymbol{x}_j)) \approx \langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j) \rangle,$$

for some sufficiently large neural network $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y} \subset \mathbb{R}^K$ with sufficiently large K.



[6] kernel (*i.e.*, continuous and symmetric function) $g : \mathcal{Y}^2 \to \mathbb{R}$ satisfying $\sum_{i,j=1}^n c_i c_j g(\boldsymbol{y}_i, \boldsymbol{y}_j) \geq 0$ for all $\{\boldsymbol{y}_i\}_{i=1}^n \subset \mathcal{Y}$, $\{c_i\}_{i=1}^n \subset \mathbb{R}$ is called positive-definite (PD).

Inner-Product Similarity (IPS)

Proof: Mercer's theorem (Minh et al., 2006) indicates that

$$g_*(\boldsymbol{f}_*(\boldsymbol{x}_i), \boldsymbol{f}_*(\boldsymbol{x}_j)) \overset{\text{Mercer}}{=} \langle \boldsymbol{\Psi}(\boldsymbol{x}_i), \boldsymbol{\Psi}(\boldsymbol{x}_j) \rangle$$

for some $\boldsymbol{\Psi}$. Considering the top-K entries $\boldsymbol{\Psi}_K$, universal approximation theorem for NN (Cybenko, 1989) proves that $\boldsymbol{\Psi}_K \approx \boldsymbol{f}$ for some NN $\boldsymbol{f}$; thus

$$\langle \boldsymbol{\Psi}(\boldsymbol{x}_i), \boldsymbol{\Psi}(\boldsymbol{x}_j) \rangle \approx \langle \boldsymbol{\Psi}_K(\boldsymbol{x}_i), \boldsymbol{\Psi}_K(\boldsymbol{x}_j) \rangle \approx \langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j) \rangle.$$

□

The inner-product similarity $\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j) \rangle$ with neural network $\boldsymbol{f}$ is

- ☺ capable of approximating any PD similarity $g_*(\boldsymbol{f}_*(\boldsymbol{x}_i), \boldsymbol{f}_*(\boldsymbol{x}_j))$.
- ☺ easy to compute.
- ☹ **limited to** approximate **PD** similarities.

Non-PD Similarities

- **Negative Poincaré distance** is not PD.

$$d_{\text{Poincare}}(\boldsymbol{y}_i, \boldsymbol{y}_j) := \cosh^{-1}\left(1 + 2\frac{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2}{(1 - \|\boldsymbol{y}_i\|_2^2)(1 - \|\boldsymbol{y}_j\|_2^2)}\right),$$

$$\cosh^{-1}(z) := \log(z + \sqrt{z^2 - 1}).$$

- **Negative squared distance (NSD)** is not PD.

$$d_{\text{NSD}}(\boldsymbol{y}_i, \boldsymbol{y}_j) := -\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$$

- **Negative Wasserstein distance** is not PD, etc...

Non-PD Similarities

- **Negative Poincaré distance** is not PD.

$$d_{\text{Poincare}}(\boldsymbol{y}_i, \boldsymbol{y}_j) := \cosh^{-1}\left(1 + 2\frac{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2}{(1 - \|\boldsymbol{y}_i\|_2^2)(1 - \|\boldsymbol{y}_j\|_2^2)}\right),$$

$$\cosh^{-1}(z) := \log(z + \sqrt{z^2 - 1}).$$

- **Negative squared distance (NSD)** is not PD.

$$d_{\text{NSD}}(\boldsymbol{y}_i, \boldsymbol{y}_j) := -\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2$$

- **Negative Wasserstein distance** is not PD, etc...

Okuno et al. (2019) Proposition 4.1

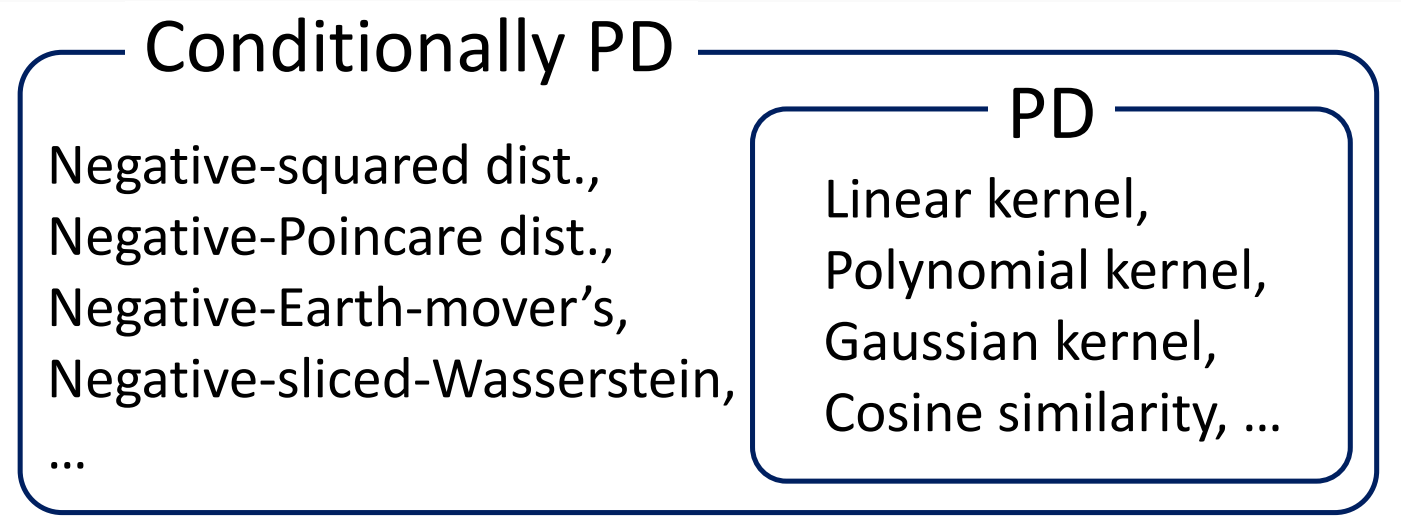
For all $M > 0, p, K \in \mathbb{N}$, and for any continuous functions $\boldsymbol{f} : \mathbb{R}^p \to \mathbb{R}^K$,

$$\frac{1}{(2M)^{2p}} \iint_{[-M,M]^{2p}} \left| \underbrace{-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}_{\text{non-PD}} - \langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}') \rangle \right| \mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{x}' \geq \frac{2pM^2}{3}.$$

Shifted Inner-Product Similarity (SIPS)

Conditionally PD (CPD) (Berg et al., 1984)

A symmetric function $h : \mathcal{X}^2 \to \mathbb{R}$ is called **conditionally-PD (CPD)** if $\sum_{i,j=1}^n c_i c_j h(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$ for all $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathcal{X}, \{c_i\}_{i=1}^n \subset \mathbb{R}$ satisfying $\sum_{i=1}^n c_i = 0$.



Shifted Inner-Product Similarity (SIPS)

Shifted Inner-Product Similarity (SIPS) (Okuno et al., 2019)

Proposed SIPS is defined as

$$\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j) \rangle + u(\boldsymbol{x}_i) + u(\boldsymbol{x}_j),$$

where $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ and $u : \mathcal{X} \to \mathbb{R}$ are neural networks.

SIPS is capable of approximating any CPD similarities as follows.

Okuno et al. (2019) Theorem 4.1, informal

$g_* : \mathcal{Y}_*^2 \to \mathbb{R}$ is a **CPD** kernel and $\boldsymbol{f}_* : \mathcal{X} \to \mathcal{Y}_*$ is a continuous function. Then,

$$g_*(\boldsymbol{f}_*(\boldsymbol{x}_i), \boldsymbol{f}_*(\boldsymbol{x}_j)) \approx \langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j) \rangle + u(\boldsymbol{x}_i) + u(\boldsymbol{x}_j),$$

for some sufficiently large neural networks $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y} \subset \mathbb{R}^K, u : \mathcal{X} \to \mathbb{R}$ with sufficiently large K.

Shifted Inner-Product Similarity (SIPS)

Proof is based on the following Lemma.

Berg et al. (1984) Lemma 2.1

$g_* : \mathcal{Y}^2 \to \mathbb{R}$ is CPD, if and only if

$$\tilde{g}_*(\boldsymbol{y}, \boldsymbol{y}') := g_*(\boldsymbol{y}, \boldsymbol{y}') - g_*(\boldsymbol{y}, \boldsymbol{y}_0) - g_*(\boldsymbol{y}_0, \boldsymbol{y}') + g_*(\boldsymbol{y}_0, \boldsymbol{y}_0)$$

with a $\boldsymbol{y}_0 \in \mathcal{Y}$ is PD.

For any CPD kernel g_*, we have

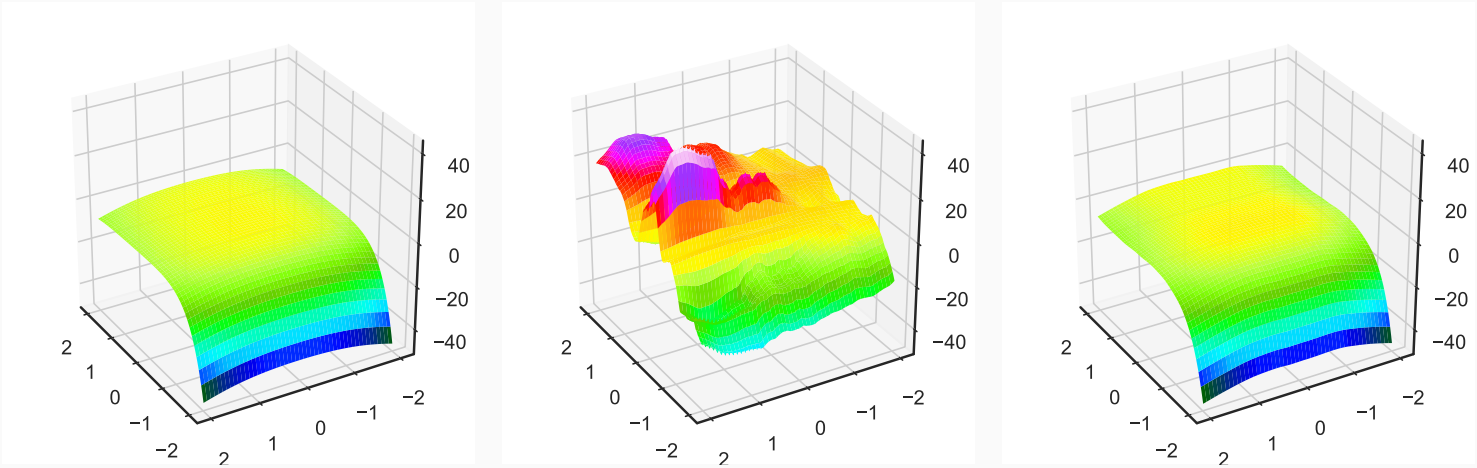
$$g_*(\boldsymbol{y}, \boldsymbol{y}') = \tilde{g}_*(\boldsymbol{y}, \boldsymbol{y}') + \underbrace{\left(g_*(\boldsymbol{y}, \boldsymbol{y}_0) - \frac{1}{2} g_*(\boldsymbol{y}_0, \boldsymbol{y}_0)\right)}_{=: r(\boldsymbol{y})} + \underbrace{\left(g_*(\boldsymbol{y}_0, \boldsymbol{y}') - \frac{1}{2} g_*(\boldsymbol{y}_0, \boldsymbol{y}_0)\right)}_{=: r(\boldsymbol{y}')}.$$

Substituting $\boldsymbol{y}_i := \boldsymbol{f}_*(\boldsymbol{x}_i)$ leads to

$$g_*(\boldsymbol{f}_*(\boldsymbol{x}_i), \boldsymbol{f}_*(\boldsymbol{x}_j)) = \underbrace{\tilde{g}_*(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}'))}_{\text{PD}} + r(\boldsymbol{f}_*(\boldsymbol{x})) + r(\boldsymbol{f}_*(\boldsymbol{x}')).$$

Approximating the terms in right-hand-side by $\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}') \rangle, u(\boldsymbol{x}), u(\boldsymbol{x}')$, respectively, lead to the assertion. □

Numerical Experiments (Okuno and Shimodaira, 2018)



(a) True (b) IPS (c) SIPS

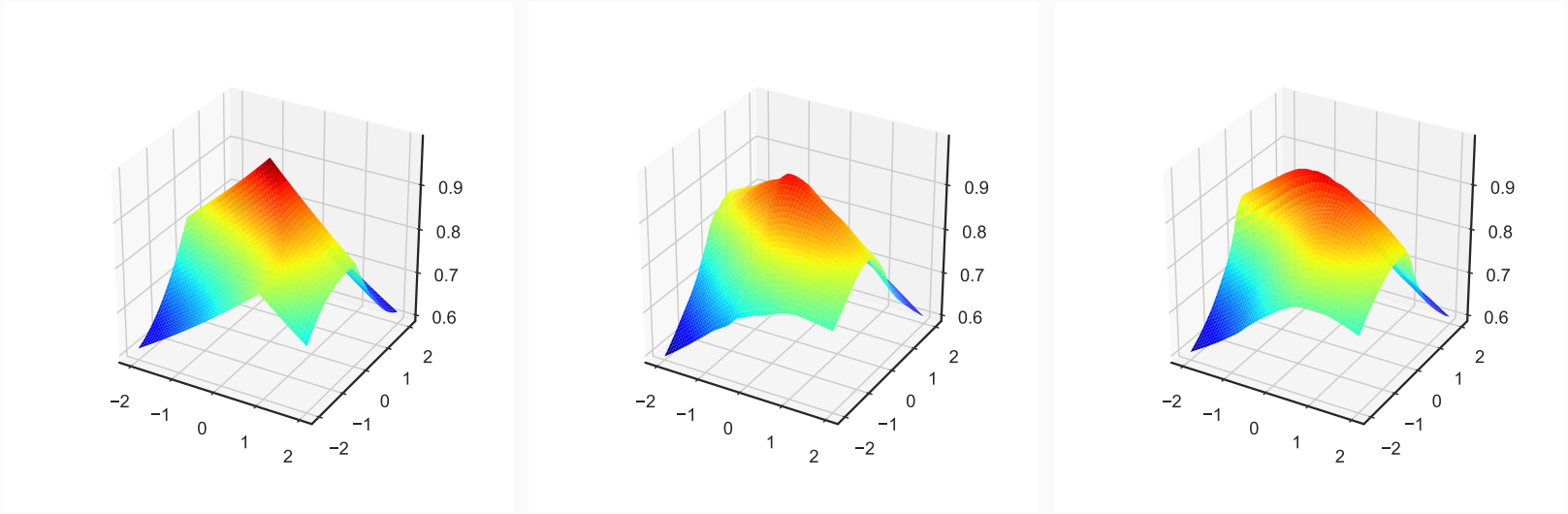
Figure: SIPS can but IPS cannot approximate negative-squared distance (CPD).

For $\boldsymbol{f}_*(\boldsymbol{x}) = (x_1, cos x_2, exp(-x_3), sin(x_4 - x_5)) \in \mathbb{R}^4$ w.r.t. $\boldsymbol{x} \in \mathbb{R}^5$,

(a) $-\|\boldsymbol{f}_*(s\boldsymbol{e}_1) - \boldsymbol{f}_*(t\boldsymbol{e}_2)\|_2^2$ (**CPD**),
(b) trained IPS $\langle \boldsymbol{f}(s\boldsymbol{e}_1), \boldsymbol{f}(t\boldsymbol{e}_2) \rangle$,
(c) trained SIPS $\langle \boldsymbol{f}(s\boldsymbol{e}_1), \boldsymbol{f}(t\boldsymbol{e}_2) \rangle + u(s\boldsymbol{e}_1) + u(t\boldsymbol{e}_2)$,

are plotted on (s, t)-plane along with two orthogonal directions $\boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{R}^5$.
$\boldsymbol{f} : \mathbb{R}^5 \to \mathbb{R}^{10}$ and $u : \mathbb{R}^5 \to \mathbb{R}$ are two-layer neural networks with 1,000 hidden units and ReLU activation.

Numerical Experiments (Okuno and Shimodaira, 2018)



(a) True (b) IPS (c) SIPS

Figure: Both of IPS and SIPS can approximate Laplace kernel (PD).

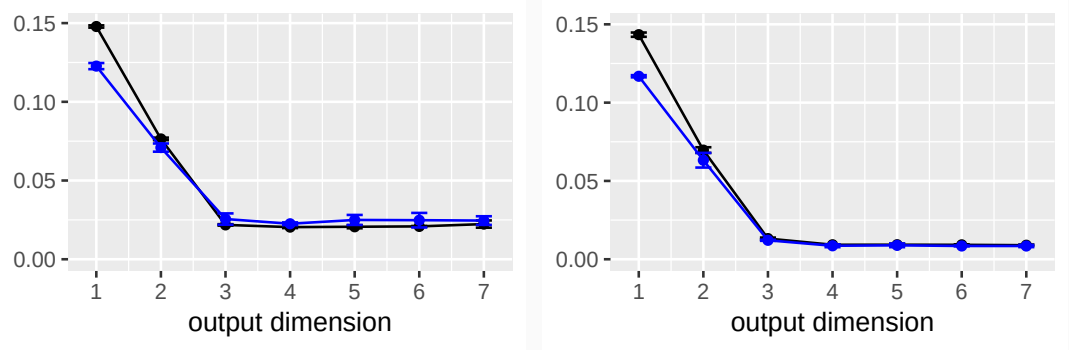
As well,

(a) Laplace kernel $\exp(-\|\boldsymbol{f}_*(s\boldsymbol{e}_1) - \boldsymbol{f}_*(t\boldsymbol{e}_2)\|_1)$ (**PD**),
(b) trained IPS $\langle \boldsymbol{f}(s\boldsymbol{e}_1), \boldsymbol{f}(t\boldsymbol{e}_2) \rangle$,
(c) trained SIPS $\langle \boldsymbol{f}(s\boldsymbol{e}_1), \boldsymbol{f}(t\boldsymbol{e}_2) \rangle + u(s\boldsymbol{e}_1) + u(t\boldsymbol{e}_2)$,

are plotted on (s, t)-plane.

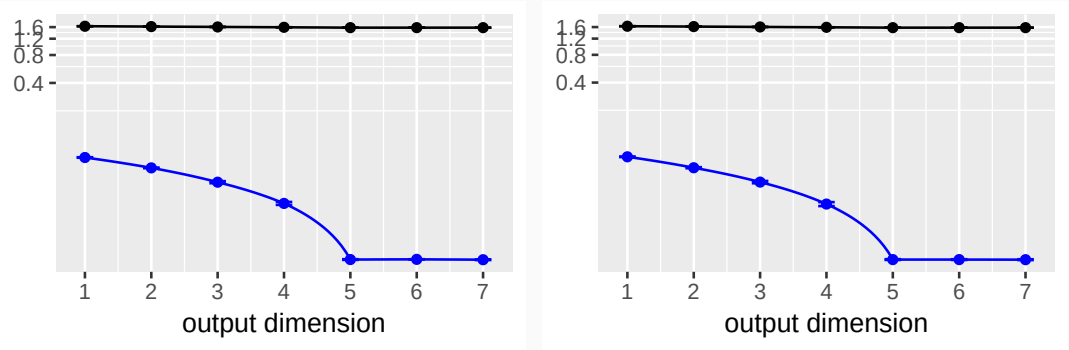
Numerical Experiments (Okuno and Shimodaira, 2018)

Prediction Mean Squared Error (PMSE) for approximating similarities. IPS (Black) and SIPS (Blue).



(a) ♯hidden units=100 (b) ♯hidden units=1000

Figure: Cosine similarity (PD)



(a) ♯hidden units=100 (b) ♯hidden units=1000

Figure: negative Poincaré distance (CPD)

Approximation Error Rate

Okuno et al. (2019) Theorem 5.2 with $\alpha = 1$, informal

For any

- compact sets $\mathcal{X} \subset \mathbb{R}^p, \mathcal{Y}_* \subset \mathbb{R}^{K_*}$,
- CPD kernel $g_* \in C^1(\mathcal{Y}_*^2; \mathbb{R})$, and
- $\boldsymbol{f}_* \in C^1(\mathcal{X}; \mathcal{Y}_*)$,

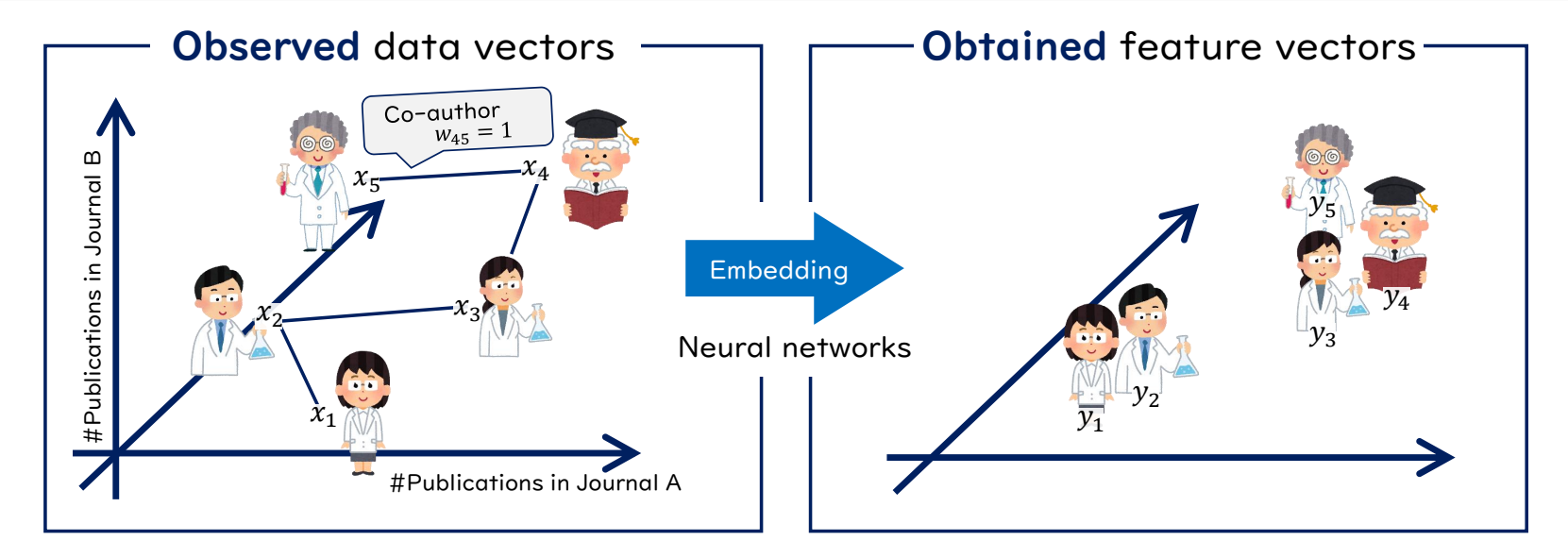
SIPS with ReLU-activated deep MLPs $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^K, u : \mathcal{X} \to \mathbb{R}$ whose depths are $O(L_f), O(L_u)$ and widths are $O(1)$, uniformly approximates $g_*(\boldsymbol{f}_*(\boldsymbol{x}), \boldsymbol{f}_*(\boldsymbol{x}'))$ with an error

$$O\left(\underbrace{K^{-1/K_*}}_{(\star)} + K^{1/2} L_f^{-2/p} + L_u^{-2/p}\right).$$

Proof is straightforwardly obtained by applying Cobos and Kühn (1990) and Yarotsky (2018). See Okuno et al. (2019) Supplement D.3 with $\alpha = 1$. □

Note: we showed only the comprehensible error rate; the above error rate can be improved by employing other existing studies.

Summary of this study



Graph embedding learns a transformation $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$, so that

$$w_{ij} \approx \mathrm{Sigmoid}(\underbrace{g(\boldsymbol{y}_i, \boldsymbol{y}_j)}_{\text{“Similarity”}}), \quad \boldsymbol{y}_i := \boldsymbol{f}(\boldsymbol{x}_i).$$

g is specified as inner-product (Tang et al., 2015), Poincaré-dist. (Nickel and Kiela, 2017), etc..

Question ▷ What kind of kernel g is good (in terms of the expressive power?)

Answer ▷ Proposed **shifted inner-product similarity (SIPS)**.

SIPS can approximate many kernels, e.g., Poincaré-distance!

Recent Progress: General Similarities (arXiv:1902.10409)

Non-CPD Similarities

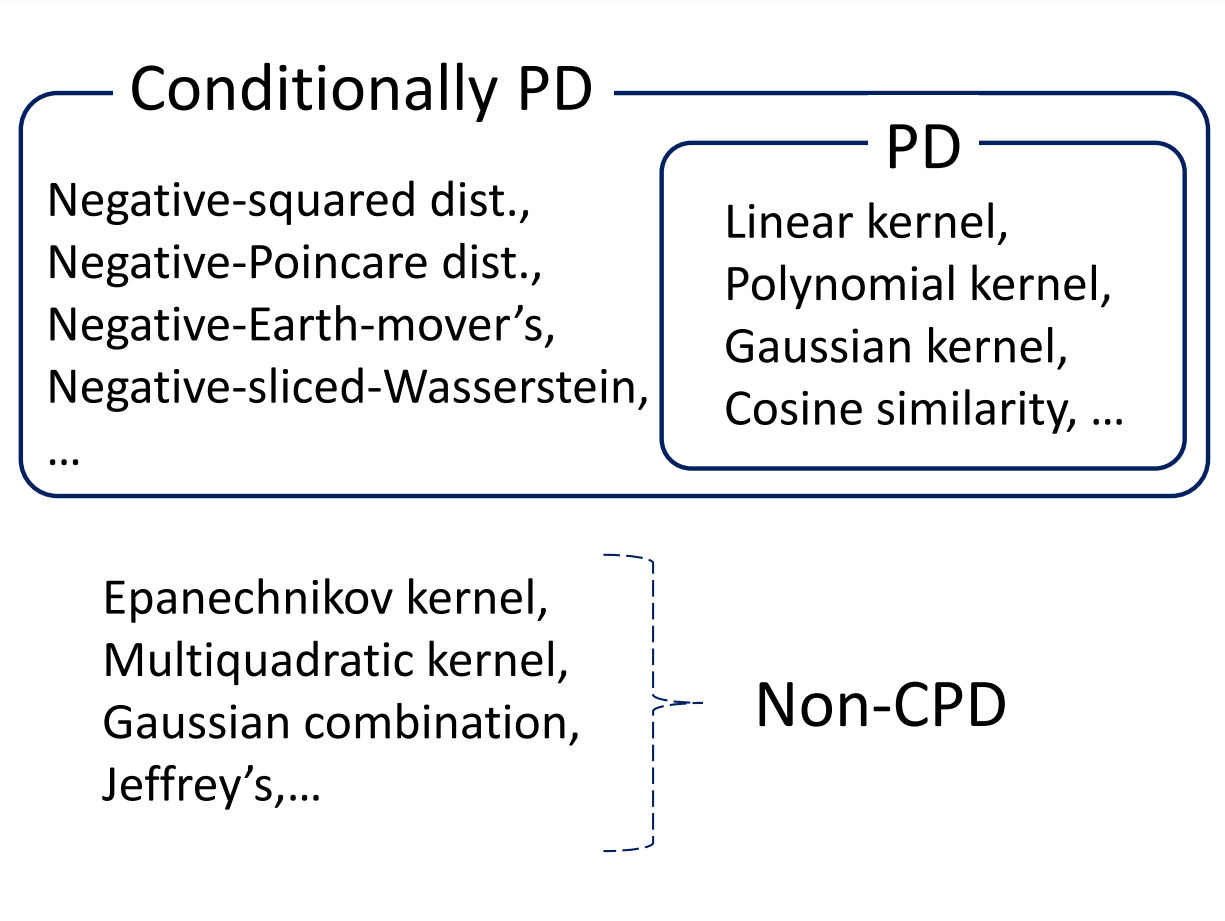


Figure: There remain non-CPD similarities, that cannot be approximated by SIPS.

Inner-Product Difference Similarity (IPDS)

General similarity $g_*(\boldsymbol{f}_*(\boldsymbol{x}_i), \boldsymbol{f}_*(\boldsymbol{x}_j))$ can be decomposed as

$$g_*(\boldsymbol{f}_*(\boldsymbol{x}_i), \boldsymbol{f}_*(\boldsymbol{x}_j)) = \underbrace{h_+(\boldsymbol{x}_i, \boldsymbol{x}_j)}_{\text{PD}} - \underbrace{h_-(\boldsymbol{x}_i, \boldsymbol{x}_j)}_{\text{PD}},$$

and PD similarities h_+, h_- can be approximated by IPS.

Inner-Product Difference Similarity (IPDS) (Okuno et al., 2019, Supplement E)

IPDS is defined as

$$\langle \boldsymbol{f}^+(\boldsymbol{x}_i), \boldsymbol{f}^+(\boldsymbol{x}_j) \rangle - \langle \boldsymbol{f}^-(\boldsymbol{x}_i), \boldsymbol{f}^-(\boldsymbol{x}_j) \rangle,$$

where $\boldsymbol{f}^+ : \mathcal{X} \to \mathcal{Y}^+ \subset \mathbb{R}^{K_+}, \boldsymbol{f}^- : \mathcal{X} \to \mathcal{Y}^- \subset \mathbb{R}^{K_-}$ are neural networks.

IPDS is capable of approximating general similarities[7].

☺ IPDS includes SIPS as a special case[8].
☹ requires to specify the rate K_+/K_-.

[7] more precisely, similarities dominated by PD similarity. see Ong et al. (2004)
[8] $\boldsymbol{f}^+(\boldsymbol{x}) := (\boldsymbol{f}(\boldsymbol{x})^\top, u(\boldsymbol{x}), 1)^\top, \boldsymbol{f}^-(\boldsymbol{x}) := u(\boldsymbol{x}) - 1.$

Weighted Inner-Product Similarity (WIPS)

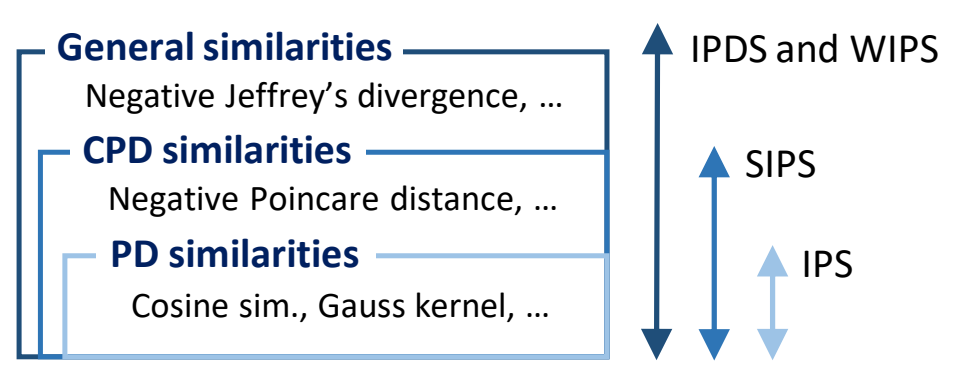
Weighted Inner-Product Similarity (WIPS) (Kim et al., 2019)

WIPS is defined as

$$\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j) \rangle_{\boldsymbol{\lambda}},$$

where $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ is a neural network, $\langle \boldsymbol{y}, \boldsymbol{y}' \rangle_{\boldsymbol{\lambda}} = \sum_{k=1}^{K} \lambda_k y_k y'_k$, and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K) \in \mathbb{R}^K$ is a weight vector **to be estimated**.

☺ WIPS using $\boldsymbol{\lambda} := (1, 1, \ldots, 1, -1, -1, \ldots, -1)$ reduces to IPDS.



☺ no need to specify the rate K_+/K_-.

Numeraical Experiments (Kim et al., 2019)

In this numerical experiment, we embed **WebKB hypertext network** [9] equipped with 1, 409-dim. Bag-of-Words data vectors.

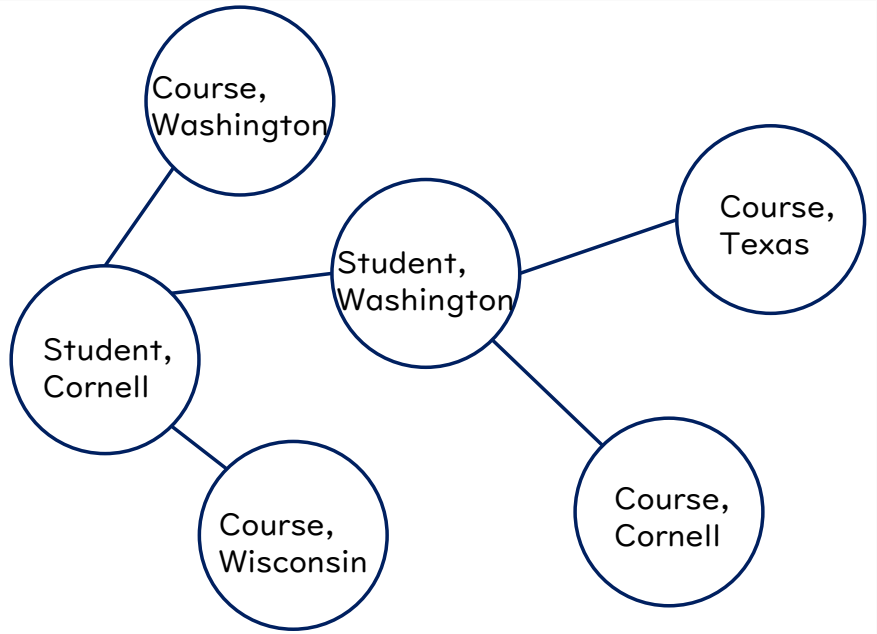


Figure: 877 hypertexts (nodes) are associated by 1, 480 hyperlinks (links).

Each node has semantic class labels

- one of {*Student, Faculty, Staff, Course, Project*},
- one of university labels in {*Cornell, Texas, Washington, Wisconsin*}.

[9] https://linqs.soe.ucsc.edu/data

Numeraical Experiments (Kim et al., 2019)

Obtained 10-dim. feature vectors are mapped to $\mathbb{R}^2$ with *t*-SNE (Maaten and Hinton, 2008).

Semantic labels are colored as

- Student (navy), Course (pink),
- Cornell (red), Texas (orange), Washington (green) and Wisconsin (blue).

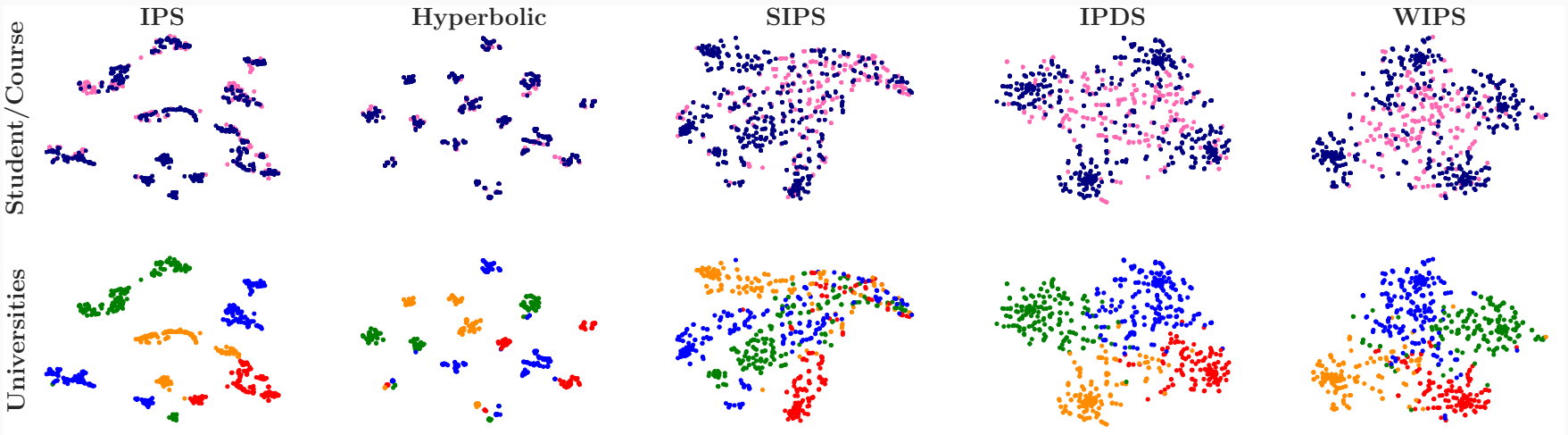


Figure: Selected nodes of embeded WebKB hypertext network using *t*-SNE.

Both class labels are clearly identified with IPDS and WIPS, whereas they become obscure in the other embeddings.

Numerical Experiments (Kim et al., 2019)

We conducted following experiments on **WebKB hypertext** network.

- **Reconstruction** of link weights from fully observed data,
- **Link prediction** of unseen nodes.

Table: **ROC-AUC**. Boldface is the best, and underlines are 2nd and 3rd best scores.

		Dimensionality					
		Reconstruction			Link prediction		
		10	50	100	10	50	100
Hypertext	IPS	91.99	94.23	94.24	77.73	77.62	77.16
	Poincaré	94.09	94.13	94.11	82.21	79.64	79.48
	SIPS	95.11	95.12	95.12	82.01	81.84	81.13
	IPDS	**95.12**	95.12	**95.12**	**82.59**	**82.75**	82.19
	WIPS	95.11	**95.12**	95.12	82.38	82.68	**82.93**

Numerical Experiments (Kim et al., 2019)

Same experiments are conducted on

- **DBLP Co-authorship network** (Prado et al., 2013)
  ▷ 41, 328 nodes, 210, 320 links, 33-dim. data vectors.
- **WordNet Taxonomy Tree** (Nickel and Kiela, 2017)
  ▷ 37, 623 nodes, 312, 885 links, 300-dim. pre-trained word vectors.

Table: **ROC-AUC**. Boldface is the best, and underlines are 2nd and 3rd best scores.

		Dimensionality					
		Reconstruction			Link prediction		
		10	50	100	10	50	100
Co-author	IPS	85.01	86.02	85.80	83.83	84.41	84.02
	Poincaré	86.84	86.69	86.72	85.82	85.92	85.93
	SIPS	90.01	91.35	91.06	88.24	88.69	88.67
	IPDS	90.13	91.68	91.59	**88.42**	88.97	88.85
	WIPS	**90.50**	**92.44**	**92.95**	88.16	**89.43**	**89.40**
Taxonomy	IPS	79.95	75.80	74.97	67.25	65.71	65.38
	Poincaré	91.69	89.10	88.97	83.04	79.52	78.97
	SIPS	98.78	99.75	99.77	90.42	92.12	92.09
	IPDS	**99.65**	**99.89**	**99.90**	**95.99**	**96.37**	96.41
	WIPS	99.64	99.85	99.87	95.07	96.36	**96.51**

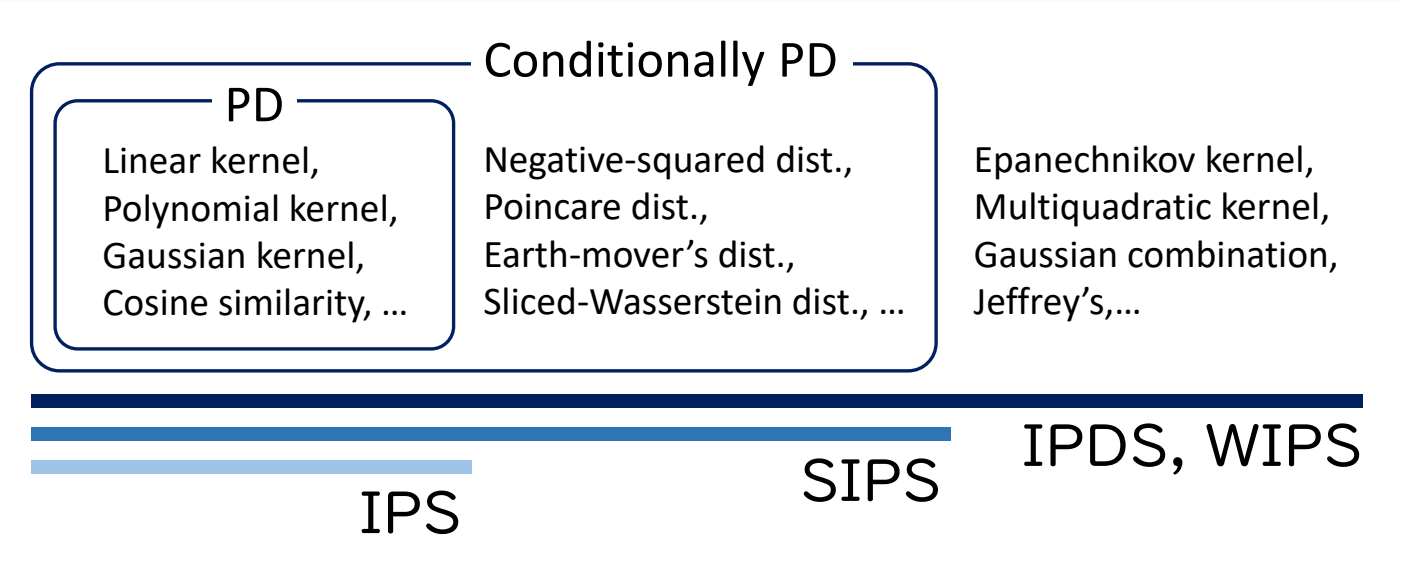
Conclusion

Conclusion

(1) As the similarity g, we propose **SIPS** and **IPDS** in Okuno et al. (2019), that are highly expressive compared with the simple inner-product similarity.

(2) We prove that **SIPS approximates any CPD similarities**, and IPDS approximates general similarities.

(3) We also propose **WIPS** in Kim et al. (2019), that generalizes IPDS.



(4) Experiments show the high expressive power of SIPS, IPDS, and WIPS.

References I

Belkin, M. and Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373–1396.

Berg, C., Christensen, J. P. R., and Ressel, P. (1984). Harmonic Analysis on Semigroups.

Bojchevski, A. and Günnemann, S. (2018). Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a “siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744.

Chung, F. R. (1997). *Spectral Graph Theory*. Number 92. American Mathematical Society.

Cobos, F. and Kühn, T. (1990). Eigenvalues of Integral Operators with Positive Definite Kernels Satisfying Integrated Hölder Conditions over Metric Compacta. *Journal of Approximation Theory*, 63(1):39–55.

Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function. *Mathematics of control, signals and systems*, 2(4):303–314.

References II

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

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

Kim, G., Okuno, A., Fukui, K., and Shimodaira, H. (2019). Representation Learning with Weighted Inner Product
for Universal Approximation of General Similarities. *arXiv preprint arXiv:1902.10409*. submitted.

Kipf, T. N. and Welling, M. (2016). Variational Graph Auto-Encoders. *NIPS Workshop*.

Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

References III

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Minh, H. Q., Niyogi, P., and Yao, Y. (2006). Mercer's Theorem, Feature Maps, and Smoothing. In *International Conference on Computational Learning Theory*, pages 154–168. Springer.

Nickel, M. and Kiela, D. (2017). Poincaré Embeddings for Learning Hierarchical Representations. In *Advances in Neural Information Processing Systems*, pages 6338–6347.

Nickel, M. and Kiela, D. (2018). Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. In *International Conference on Machine Learning*, pages 3776–3785.

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 International Conference on Machine Learning (ICML)*, pages 3885–3894.

References IV

Okuno, A., Kim, G., and Shimodaira, H. (2019). Graph Embedding with Shifted Inner Product Similarity and Its Improved Approximation Capability. In *Proceedings of International Conference on Artificial Intelligence and Statistics*. to appear.

Okuno, A. and Shimodaira, H. (2018). On representation power of neural-network based graph embedding and beyond. In *ICML2018 workshop on Theoretical Foundations and Applications of Deep Generative Models (TADGM)*.

Ong, C. S., Mary, X., Canu, S., and Smola, A. J. (2004). Learning with non-positive kernels. In *Proceedings of the International Conference on Machine Learning*, page 81. ACM.

Prado, A., Plantevit, M., Robardet, C., and Boulicaut, J.-F. (2013). Mining graph topological patterns: Finding covariations among vertex descriptors. *IEEE Transactions on Knowledge and Data Engineering*, 25(9):2090–2104.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077.

References V

Vilnis, L. and McCallum, A. (2015). Word representations via gaussian embedding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yarotsky, D. (2018). Optimal approximation of continuous functions by very deep ReLU networks. In *Conference On Learning Theory*, pages 639–649.

Shifted Inner-Product Similarity (SIPS)

If $\nu(\cdot) = \exp(\cdot)$, using SIPS incorporates weights into IPS-based model, as

$$\underbrace{\exp\left(\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j)\rangle + u(\boldsymbol{x}_i) + u(\boldsymbol{x}_j)\right)}_{\text{SIPS-based model}} = \beta(\boldsymbol{x}_i) \cdot \beta(\boldsymbol{x}_j) \cdot \underbrace{\exp(\langle \boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{f}(\boldsymbol{x}_j)\rangle)}_{\text{IPS-based model}},$$

where $\beta(\boldsymbol{x}) := \exp(u(\boldsymbol{x}))$ is **weight function**.

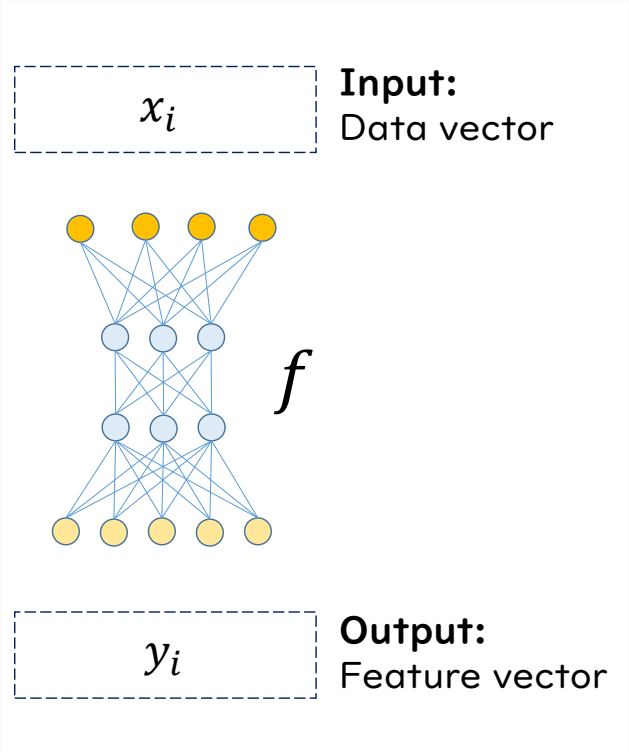
In practice, we utilize $\tilde{\boldsymbol{f}}(\boldsymbol{x}) := (\boldsymbol{f}(\boldsymbol{x})^\top, u(\boldsymbol{x}))^\top$ for subsequent statistical analyses (e.g. clustering, visualization,...).

- ☺ SIPS is capable of **approximating** any **CPD** similarities.
- ☺ SIPS-based model naturally **incorporates weight function** into IPS-based model.
- ☹ $\boldsymbol{f}, u$ in SIPS are unidentifiable (**hard to train**).

Proper regularization is required, as a future research.

Numeraical Experiments (Kim et al., 2019)

Neural network architecture is specified as follows.



- We employ 1-**hidden layer multi-layer perceptron** with $2,000$ hidden Units and ReLU activation, for each neural networks.
- For IPDS, K_+/K_- is grid-searched over $\{0.01, 0.25, 0.5, 0.75, 0.9\}$.
- For WIPS, $\boldsymbol{\lambda}$ is uniformly randomly initialized in $(0, 1/K)^K$.

For details, see Kim et al. (2019) Section 6.

Gaussian Embedding

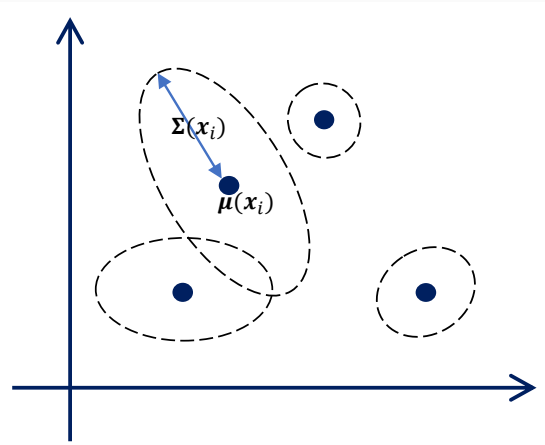
Gaussian embedding (Vilnis and McCallum, 2015) can be regarded as learning a neural network[10]

$$\boldsymbol{y}_i = \mathrm{N}_K(\underbrace{\boldsymbol{\mu}(\boldsymbol{x}_i), \boldsymbol{\Sigma}(\boldsymbol{x}_i)}_{\text{Neural network}}),$$

where N_K represents K-dim. Normal dist., so that

$$w_{ij} \approx \sigma(-D_{\mathrm{KL}}(\boldsymbol{y}_i, \boldsymbol{y}_j)).$$

☺ $\boldsymbol{\Sigma}(\boldsymbol{x}_i)$ represents the **uncertainty** of the embedding (Vilnis and McCallum, 2015).



Gaussian embedding also captures the **uncertainty** $\boldsymbol{\Sigma}(\boldsymbol{x}_i)$ of the point embedding $\boldsymbol{\mu}(\boldsymbol{x}_i)$.

D_{KL} may be replaced with symmetric Jeffrey's divergence

$$D_{\mathrm{Jeff.}}(\boldsymbol{y}, \boldsymbol{y}') := D_{\mathrm{KL}}(\boldsymbol{y}_i, \boldsymbol{y}_j) + D_{\mathrm{KL}}(\boldsymbol{y}_j, \boldsymbol{y}_i),$$

then **Jeffrey's divergence is not CPD** (Okuno et al., 2019, Supplement E.3).

[10] Vilnis and McCallum (2015) corresponds to using 1-hot vectors (as no data vectors are available), and Deep Gaussian embedding (Bojchevski and Günnemann, 2018) incorporates neural network into Gaussian embedding.