Gromov-Wasserstein Factorization Models for Graph Clustering

Hongteng Xu[1,2]
[1]Infinia ML Inc. [2]Department of ECE, Duke Univeristy
hongteng.xu@duke.edu

Abstract

We propose a new nonlinear factorization model for graphs that are with topological structures, and optionally, node attributes. This model is based on a pseudometric called Gromov-Wasserstein (GW) discrepancy, which compares graphs in a relational way. It estimates observed graphs as GW barycenters constructed by a set of atoms with different weights. By minimizing the GW discrepancy between each observed graph and its GW barycenter-based estimation, we learn the atoms and their weights associated with the observed graphs. The model achieves a novel and flexible factorization mechanism under GW discrepancy, in which both the observed graphs and the learnable atoms can be unaligned and with different sizes. We design an effective approximate algorithm for learning this Gromov-Wasserstein factorization (GWF) model, unrolling loopy computations as stacked modules and computing gradients with backpropagation. The stacked modules can be with two different architectures, which correspond to the proximal point algorithm (PPA) and Bregman alternating direction method of multipliers (BADMM), respectively. Experiments show that our model obtains encouraging results on clustering graphs.

Introduction

As an important methodology for machine learning, factorization models explore intrinsic structures of high-dimensional observations explicitly, which have been widely used in many learning tasks, $e.g.$, data clustering (Ng, Jordan, and Weiss 2002), dimensionality reduction (Candès et al. 2011), recommendation systems (Wang and Blei 2011), etc. In particular, factorization models decompose high-dimensional observations into a set of atoms under specific criteria and achieve their latent representations accordingly. For each observation, its latent representation corresponds to the coefficients associated with the atoms.[1]

However, most of the existing factorization models, such as principal component analysis (PCA) (Pearson 1901), nonnegative matrix factorization (NMF) (Sra and Dhillon 2006), and dictionary learning (Aharon, Elad, and Bruckstein 2006), are designed for vectorized samples with the same dimension. They are inapplicable to structural data, $e.g.$, graphs and point clouds. For example, in the task of graph clustering, the observed graphs are often with different numbers of nodes, and the correspondences between their nodes are often unknown, $i.e.$, the graphs are unaligned. These unaligned graphs cannot be represented as vectors directly. Although many graph embedding methods have been proposed for these years with the help of graph neural networks (Kipf and Welling 2016; Ying et al. 2018), they often require side information like node attributes and labels, which may not be available in practice. Moreover, without explicit factorization mechanisms, these methods cannot find the atoms that can reconstruct observed graphs, and thus, the graph embeddings derived by them are not so interpretable as the latent representation derived by factorization models. Therefore, it is urgent to build a flexible factorization model applicable to structural data.

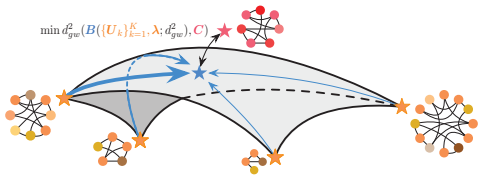


Figure 1: An illustration of our Gromov-Wasserstein factorization model. Each star indicates a graph. For each graph, the black curves show its edges, and the dots with different colors are its nodes with different attributes.

To overcome the challenges above, we propose a novel Gromov-Wasserstein factorization (GWF) model based on Gromov-Wasserstein (GW) discrepancy (Mémoli 2011; Chowdhury and Mémoli 2018) and barycenters (Peyré, Cuturi, and Solomon 2016). As illustrated in Fig. 1, for each observed graph ($i.e.$, the red star), our GWF model reconstructs it based on a set of atoms ($i.e.$, the orange stars corresponding to four graphs). The reconstruction ($i.e.$, the blue star) is the GW barycenter of the atoms that minimizes the GW discrepancy to the observed graph. The weights of the atoms ($i.e.$, the blue arrows with different widths) formu-

[1]In this paper, we borrow the terms "atoms" and "coefficients" from a kind of factorization model called dictionary learning (Aharon, Elad, and Bruckstein 2006).

late the embedding of the observed graph. Learning GW barycenters reconstructs graphs from atoms, and the GW discrepancy provides a pseudometric to measure their reconstruction errors. We design an effective approximate algorithm for learning the atoms and the graph embeddings (the weights of the atoms), unrolling loopy computations of GW discrepancy and barycenters and simplifying backpropagation based on the envelope theorem (Afriat 1971). The approximate algorithm can be implemented based on either proximal point algorithm (PPA) (Xu, Luo, and Carin 2019) or Bregman alternating direction method of multipliers (BADMM) (Wang and Banerjee 2014).

Our GWF model explicitly factorizes graphs into a set of atoms. The atoms are shared by all observations while their weights specialized for different individuals. This model has several advantages. Firstly, it is with high flexibility. The observed graphs, the atoms, and the GW barycenters can be with different sizes, and their alignment is achieved by the optimal transport corresponding to the GW discrepancy between them. Secondly, this model is compatible with existing models, which can be learned based on backpropagation and can be used as a structural regularizer in supervised learning. In the aspect of data, this model is applicable no matter whether the graphs are with node attributes or not. Thirdly, the graph embeddings derived by our model is more interpretable — they directly reflect the significance of the atoms. To our knowledge, our work makes the first attempt to establish an explicit factorization mechanism for graphs, which extends traditional factorization models under GW discrepancy. Experimental results show that the GWF model achieves encouraging results in the task of graph clustering.

Proposed Model

In this work, we represent a graph as its adjacency matrix $\boldsymbol{C} \in \mathbb{R}^{N\times N}$, whose elements are nonnegative. Optionally, when the graph is with D-dimensional node attributes, we represent its node attributes as a matrix $\boldsymbol{F} \in \mathbb{R}^{N\times D}$. Furthermore, for a graph we denote the empirical distribution of its nodes as $\boldsymbol{\mu} \in \Delta^{N-1}$, where $\Delta^{N-1} = \{\boldsymbol{x} = [x_n] \in \mathbb{R}^N | x_n \geq 0, \text{ and } \sum_k x_k = 1\}$ represents a $(N-1)$-simplex. Following (Peyré, Cuturi, and Solomon 2016), we assume that the empirical distribution is uniform, $i.e.$, $\boldsymbol{\mu} = \frac{1}{N}\mathbf{1}_N$.[2] Given a set of observations $\{\boldsymbol{C}_i \in [0,\infty)^{N_i\times N_i}, \boldsymbol{\mu}_i \in \Delta^{N_i-1}\}_{i=1}^I$, we aim at designing a factorization model with K atoms $\boldsymbol{U}_{1:K} = \{\boldsymbol{U}_k = [u^k_{ij}] \in [0,\infty)^{N_k\times N_k}\}_{k=1}^K$ and representing each observation $\boldsymbol{C}_i$ as an embedding vector $\boldsymbol{\lambda}_i = [\lambda_{ik}] \in \mathbb{R}^K$, such that the element λ_{ik} can be interpreted as the significance of the k-th atom for the i-th observation. Here, we assume that each $\boldsymbol{\lambda}_i$ is in a $(K-1)$-simplex as well, $i.e.$, $\boldsymbol{\lambda}_i \in \Delta^{K-1}$ for $i = 1, ..., I$. It should be noted that generally for different $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$, their nodes are unaligned and $N_i \neq N_j$.

Revisiting factorization models

Many existing factorization models assume that each vectorized data $\boldsymbol{y} \in \mathbb{R}^N$ can be represented as the weighted sum of K atoms, $i.e.$, $\boldsymbol{y} = \sum_k \lambda_k \boldsymbol{a}_k = \boldsymbol{A}\boldsymbol{\lambda}$, where $\boldsymbol{A} = [\boldsymbol{a}_k] \in \mathbb{R}^{N\times K}$ contains K atoms and $\boldsymbol{\lambda} = [\lambda_k] \in \mathbb{R}^K$ is the latent representation of $\boldsymbol{y}$. Given a set of observations $\{\boldsymbol{y}_i\}_{i=1}^I$, a straightforward method to learn a factorization model is solving the following optimization problem:

$$\min_{\{\boldsymbol{A},\boldsymbol{\lambda}_{1:I}\}\in\Omega} \sum_{i=1}^I d^p_{\text{loss}}(\boldsymbol{A}\boldsymbol{\lambda}_i, \boldsymbol{y}_i). \tag{1}$$

Here, Ω represents the constraints on $\boldsymbol{A}$ and/or $\boldsymbol{\lambda}_{1:I}$. $d_{\text{loss}}(\cdot,\cdot)$ defines the distance between $\boldsymbol{y}_i$ and its estimation $\boldsymbol{A}\boldsymbol{\lambda}_i$, which is used as the loss function, and p indicates the order of d_{loss}. This optimization problem can be specialized as many classical models: without any constraints, PCA (Pearson 1901) sets d_{loss} to Euclidean distance and $p = 2$; robust PCA (Candès et al. 2011) sets d_{loss} to l_1-norm and $p = 1$; NMF (Sra and Dhillon 2006) sets Ω to the set of nonnegative matrices; the Wasserstein dictionary learning (Rolet, Cuturi, and Peyré 2016) sets d_{loss} to Wasserstein distance and Ω the set of the nonnegative matrices with normalized columns.

Furthermore, when $\boldsymbol{\lambda} \in \Delta^{K-1}$, the linear factorization model $\boldsymbol{A}\boldsymbol{\lambda}$ corresponds to learning a barycenter of the atoms in the Euclidean space, which can be rewritten as

$$\boldsymbol{b}(\boldsymbol{A},\boldsymbol{\lambda};d_2^2) := \boldsymbol{A}\boldsymbol{\lambda} = \arg\min_{\boldsymbol{y}} \sum_{k=1}^K \lambda_k \|\boldsymbol{y} - \boldsymbol{a}_k\|_2^2. \tag{2}$$

Here, $\boldsymbol{y}$ represents the variable of the optimization problem and its optimal solution is $\boldsymbol{A}\boldsymbol{\lambda}$ ($i.e.$, an estimation of $\boldsymbol{y}_i$ in (1)). Extending the Euclidean metric to other metrics, we obtain nonlinear factorization models, $i.e.$, $\boldsymbol{b}(\boldsymbol{A},\boldsymbol{\lambda};d_{\text{b}}^q) := \min_{\boldsymbol{y}} \sum_{k=1}^K \lambda_k d_{\text{b}}^q(\boldsymbol{y},\boldsymbol{a}_k)$, where $d_{\text{b}}(\cdot,\cdot)$ is the metric used to calculate barycenters and q is its order. For example, when $\boldsymbol{a}_k \in \Delta^{N-1}$ for $k = 1, ..., K$ and d_{b} is Wasserstein metric, $\boldsymbol{b}(\boldsymbol{a}_{1:K},\boldsymbol{\lambda};d_{\text{b}}^q)$ corresponds to the Wasserstein factorization models (Schmitz et al. 2018). In summary, many existing factorization models are in the following framework:

$$\min_{\{\boldsymbol{A},\boldsymbol{\lambda}_{1:I}\}\in\Omega} \sum_{i=1}^I d^p_{\text{loss}}(\boldsymbol{b}(\boldsymbol{A},\boldsymbol{\lambda}_i;d_{\text{b}}^q),\boldsymbol{y}_i). \tag{3}$$

Gromov-Wasserstein factorization

As aforementioned, when the observed data are graphs, $i.e.$, replacing the vectors $\boldsymbol{y}_{1:I}$ and the atoms $\boldsymbol{A}$ in (3) with graphs $\boldsymbol{C}_{1:I}$ and their atoms $\boldsymbol{U}_{1:K}$, respectively, the classic metrics become inapplicable. In such a situation, we set the d_{loss} and d_{b} in (3) to GW discrepancy, denoted as d_{gw}, and achieve the proposed GWF model. The GW discrepancy is an extension of the Gromov-Wasserstein distance (Mémoli 2011) on metric measure spaces:

Definition *(Gromov-Wasserstein distance) Let (X, d_X, u_X) and (Y, d_Y, u_Y) be two metric measure spaces, where (X, d_X) is a compact metric space and u_X is a Borel probability measure on X (with (Y, d_Y, u_Y) defined in the same way). For $p \in [1,\infty)$, the p-th order Gromov-Wasserstein distance $d_{gw}(u_X, u_Y)$ is defined as*

$$\inf_{\pi\in\Pi(u_X,u_Y)} \Big(\iint_{X\times Y, X\times Y} L^p_{x,y,x',y'}\, d\pi(x,y)\, d\pi(x',y')\Big)^{\frac{1}{p}},$$

where $L_{x,y,x',y'} = |d_X(x,x') - d_Y(y,y')|$ is the loss function and $\Pi(u_X, u_Y)$ is the set of all probability measures on $X \times Y$ with u_X and u_Y as marginals.

[2] Our model is applicable for other distributions, $e.g.$, the distribution derived from node degree (Xu, Luo, and Carin 2019).

The GW discrepancy is similar to the GW distance, but it does not require the d_X and the d_Y to be strict metrics. Therefore, it defines a flexible pseudometric on structural data like graphs (Chowdhury and Mémoli 2018; Xu, Luo, and Carin 2019) and point clouds (Peyré, Cuturi, and Solomon 2016). In particular, given two graphs $\{\boldsymbol{C}_s = [c^s_{ij}] \in \mathbb{R}^{N_s \times N_s}, \boldsymbol{C}_t = [c^t_{i'j'}] \in \mathbb{R}^{N_t \times N_t}\}$ and their empirical entity distributions $\{\boldsymbol{\mu}_s \in \mathbb{R}^{N_s}, \boldsymbol{\mu}_t \in \mathbb{R}^{N_t}\}$, the order-2 GW discrepancy between them, denoted as $d_{gw}(\boldsymbol{C}_s, \boldsymbol{C}_t)$, is defined as

$$\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_s, \boldsymbol{\mu}_t)} \Big(\sum\nolimits_{i,j=1}^{N_s} \sum\nolimits_{i',j'=1}^{N_t} |c^s_{ij} - c^t_{i'j'}|^2 T_{ii'} T_{jj'} \Big)^{\frac{1}{2}}, \tag{4}$$

where $\Pi(\boldsymbol{\mu}_s, \boldsymbol{\mu}_t) = \{\boldsymbol{T} \geq \boldsymbol{0} | \boldsymbol{T}\boldsymbol{1}_{N_t} = \boldsymbol{\mu}_s, \boldsymbol{T}^\top \boldsymbol{1}_{N_s} = \boldsymbol{\mu}_t\}$. The optimal $\boldsymbol{T}$ indicates the optimal transport between the nodes of $\boldsymbol{C}_s$ and those of $\boldsymbol{C}_t$. According to (Peyré, Cuturi, and Solomon 2016), we can rewrite (4) as

$$d_{gw}(\boldsymbol{C}_s, \boldsymbol{C}_t) := \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_s, \boldsymbol{\mu}_t)} (\langle \boldsymbol{C}_{st} - 2\boldsymbol{C}_s \boldsymbol{T} \boldsymbol{C}_t^\top, \boldsymbol{T} \rangle)^{\frac{1}{2}}, \tag{5}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product of matrices, $\boldsymbol{C}_{st} = (\boldsymbol{C}_s \odot \boldsymbol{C}_s)\boldsymbol{\mu}_s \boldsymbol{1}_{N_t}^\top + \boldsymbol{1}_{N_s} \boldsymbol{\mu}_t^\top (\boldsymbol{C}_t \odot \boldsymbol{C}_t)^\top$ and $\odot$ represents the Hadamard product of matrices. The computation of the GW discrepancy corresponds to an optimal transport problem. This problem can be solved iteratively by the entropic regularization-based method (Peyré, Cuturi, and Solomon 2016) or the proximal point algorithm (Xu, Luo, and Carin 2019).

Given K graphs $\{\boldsymbol{U}_k\}_{k=1}^K$ and their weights $\boldsymbol{\lambda} = [\lambda_k] \in \Delta^{K-1}$, we can naturally define their order-2 GW barycenter based on the GW discrepancy in (5):

$$\boldsymbol{B}(\boldsymbol{U}_{1:K}, \boldsymbol{\lambda}; d^2_{gw}) := \arg\min_{\boldsymbol{B}} \sum\nolimits_k \lambda_k d^2_{gw}(\boldsymbol{B}, \boldsymbol{U}_k). \tag{6}$$

According to the definition we can find that the computation of the GW barycenter involves solving K optimal transport problems iteratively (Peyré, Cuturi, and Solomon 2016).

Plugging d_{gw} into (3) and setting $p = q = 2$, we obtain the proposed Gromov-Wasserstein factorization model:

$$\min_{\boldsymbol{U}_{1:K} \geq \boldsymbol{0}, \boldsymbol{\lambda}_{1:I} \in \Delta^{K-1}} \sum\nolimits_{i=1}^{I} d^2_{gw}(\boldsymbol{B}(\boldsymbol{U}_{1:K}, \boldsymbol{\lambda}_i; d^2_{gw}), \boldsymbol{C}_i). \tag{7}$$

In this model, each graph $\boldsymbol{C}_i$ is estimated by a GW barycenter of atoms $\boldsymbol{U}_{1:K}$. The weights associated with the atoms formulate the embedding vector $\boldsymbol{\lambda}_i$. Both the atoms and the embeddings are learned to minimize the GW discrepancy between the observed graphs and their GW barycenter-based estimations. As aforementioned, we require the elements of each atom to be nonnegative and make each embedding in a $(K-1)$-simplex.

As shown in Figure 1, this GWF model applies several atoms to construct a collection of graphs, in which each graph is a barycenter of the atoms. Each observed graph is reconstructed by the most similar graph in the collection, and the embedding vector $\boldsymbol{\lambda}_i$ corresponding to the reconstructed graph indicates the significance of different atoms. The embeddings of observed graphs can be used as features for many downstream tasks, $e.g.$, graph clustering. The GWF model is very flexible — the atoms and the observed graphs can be with different sizes. Differing from the graph embedding methods in (Henaff, Bruna, and LeCun 2015; Ying et al. 2018), the proposed GWF model does not rely on any side information like labels of graphs in the learning phase. Additionally, because of using an explicit factorization mechanism, the proposed model has better interpretability than many existing methods.

Learning Algorithm

Reformulation of the problem

Learning GWF models is challenging because (7) is a highly-nonlinear constrained optimization problem. As shown in (5, 6), the computation of the GW discrepancy and that of the GW barycenter correspond to two optimization problems, respectively, so the GWF model in (7) is a complicated composition of multiple optimization tasks that are with different variables. Facing such a difficult problem, we reformulate it and simplify its computations.

Reparametrization of target variables To reformulate (7) as an unconstrained problem, we further parametrize the parameters $\boldsymbol{U}_{1:K}$ and $\boldsymbol{\lambda}_{1:I}$ as

$$\begin{aligned} &u^k_{ij} = f(v^k_{ij}),\ \forall k = 1, ..., K,\ \forall i, j = 1, ..., N_k, \\ &\boldsymbol{\lambda}_i = g(\boldsymbol{z}_i),\ \forall i = 1, ..., I. \end{aligned} \tag{8}$$

where $\boldsymbol{V}_{1:K} = \{\boldsymbol{V}_k = [v^k_{ij}] \in \mathbb{R}^{N_k \times N_k}\}_{k=1}^K$ and $\boldsymbol{z}_{1:I} = \{\boldsymbol{z}_i = [z_{ik}] \in \mathbb{R}^K\}$ are new unconstrained parameters, $f(\cdot) = \text{ReLU}(\cdot)$ and $g(\cdot) = \text{Softmax}(\cdot)$ are two functions mapping the new parameters to the feasible domains of original problem. Note that we use $\text{ReLU}(\cdot)$ to pursue atoms with sparse edges. Accordingly, we use the $\boldsymbol{z}_i$ as the embedding of the graph $\boldsymbol{C}_i$.

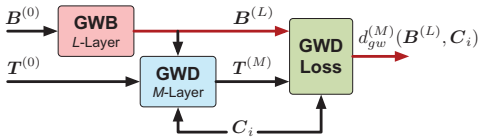
Unrolling loops For each $\boldsymbol{C}_i$, the feedforward computation of its objective function in (7) corresponds to two steps: i) solving K optimal transport problems iteratively to estimate a GW barycenter; ii) solving one more optimal transport problem to derive the GW discrepancy between the GW barycenter and $\boldsymbol{C}_i$. Each step contains loopy computations of optimal transport matrices. We unroll the loops as stacked modules shown in Figure 2(a). Specifically, we design a Gromov-Wasserstein discrepancy (GWD) module with M layers to achieve an approximation of GW discrepancy. Based on this GWD module, we further propose a GW barycenter (GWB) module with L layers to obtain an approximation of GW barycenter. Figure 2(b) illustrates one layer of the GWB module.

Based on these two modifications, we reformulate (7) as

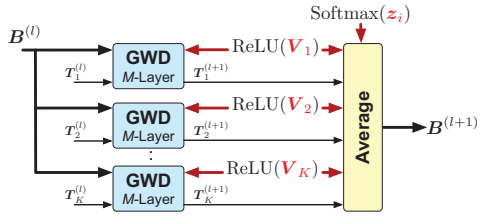
$$\min_{\boldsymbol{V}_{1:K}, \boldsymbol{z}_{1:I}} \sum\nolimits_{i=1}^{I} d^{2(M)}_{gw}(\boldsymbol{B}^{(L)}(f(\boldsymbol{V}_{1:K}), g(\boldsymbol{z}_i); d^2_{gw}), \boldsymbol{C}_i). \tag{9}$$

Implementations of the modules

The GWD module is the backbone of our algorithm. In this paper, we propose two options to implement this module. This first one is the proximal point algorithm (PPA) in (Xu et al. 2019; Xu, Luo, and Carin 2019). Given $\{\boldsymbol{C}_s, \boldsymbol{\mu}_s\}$ and



(a) Stacked computational modules



(b) One layer of the GW barycenter module

Figure 2: (a) The stacked modules for learning GWF models. (b) One layer of the GWB module. In each figure, the red arrows represent the paths used for both feedforward computation and backpropagation, while the black ones are just for feedforward computation. The GWD corresponds to Algorithms 1 and 2, the GWB corresponds to Algorithm 3, and the GWD loss corresponds to (12).

Algorithm 1 $\text{GWD}_{\text{PPA}}^{(M)}(\boldsymbol{C}_s, \boldsymbol{C}_t)$

$\boldsymbol{C}_{st} = (\boldsymbol{C}_s \odot \boldsymbol{C}_s)\boldsymbol{1}\boldsymbol{\mu}_t^\top + \boldsymbol{\mu}_s \boldsymbol{1}^\top (\boldsymbol{C}_t \odot \boldsymbol{C}_t)^\top$
$\boldsymbol{T}^{(0)} = \boldsymbol{\mu}_s \boldsymbol{\mu}_t^\top$, $\boldsymbol{a} = \boldsymbol{\mu}_s$
for $m = 0, ..., M-1$
  $\boldsymbol{\Phi} = \exp(-\frac{1}{\gamma}(\boldsymbol{C}_{st} - 2\boldsymbol{C}_s \boldsymbol{T}^{(m)} \boldsymbol{C}_t^\top)) \odot \boldsymbol{T}^{(m)}$.
  $\boldsymbol{b} = \frac{\boldsymbol{\mu}_t}{\boldsymbol{\Phi}^\top \boldsymbol{a}}$, $\boldsymbol{a} = \frac{\boldsymbol{\mu}_s}{\boldsymbol{\Phi} \boldsymbol{b}}$, and $\boldsymbol{T}^{(m+1)} = \text{diag}(\boldsymbol{a}) \boldsymbol{\Phi} \text{diag}(\boldsymbol{b})$.
return $\boldsymbol{T}^{(M)}$

$\{\boldsymbol{C}_t, \boldsymbol{\mu}_t\}$, the PPA solve (5) iteratively. In each iteration, it solves the following problem:

$$\begin{aligned}\boldsymbol{T}^{(m+1)} =& \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_s, \boldsymbol{\mu}_t)} \langle \boldsymbol{C}_{st} - 2\boldsymbol{C}_s \boldsymbol{T}^{(m)} \boldsymbol{C}_t^\top, \boldsymbol{T} \rangle \\ &+ \gamma \text{KL}(\boldsymbol{T} \| \boldsymbol{T}^{(m)}),\end{aligned}$$

where $\text{KL}(\boldsymbol{T} \| \boldsymbol{T}^{(m)})$ computes the KL-divergence between the optimal transport matrix and its previous estimation. We solve this problem approximately by one-step Sinkhorn-Knopp update (Sinkhorn and Knopp 1967; Xu, Luo, and Carin 2019). After M iterations, we derive the M-step approximation of the optimal transport matrix. Accordingly, we show the scheme of the PPA-based GWD module in Algorithm 1. This method is based on the work in (Xu et al. 2019), which replaces the entropy regularizer in (Peyré, Cuturi, and Solomon 2016) with a KL divergence. According to (Xu et al. 2019; Xu, Luo, and Carin 2019), the PPA outperforms the entropic GW method (Peyré, Cuturi, and Solomon 2016) on both stability and convergence.

Besides the PPA-based GWD module, we propose a different kind of GWD module based on the Bregman alternating direction method of multipliers (BADMM) (Wang and Banerjee 2014). Specifically, introducing an auxiliary variable $\boldsymbol{S}$, we rewrite (5) as

$$\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_s, \cdot), \boldsymbol{S} \in \Pi(\cdot, \boldsymbol{\mu}_t), \boldsymbol{T} = \boldsymbol{S}} \langle \boldsymbol{C}_{st} - 2\boldsymbol{C}_s \boldsymbol{S} \boldsymbol{C}_t^\top, \boldsymbol{T} \rangle, \quad (10)$$

where $\Pi(\boldsymbol{\mu}_s, \cdot) = \{\boldsymbol{T} \geq \boldsymbol{0} \mid \boldsymbol{T}\boldsymbol{1}_M = \boldsymbol{\mu}_s\}$ and $\Pi(\cdot, \boldsymbol{\mu}_t) = \{\boldsymbol{T} \geq \boldsymbol{0} \mid \boldsymbol{T}^\top \boldsymbol{1}_N = \boldsymbol{\mu}_t\}$. We further introduce a dual variable $\boldsymbol{Z}$ and solve (10) by the following three steps (Wang and Banerjee 2014; Ye et al. 2017).

$$\begin{aligned}\boldsymbol{T}^{(m+1)} =& \arg\min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_s, \cdot)} \langle \boldsymbol{C}_{st} - 2\boldsymbol{C}_s \boldsymbol{S}^{(m)} \boldsymbol{C}_t^\top, \boldsymbol{T} \rangle \\ &+ \langle \boldsymbol{Z}^{(m)}, \boldsymbol{T} - \boldsymbol{S}^{(m)} \rangle + \gamma \text{KL}(\boldsymbol{T} \| \boldsymbol{S}^{(m)}), \\ \boldsymbol{S}^{(m+1)} =& \arg\min_{\boldsymbol{S} \in \Pi(\cdot, \boldsymbol{\mu}_t)} \langle -2\boldsymbol{C}_s^\top \boldsymbol{T}^{(m+1)} \boldsymbol{C}_t, \boldsymbol{S} \rangle \\ &+ \langle \boldsymbol{Z}^{(m)}, \boldsymbol{T}^{(m+1)} - \boldsymbol{S} \rangle + \gamma \text{KL}(\boldsymbol{S} \| \boldsymbol{T}^{(m+1)}), \\ \boldsymbol{Z}^{(m+1)} =& \boldsymbol{Z}^{(m)} + \gamma(\boldsymbol{T}^{(m+1)} - \boldsymbol{S}^{(m+1)}).\end{aligned} \quad (11)$$

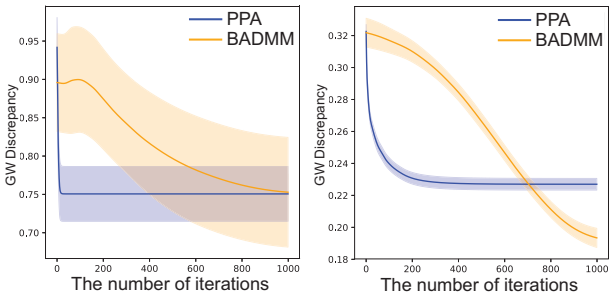
Accordingly, Algorithm 2 gives the scheme of the BADMM-based GWD module.

The BADMM algorithm is originally designed for computing Wasserstein distance (Wang and Banerjee 2014; Ye et al. 2017). To our knowledge, our work is the first attempt to apply BADMM to compute the GW discrepancy. We test these two GWD modules on synthetic graphs and find that they are suitable for different scenarios. In particular, we synthesize 100 pairs of undirected graphs and 100 pairs of directed graphs, respectively. Each graph is with 100 nodes. The directed graphs are generated based on Barabási-Albert (BA) model (Barabási and others 2016). For each directed graph, we add its adjacency matrix to its transpose and derive an undirected graph accordingly. For each pair of graphs, we apply our GWD modules to compute the GW discrepancy between the two graphs, and then, we calculate the mean and the standard deviation of the GW discrepancy in each step. The comparisons for the PPA-based module and the BADMM-based module are shown in Figures 3(a) and 3(b). The PPA-based module requires fewer steps than the BADMM-based module to converge to a stable optimal transport matrix for both undirected and directed graphs. However, the BADMM-based module can achieve smaller GW discrepancy when applying to directed graphs. In other words, we need to select different GWD modules according to the structures of observed graphs and the practical constraints on computational complexity.

The GWB module is implemented based on the GWD modules. As shown in Algorithm 3, given an initial GW

Algorithm 2 $\text{GWD}_{\text{BADMM}}^{(M)}(\boldsymbol{C}_s, \boldsymbol{C}_t)$

$\boldsymbol{C}_{st} = (\boldsymbol{C}_s \odot \boldsymbol{C}_s)\boldsymbol{1}\boldsymbol{\mu}_t^\top + \boldsymbol{\mu}_s \boldsymbol{1}^\top (\boldsymbol{C}_t \odot \boldsymbol{C}_t)^\top$
$\boldsymbol{T}^{(0)} = \boldsymbol{\mu}_s \boldsymbol{\mu}_t^\top$, $\boldsymbol{Z} = \boldsymbol{0}$.
for $m = 0, ..., M-1$
  $\boldsymbol{\Phi}_1 = \exp(\frac{1}{\gamma}(2\boldsymbol{C}_s^\top \boldsymbol{T}^{(m)} \boldsymbol{C}_t + \boldsymbol{Z})) \odot \boldsymbol{T}^{(m)}$.
  $\boldsymbol{S} = \boldsymbol{\Phi}_1 \text{diag}(\frac{\boldsymbol{\mu}_t}{\boldsymbol{\Phi}_1^\top \boldsymbol{1}})$.
  $\boldsymbol{\Phi}_2 = \exp(-\frac{1}{\gamma}(\boldsymbol{C}_{st} - 2\boldsymbol{C}_s \boldsymbol{S} \boldsymbol{C}_t^\top + \boldsymbol{Z})) \odot \boldsymbol{S}$.
  $\boldsymbol{T}^{(m+1)} = \text{diag}(\frac{\boldsymbol{\mu}_s}{\boldsymbol{\Phi}_2 \boldsymbol{1}}) \boldsymbol{\Phi}_2$.
  $\boldsymbol{Z} = \boldsymbol{Z} + \gamma(\boldsymbol{T}^{(m+1)} - \boldsymbol{S})$.
return $\boldsymbol{T}^{(M)}$



(a) Undirected graphs (b) Directed graphs

Figure 3: Comparisons for PPA and BADMM.

Algorithm 3 $\text{GWB}^{(L)}(\boldsymbol{U}_{1:K}, \boldsymbol{\lambda}, \boldsymbol{B}^{(0)}, \boldsymbol{\mu}_b)$

1: **for** $l = 0, ..., L-1$
2: **for** $k = 1, ..., K$
3: $\boldsymbol{T}_k^{(l+1)} = \text{GWD}^{(M)}(\boldsymbol{B}^{(l)}, \boldsymbol{U}_k)$.
4: $\boldsymbol{B}^{(l+1)} = \frac{1}{\boldsymbol{\mu}_b \boldsymbol{\mu}_b^\top} \sum_{k=1}^K \lambda_k \boldsymbol{T}_k^{(l+1)} \boldsymbol{U}_k (\boldsymbol{T}_k^{(l+1)})^\top$.
5: **return** $\boldsymbol{B}^{(L)}, \boldsymbol{T}_{1:K}^{(L)}$

barycenter $\boldsymbol{B}^{(0)}$ and its empirical distribution $\boldsymbol{\mu}_b$, we obtain a L-step approximation of GW barycenter and the optimal transport matrices between the barycenter and the atoms. In particular, we calculate the GW barycenter via alternating optimization: 1) update the optimal transports between the atoms and the barycenter; and 2) update the barycenter accordingly. Figure 2b illustrates one step of the GWB module ($i.e.$, the lines 2-4 in Algorithm 3), where $\boldsymbol{T}_k^{(l)}$ is the estimated optimal transport in the l-th step of the GWB module. We update it by calling a GWD-module with M inner iterations, and the output is denoted as $\boldsymbol{T}_k^{(l+1)}$.

Envelope theorem-based backpropagation

Given the modules proposed above, we can compute $d_{gw}^{2(M)}(\boldsymbol{B}^{(L)}(f(\boldsymbol{V}_{1:K}), g(\boldsymbol{z}_i); d_{gw}^2), \boldsymbol{C}_i)$ easily and apply backpropagation to update variables $\boldsymbol{V}_{1:K}$ and $\boldsymbol{z}_{1:L}$. Here, we take advantage of the envelope theorem (Afriat 1971) to simplify the computation of backpropagation. In particular, we compute each optimal transport matrix, $i.e.$, the $\boldsymbol{T}_{1:K}^{(L)}$ used to calculate the GW barycenter and the $\boldsymbol{T}^{(M)}$ between the GW barycenter and the observed graph, based on current parameters. Accordingly, the objective function becomes

$$\begin{aligned} &d_{gw}^{2(M)}(\boldsymbol{B}^{(L)}(f(\boldsymbol{V}_{1:K}), g(\boldsymbol{z}_i); d_{gw}^2), \boldsymbol{C}_i) \\ &= \langle \boldsymbol{C}_{st} - 2\boldsymbol{B}^{(L)} \boldsymbol{T}^{(M)} \boldsymbol{C}_i^\top, \boldsymbol{T}^{(M)} \rangle \end{aligned} \tag{12}$$

where

$$\begin{aligned} \boldsymbol{C}_{st} &= (\boldsymbol{B}^{(L)} \odot \boldsymbol{B}^{(L)}) \boldsymbol{\mu}_b \mathbf{1}_{N_i}^\top + \mathbf{1}_{N_i} \boldsymbol{\mu}_i^\top (\boldsymbol{C}_i \odot \boldsymbol{C}_i)^\top, \\ \boldsymbol{B}^{(L)} &= \frac{1}{\boldsymbol{\mu}_b \boldsymbol{\mu}_b^\top} \sum_{k=1}^K g_k(\boldsymbol{z}_i) \boldsymbol{T}_k^{(L)} f(\boldsymbol{V}_k) (\boldsymbol{T}_k^{(L)})^\top. \end{aligned} \tag{13}$$

Based on the envelope theorem above, we calculate the gradient of (12) with fixed optimal transport matrices. The gradients for the optimal transport matrices can be ignored when applying backpropagation, which improves the efficiency of our algorithm significantly. In summary, we learn our GWF model by Algorithm 4.

Algorithm 4 Learning GWF models

Require: A dataset $\mathcal{D} = \{\boldsymbol{C}_i, \boldsymbol{\mu}_i\}_{i=1}^I$.
1: Initialize atoms $\boldsymbol{V}_{1:K}$ and $\boldsymbol{z}_{1:I}$ randomly.
2: Set the node distributions of the atoms as uniform distributions.
3: For epoch $= 1, 2, ...$
4: For $i = 1, ..., I$
5: Initialize $\boldsymbol{B}^{(0)} = \boldsymbol{C}_i$, $\boldsymbol{\mu}_b = \boldsymbol{\mu}_i$.
6: $\boldsymbol{B}^{(L)}, \boldsymbol{T}_{1:K}^{(L)} = \text{GWB}^{(L)}(f(\boldsymbol{V}_{1:K}), g(\boldsymbol{z}_i), \boldsymbol{B}^{(0)}, \boldsymbol{\mu}_b)$.
7: $\boldsymbol{T}^{(M)} = \text{GWD}^{(M)}(\boldsymbol{B}^{(L)}, \boldsymbol{C}_i)$.
8: Get $d_{gw}^{2(M)}(\boldsymbol{B}^{(L)}(f(\boldsymbol{V}_{1:K}), g(\boldsymbol{z}_i); d_{gw}^2), \boldsymbol{C}_i)$ via (12).
9: Update $\boldsymbol{V}_{1:K}$ and $\boldsymbol{z}_i$ via backpropagation.
10: **return** $\boldsymbol{V}_{1:K}$ and $\boldsymbol{z}_{1:I}$.

Extensions

Given more side information, we can extend our GWF models to more complicated scenarios.

Learning with labels When some graphs are labeled, we can achieve semi-supervised learning of our GWF model by adding a label-related loss:

$$\begin{aligned} \min_{\boldsymbol{V}_{1:K}, \boldsymbol{z}_{1:I}, \phi} &\sum_{i=1}^I d_{gw}^{2(M)}(\boldsymbol{B}^{(L)}(f(\boldsymbol{V}_{1:K}), g(\boldsymbol{z}_i); d_{gw}^2), \boldsymbol{C}_i) \\ &+ \beta \sum_{i \in \mathcal{D}_l} \text{loss}(\phi(\boldsymbol{z}_i), l_i). \end{aligned} \tag{14}$$

Here, $\text{loss}(\phi(\boldsymbol{z}_i), l_i)$ represents the label-related loss, where l_i is the label of the i-th graph and $\phi(\cdot)$ is a learnable function mapping the embedding $\boldsymbol{z}_i$ to the label space. A typical choice of the loss is $\text{CrossEntropy}(\text{MLP}(\boldsymbol{z}_i)), l_i)$.

Learning with node attributes Sometimes the nodes of each graph are associated with vectorized features. In such a situation, the observations are $\{\boldsymbol{C}_i, \boldsymbol{\mu}_i, \boldsymbol{F}_i\}_{i=1}^I$, where $\boldsymbol{F}_i \in \mathbb{R}^{N_i \times D}$ represents the features of nodes. Accordingly, the atoms of our GWF model becomes $\{\boldsymbol{U}_{1:K}, \boldsymbol{H}_{1:K}\}$, where $\boldsymbol{U}_k \in [0, \infty)^{N_k \times N_k}$ is the adjacency matrix of the k-th atom and $\boldsymbol{H}_k \in \mathbb{R}^{N_k \times D}$ represents the features of its nodes. To learn this GWF model, we can replace the GW discrepancy with the fused Gromov-Wasserstein (FGW) discrepancy in (Vayer et al. 2019a): for two graphs $\{\boldsymbol{C}_s, \boldsymbol{\mu}_s, \boldsymbol{F}_s\}$ and $\{\boldsymbol{C}_t, \boldsymbol{\mu}_t, \boldsymbol{F}_t\}$, their FGW discrepancy is

$$\begin{aligned} &d_{fgw}(\{\boldsymbol{C}_s, \boldsymbol{F}_s\}, \{\boldsymbol{C}_t, \boldsymbol{F}_t\}) \\ &= \min_{\boldsymbol{T} \in \Pi(\boldsymbol{\mu}_s, \boldsymbol{\mu}_t)} (\langle \boldsymbol{C}_{st} - 2\boldsymbol{C}_s \boldsymbol{T} \boldsymbol{C}_t^T + \boldsymbol{D}, \boldsymbol{T} \rangle)^{\frac{1}{2}}, \end{aligned} \tag{15}$$

where $\boldsymbol{D} = (\boldsymbol{F}_s \odot \boldsymbol{F}_s) \mathbf{1}_D \mathbf{1}_{N_t}^\top + \mathbf{1}_{N_s} \mathbf{1}_D^\top (\boldsymbol{F}_t \odot \boldsymbol{F}_t)^\top - 2\boldsymbol{F}_s \boldsymbol{F}_t^\top$ is the Euclidean distance matrix computed based on features. We can approximate the FGW discrepancy by our GWD modules as well — just replace the line 1 in Algorithm 1 (or Algorithm 2) with $\boldsymbol{C}_{st} = (\boldsymbol{C}_s \odot \boldsymbol{C}_s) \mathbf{1} \boldsymbol{\mu}_t^\top + \boldsymbol{\mu}_s \mathbf{1}^\top (\boldsymbol{C}_t \odot$

$\boldsymbol{C}_t)^\top + \boldsymbol{D}$. Similar to (12), the loss function becomes

$$\begin{aligned} & d_{fgw}^{2(M)}(\{\boldsymbol{B}^{(L)}, \boldsymbol{F}_b^{(L)}\}, \{\boldsymbol{C}_i, \boldsymbol{F}_i\}) \\ & = \langle \boldsymbol{C}_{st} - 2\boldsymbol{B}^{(L)}\boldsymbol{T}^{(M)}\boldsymbol{C}_i^\top + \boldsymbol{D}^{(L)},\ \boldsymbol{T}^{(M)} \rangle, \end{aligned} \tag{16}$$

where the new terms are

$$\begin{aligned} \boldsymbol{D}^{(L)} =& (\boldsymbol{F}_b^{(L)} \odot \boldsymbol{F}_b^{(L)})\mathbf{1}_D\mathbf{1}_{N_i}^\top + \mathbf{1}_{N_i}\mathbf{1}_D^\top(\boldsymbol{F}_i \odot \boldsymbol{F}_i)^\top \\ & - 2\boldsymbol{F}_b^{(L)}\boldsymbol{F}_i^\top, \\ \boldsymbol{F}_b^{(L)} =& \frac{1}{\boldsymbol{\mu}_b\mathbf{1}_D^\top}\sum\nolimits_{k=1}^{K} g_k(\boldsymbol{z}_i)\boldsymbol{T}_k^{(L)}\boldsymbol{H}_k. \end{aligned} \tag{17}$$

Related Work

GW discrepancy and its applications

As a pseudometric of structural data like graphs, GW discrepancy (Chowdhury and Mémoli 2018) has been applied to many problems, $e.g.$, registering 3D point clouds (Mémoli 2011), aligning protein networks of different species (Xu, Luo, and Carin 2019), and matching vocabulary sets of different languages (Alvarez-Melis and Jaakkola 2018). For the graphs with node attributes, the work in (Vayer et al. 2019a) proposes FGW discrepancy, which combines the GW discrepancy between graph structures with the Wasserstein discrepancy (Villani 2008) between node attributes. GW barycenters are proposed in (Peyré, Cuturi, and Solomon 2016), achieving the interpolation of multiple graphs. Recently, GW discrepancy is applied as objective functions when learning machine learning models. The work in (Bunne et al. 2019) trains coupled generative models in incomparable spaces by minimizing the GW discrepancy between their samples. The work in (Xu et al. 2019) learns node embeddings for unaligned pairwise graphs based on their GW discrepancy. Most of the existing works calculate GW discrepancy by Sinkhorn iterations(Sinkhorn and Knopp 1967), whose complexity per iteration is $\mathcal{O}(N^3)$ for the graphs with N nodes. The high computational complexity limits the applications of GW discrepancy. These years, many variants of GW discrepancy have been proposed, $e.g.$, the recursive GW discrepancy (Xu, Luo, and Carin 2019), and the sliced GW discrepancy (Vayer et al. 2019b). Although these works have achieved encouraging results in many tasks, none of them consider building Gromov-Wasserstein factorization models as we do.

Graph clustering methods

Graph clustering is significant for many practical applications, $e.g.$, molecules modeling (Borgwardt et al. 2005) and social network analysis (Yanardag and Vishwanathan 2015). Different from graph partitioning, which finds clusters of nodes in a graph, graph clustering aims at finding clusters for different graphs. The key to this problem is embedding unaligned graphs. Many methods have been proposed to attack this problem, and most of them can be categorized into kernel-based methods, $e.g.$, the Weisfeiler-Lehman kernel in (Vishwanathan et al. 2010). In principle, these methods iteratively aggregate node features according to the topology of the graphs. Recently, such a strategy becomes learnable with the help of graph convolutional networks (GCNs) (Kipf and Welling 2016). Many GCN-based methods have been proposed to embed graphs, $e.g.$, the large-scale embedding method in (Nie, Zhu, and Li 2017), and the hierarchical embedding method in (Ying et al. 2018). However, these methods rely on the labels and the attributes of nodes, which are often inapplicable for unsupervised learning. Moreover, the embeddings achieved by them are often with low interpretability because of their deep and highly-nonlinear processes. Besides the methods above, the GW discrepancy-based methods in (Peyré, Cuturi, and Solomon 2016; Vayer et al. 2019a) provide another strategy — learning a distance matrix for graphs based on their pairwise (fused) GW discrepancy and applying spectral clustering accordingly. This strategy is feasible even if the side information of nodes is not available, but its computational complexity is very high. Compared with the methods above, our GWF model provides another strategy to embed graphs in a scalable and theoretically-supportive way and make the embeddings interpretable in the framework of factorization models.

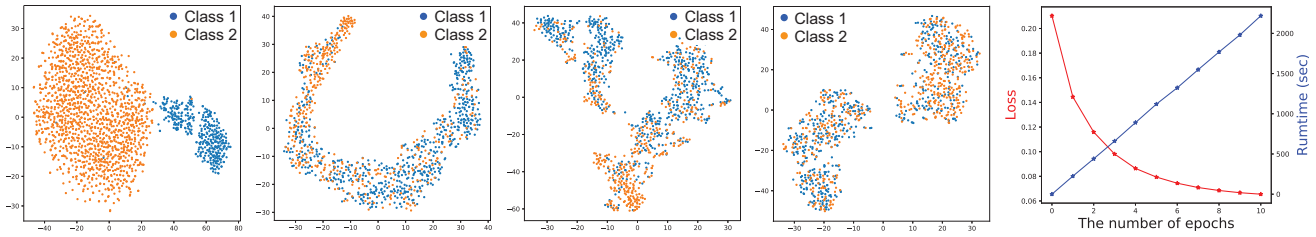
Experiments

To demonstrate the usefulness of our GWF model, we test it on four graph datasets and compare it with state-of-the-art methods on graph clustering. When implementing our GWF model, we set its hyperparameters as follows: the number of atoms is $K = 30$; the number of layers in the GWD module is $M = 50$; the weight of regularizer γ is 0.01 for the PPA-based GWD module and 1 for the BADMM-based GWD module; the number of layers in the GWB module is $L = 2$. For the graph data without node attributes, the parameters of our GWF model involve $\boldsymbol{V}_{1:K}$ and $\boldsymbol{z}_{1:I}$. For the graph data with node attributes, we will further learn node embeddings of the atoms, $i.e.$, $\boldsymbol{H}_{1:K}$. We use Adam (Kingma and Ba 2014) to train our GWF model. The learning rate of our algorithm is 0.05, and the number of epochs is 10. To accelerate the convergence of our algorithm, we apply a warm-start strategy: we randomly select K observed graphs as initial atoms. For each atom, the number of its nodes is equal to that of the selected graph. The code is at https://github.com/HongtengXu/Relational-Factorization-Model.

Graph clustering

We consider four commonly-used graph datasets (Kersting et al. 2016) in our experiments: the AIDS (Riesen and Bunke 2008), the PROTEIN dataset and its full version PROTEIN-F (Borgwardt et al. 2005), and the IMDB-B (Yanardag and Vishwanathan 2015). For each dataset, their graphs are categorized into two classes. The graphs in AIDS, PROTEIN, and PROTEIN-F represent molecules or proteins, which are with node attributes. The dimensions of the node attributes are 4 for AIDS, 1 for PROTEIN, and 29 for PROTEIN-F. The graphs in IMDB-B represent user interactions in social networks, which merely contain topological information. We show the details of the datasets in Table 1.

We apply various methods to cluster the graphs in each dataset and compare their performance in the aspect of clustering accuracy. For each dataset, we train a GWF model and applying K-means to learned embeddings $\boldsymbol{z}_{1:I}$. Besides



(a) AIDS (b) PROTEIN (c) PROTEIN-F (d) IMDB-B (e) Loss v.s. Epochs

Figure 4: (a-d) Visualizations of $\boldsymbol{z}_{1:I}$ based on t-SNE. (e) The convergence and the runtime of our method on IMDB-B.

our GWF model, we consider two state-of-the-art methods as our baselines. The first is the fused Gromov-Wasserstein kernel method (FGWK) in (Vayer et al. 2019a). Given N graphs, the FGWK method computes their pairwise fused GW discrepancy and construct a $N \times N$ kernel matrix $\exp(-d_{fgw}/\beta)$.[3] Applying spectral clustering to the kernel matrix, we can cluster observed graphs into two clusters. When the node attributes are unavailable for the observed graphs, this method computes pairwise GW discrepancy instead to construct the kernel matrix. This method has been proven to be superior to traditional kernel-based methods on many datasets, including the PROTEIN and the IMDB-B datasets used in this work. The second competitor of our method is the K-means of graphs based on GW barycenter (Peyré, Cuturi, and Solomon 2016) (GWB-KM). This method applies K-means to find centers of clusters iteratively. The centers are initialized randomly as two observed graphs. In each iteration, we first categorize the graphs into different clusters according to their GW discrepancy to the centers, and then we recalculate each center as the GW barycenter of the graphs in the corresponding cluster.

Table 1 shows that our GWF model outperforms its competitors consistently, which demonstrates its superiority in the task of graph clustering.[4] We implement our GWF model based on PPA and BADMM, respectively. The performance of the PPA-based model is better than that of the BADMM-based model in general because i) the graphs in the four datasets are undirected; ii) the BADMM-based model generally requires more steps to converge in the training phase, and 50 steps may be too few for it.[5] Additionally, given I graphs, FGWK calculates $\mathcal{O}(I^2)$ GW discrepancy, while both GWB-KM and our GWF model only need to calculate $\mathcal{O}(LKI)$ GW discrepancy. Because we can set $LK \ll I$, our model is more suitable for large-scale clustering problems. Figure 4 further visualizes the embeddings $\boldsymbol{z}_{1:I}$ for the four datasets based on t-SNE (Maaten and Hinton 2008). The visualization results further verify the effectiveness of our GWF model — for each dataset, the embeddings derived by our GWF model indeed reflect the clustering structure of the graphs. We show the convergence and the runtime of our learning method in Figure 4e.

Table 1: Comparisons on clustering accuracy (%)

Dataset	# GWD	AIDS	PROTEIN	PROTEIN-F	IMDB-B
# graphs		2000	1113	1113	1000
Ave. #nodes		15.69	39.06	39.06	19.77
Ave. #edges		16.20	72.82	72.82	96.53
FGWK	$\mathcal{O}(I^2)$	91.0±0.7	66.4±0.8	66.0±0.9	56.7±1.5
GWB-KM	$\mathcal{O}(LKI)$	95.2±0.9	64.7±1.1	62.9±1.3	53.5±2.3
$\text{GWF}_{\text{BADMM}}$	$\mathcal{O}(LKI)$	97.6±0.8	69.2±1.0	68.1±1.1	55.9±1.8
GWF_{PPA}	$\mathcal{O}(LKI)$	**99.5**±0.4	**70.7**±0.7	**69.3**±0.8	**60.2**±1.6

Table 2: Comparisons on classification accuracy (%)

Method	PROTEIN	Method	IMDB-B
HOPPERK	71.6±3.7	GCK	56.9±4.0
PROPAK	60.3±5.1	SPK	56.2±3.1
FGWK	75.1±2.9	FGWK	64.2±3.3
$\text{GWF}_{\text{BADMM}}$	71.4±3.6	$\text{GWF}_{\text{BADMM}}$	62.4±3.8
GWF_{PPA}	73.7±2.0	GWF_{PPA}	63.9±2.7

Besides graph clustering, we also consider graph classification given the labels of graphs. In such a situation, we apply the learning strategy in (14) to learn our GWF model. The baselines include the kernel-based graph classification methods like the shortest path kernel (SPK) (Borgwardt et al. 2005), the HOPPER kernel (HOPPERK) (Feragen et al. 2013), the propagation kernel (PROPAK) (Neumann et al. 2016), the graphlet count kernel (GCK) (Shervashidze et al. 2009), and the FGWK mentioned above. Each of these methods derives a kernel matrix and train a classifier based on kernel SVM. We test different methods based on 10-fold cross-validation. Table 2 shows the classification accuracy achieved by different methods on two datasets. Compared with the state-of-the-art method FGWK, our GWF model achieves comparable performance in the task of graph classification, and the fluctuations of our results are smaller than those of FGWK's results.

[3] In (Vayer et al. 2019a), the authors claim that $\exp(-d_{fgw}/\beta)$ is a noisy observation of a true positive semidefinite kernel. Our implementation confirms its performance, but whether it can always be positive semidefinite may be questionable.

[4] For the data with two clusters, given the ground truth labels $\boldsymbol{y} \in \{0,1\}^N$ and the estimated labels $\hat{\boldsymbol{y}} \in \{0,1\}^N$, we calculate the clustering accuracy via $1 - \frac{1}{N}\min(\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_1, \|\boldsymbol{y} - \boldsymbol{1} + \hat{\boldsymbol{y}}\|_1)$.

[5] When applying BADMM with $M = 300$, its performance becomes close to that of PPA while the runtime is much longer. Unless the graphs are directed and the dataset is small, we prefer using the PPA-based method.

Conclusion and Future Work

In this paper, we propose a novel Gromov-Wassserstein factorization model. It is a pioneering work achieving an explicit factorization mechanism for graph clustering. We design an efficient learning algorithm for learning this model with the help of the envelope theorem. Experiments demonstrate that our model outperforms many existing methods in the tasks of graph clustering. In the future, we plan to reduce the computational complexity of our learning algorithm further and consider its applications to large-scale graphs.

Acknowledgements This research was supported in part by DARPA, DOE, NIH, ONR, NSF, and Inifina ML.

References

Afriat, S. 1971. Theory of maxima and the method of lagrange. *SIAM Journal on Applied Mathematics* 20(3):343–357.

Aharon, M.; Elad, M.; and Bruckstein, A. 2006. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing* 54(11):4311–4322.

Alvarez-Melis, D., and Jaakkola, T. S. 2018. Gromov-Wasserstein alignment of word embedding spaces. In *EMNLP*.

Barabási, A.-L., et al. 2016. *Network science*. Cambridge university press.

Borgwardt, K. M.; Ong, C. S.; Schönauer, S.; Vishwanathan, S.; Smola, A. J.; and Kriegel, H.-P. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21(suppl_1):i47–i56.

Bunne, C.; Alvarez-Melis, D.; Krause, A.; and Jegelka, S. 2019. Learning generative models across incomparable spaces. In *ICML*.

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis? *Journal of the ACM* 58(3):11.

Chowdhury, S., and Mémoli, F. 2018. The gromov-Wasserstein distance between networks and stable network invariants. *arXiv preprint arXiv:1808.04337*.

Feragen, A.; Kasenburg, N.; Petersen, J.; de Bruijne, M.; and Borgwardt, K. 2013. Scalable kernels for graphs with continuous attributes. In *NeurIPS*.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Kersting, K.; Kriege, N. M.; Morris, C.; Mutzel, P.; and Neumann, M. 2016. Benchmark data sets for graph kernels.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Mémoli, F. 2011. Gromov-Wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics* 11(4):417–487.

Neumann, M.; Garnett, R.; Bauckhage, C.; and Kersting, K. 2016. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning* 102(2):209–245.

Ng, A. Y.; Jordan, M. I.; and Weiss, Y. 2002. On spectral clustering: Analysis and an algorithm. In *NeurIPS*.

Nie, F.; Zhu, W.; and Li, X. 2017. Unsupervised large graph embedding. In *AAAI*.

Pearson, K. 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2(11):559–572.

Peyré, G.; Cuturi, M.; and Solomon, J. 2016. Gromov-Wasserstein averaging of kernel and distance matrices. In *ICML*.

Riesen, K., and Bunke, H. 2008. Iam graph database repository for graph based pattern recognition and machine learning. In *Joint IAPR Workshop*, 287–297.

Rolet, A.; Cuturi, M.; and Peyré, G. 2016. Fast dictionary learning with a smoothed Wasserstein loss. In *AISTATS*.

Schmitz, M. A.; Heitz, M.; Bonneel, N.; Ngole, F.; Coeurjolly, D.; Cuturi, M.; Peyré, G.; and Starck, J.-L. 2018. Wasserstein dictionary learning: Optimal transport-based unsupervised nonlinear dictionary learning. *SIAM Journal on Imaging Sciences* 11(1):643–678.

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In *AISTATS*.

Sinkhorn, R., and Knopp, P. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics* 21(2):343–348.

Sra, S., and Dhillon, I. S. 2006. Generalized nonnegative matrix approximations with bregman divergences. In *NeurIPS*.

Vayer, T.; Chapel, L.; Flamary, R.; Tavenard, R.; and Courty, N. 2019a. Optimal transport for structured data with application on graphs. In *ICML*.

Vayer, T.; Flamary, R.; Tavenard, R.; Chapel, L.; and Courty, N. 2019b. Sliced gromov-Wasserstein. *arXiv preprint arXiv:1905.10124*.

Villani, C. 2008. *Optimal transport: Old and new*, volume 338. Springer Science & Business Media.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph kernels. *Journal of Machine Learning Research* 11(Apr):1201–1242.

Wang, H., and Banerjee, A. 2014. Bregman alternating direction method of multipliers. In *NeurIPS*.

Wang, C., and Blei, D. M. 2011. Collaborative topic modeling for recommending scientific articles. In *KDD*.

Xu, H.; Luo, D.; Zha, H.; and Carin, L. 2019. Gromov-Wasserstein learning for graph matching and node embedding. In *ICML*.

Xu, H.; Luo, D.; and Carin, L. 2019. Scalable Gromov-Wasserstein learning for graph partitioning and matching. *arXiv preprint arXiv:1905.07645*.

Yanardag, P., and Vishwanathan, S. 2015. Deep graph kernels. In *KDD*.

Ye, J.; Wu, P.; Wang, J. Z.; and Li, J. 2017. Fast discrete distribution clustering using Wasserstein barycenter with sparse support. *IEEE Transactions on Signal Processing* 65(9):2317–2332.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*.