Graph Matching and Partitioning

Shen Yuan

June 10, 2021

- Introduction
- Deep Learning of Graph Matching
- Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach
- Scalable Gromov-Wasserstein Learning for Graph Partitioning and Matching
- Baselines

Introduction

Graph matching:

- Input: two graphs $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$, with $|V_1| = n$, $|V_2| = m$, $|E_1| = p$ and $|E_2| = q$
- Output: an assignment between the nodes of the two graphs, so that a predefined criterion over the corresponding nodes and edges is optimized.

Graph partitioning:

- Input: one graph $\mathcal{G} = (V, E)$
- Output: k components of the $\mathcal{G}$.

- Introduction
- Deep Learning of Graph Matching
- Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach
- Scalable Gromov-Wasserstein Learning for Graph Partitioning and Matching
- Baselines

Deep Learning of Graph Matching

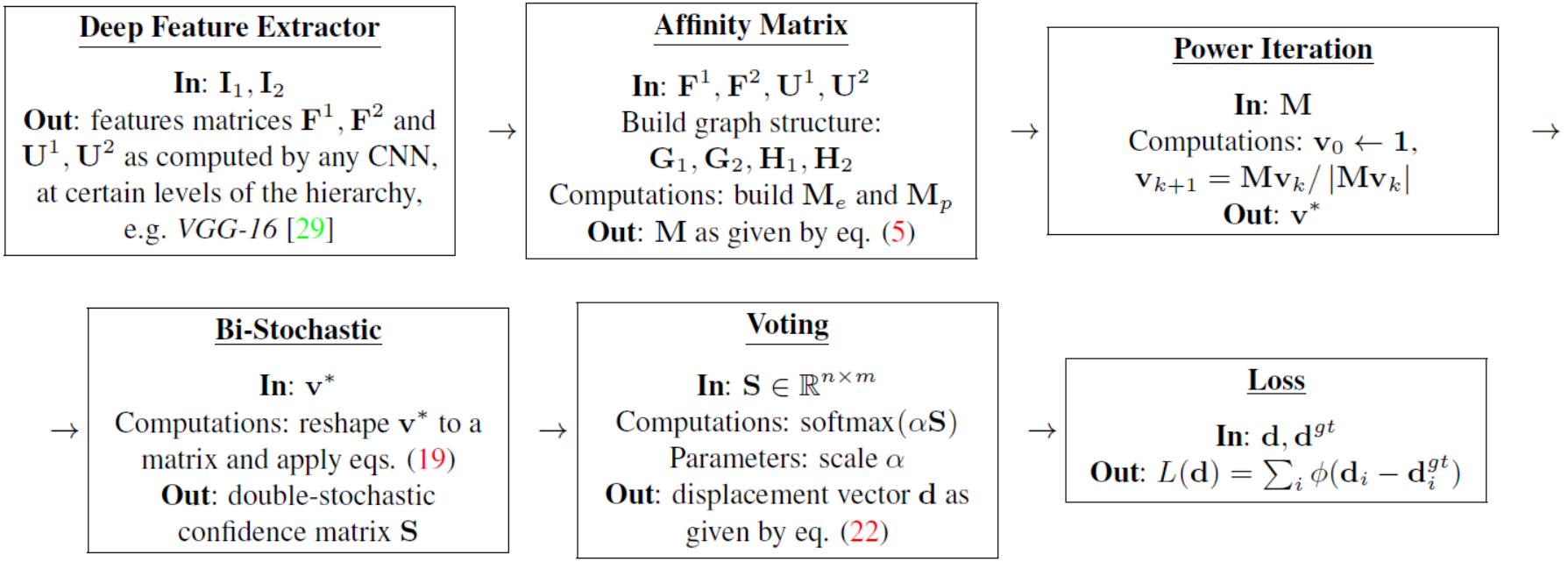

Figure 1: The pipeline.

Affinity Matrix Layer

$$\boldsymbol{A}_1 = \boldsymbol{G}_1 \boldsymbol{H}_1^\top, \ \boldsymbol{A}_2 = \boldsymbol{G}_2 \boldsymbol{H}_2^\top$$

where $\boldsymbol{A}_1 \in \{0,1\}^{n \times n}, \boldsymbol{A}_2 \in \{0,1\}^{m \times m}$ are the node-to-node adjacency matrices.

Affinity Matrix Layer

$$\boldsymbol{A}_1 = \boldsymbol{G}_1 \boldsymbol{H}_1^\top, \ \boldsymbol{A}_2 = \boldsymbol{G}_2 \boldsymbol{H}_2^\top$$

where $\boldsymbol{A}_1 \in \{0,1\}^{n \times n}, \boldsymbol{A}_2 \in \{0,1\}^{m \times m}$ are the node-to-node adjacency matrices.

$$\boldsymbol{M} = [\mathrm{vec}(\boldsymbol{M}_p)] + (\boldsymbol{G}_2 \otimes \boldsymbol{G}_1)[\mathrm{vec}(\boldsymbol{M}_e)](\boldsymbol{H}_2 \otimes \boldsymbol{H}_1)^\top$$

where the matrix $\boldsymbol{M}_p \in \mathbb{R}^{n \times m}$ measures the node-to-node similarities, and $\boldsymbol{M}_e \in \mathbb{R}^{p \times q}$ measures edge-to-edge similarities. $[x]$ represents the diagonal matrix with x on the main diagonal, and $\otimes$ is the Kronecker product.

Bi-Stochastic Layer

Given a starting matrix $\boldsymbol{S}_0 = (\boldsymbol{v}^*)_{n \times m}$,

$$\boldsymbol{S}_{k+1} = \boldsymbol{S}_k [\mathbf{1}_n^\top \boldsymbol{S}_k]^{-1}, \boldsymbol{S}_{k+2} = [\boldsymbol{S}_{k+1} \mathbf{1}_m]^{-1} \boldsymbol{S}_{k+1}$$

Voting layer

$$\boldsymbol{d}_i = \frac{\exp[\alpha \boldsymbol{S}(i, 1 \dots m)]}{\sum_i \exp[\alpha \boldsymbol{S}(i, j)]} \boldsymbol{P} - \boldsymbol{P}_i$$

where $\boldsymbol{P} \in \mathbb{R}^{m \times 2}$ is the matrix of positions.

- Introduction
- Deep Learning of Graph Matching
- Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach
- Scalable Gromov-Wasserstein Learning for Graph Partitioning and Matching
- Baselines

Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach

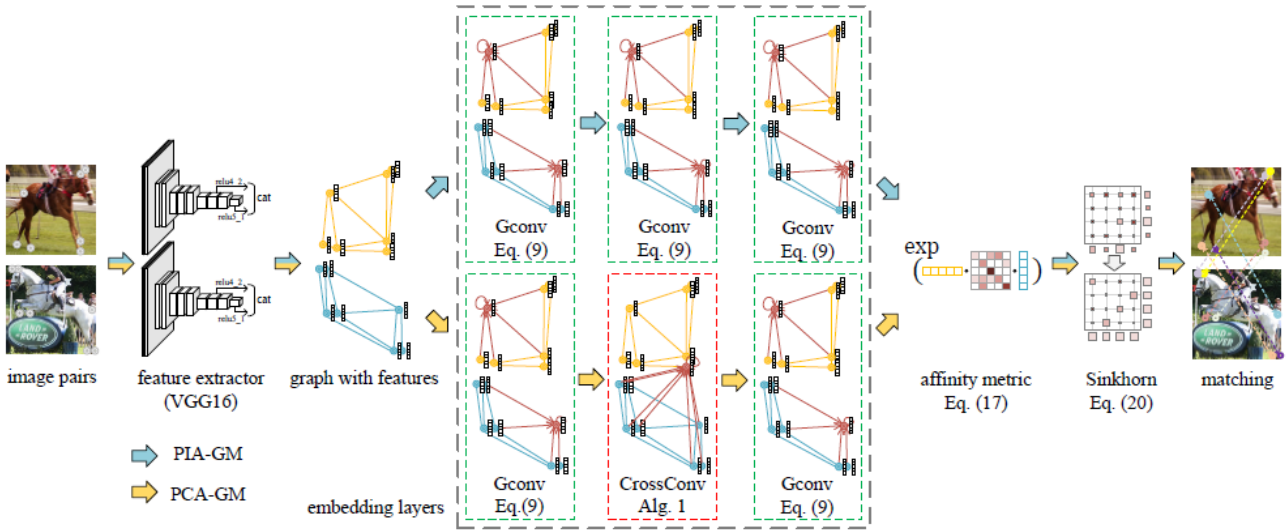

Figure 2: The pipelines.

Node Feature Extraction on Images

The extracted feature on the keypoint P_{si} of image I_s is:

$$\boldsymbol{h}_{si}^{(0)} = \text{Interp}(P_{si}, \text{CNN}(I_s))$$

where $\text{Interp}(P, X)$ bi-linearly interpolates on point P from the feature map X.

Intra-graph Node Embedding

$$
\begin{aligned}
\boldsymbol{m}_{si}^{(k)} &= \frac{1}{|(i,j) \in E_s|} \sum_{j:(i,j)\in E_s} f_{\text{msg}}(\boldsymbol{h}_{sj}^{(k-1)}) \\
\boldsymbol{n}_{si}^{(k)} &= f_{\text{node}}(\boldsymbol{h}_{si}^{(k-1)}) \\
\boldsymbol{h}_{si}^{(k)} &= f_{\text{update}}(\boldsymbol{m}_{si}^{(k)}, \boldsymbol{n}_{si}^{(k)})
\end{aligned}
$$

where f_{msg} denotes the message passing function, f_{node} denotes a self-passing function and f_{update} updates the the state of nodes.

Cross-graph Node Embedding

$$\hat{\boldsymbol{M}}_{i,j} = f_{\text{aff}}(\boldsymbol{h}_{1i}^{(1)}, \boldsymbol{h}_{2j}^{(1)}),\ i \in \mathcal{V}_1, j \in \mathcal{V}_2$$
$$\hat{\boldsymbol{S}} = \text{Sinkhorn}(\hat{\boldsymbol{M}})$$

where f_{aff} represents the affinity measure,

Cross-graph Node Embedding

$$\hat{\boldsymbol{M}}_{i,j} = f_{\text{aff}}(\boldsymbol{h}_{1i}^{(1)}, \boldsymbol{h}_{2j}^{(1)}),\ i \in \mathcal{V}_1, j \in \mathcal{V}_2$$
$$\hat{\boldsymbol{S}} = \text{Sinkhorn}(\hat{\boldsymbol{M}})$$

where f_{aff} represents the affinity measure,

$$f_{\text{aff}}(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j}) = \exp\left(\frac{\boldsymbol{h}_{1i}^{\top} \boldsymbol{A} \boldsymbol{h}_{2j}}{\tau}\right)$$

Cross-graph Node Embedding

$$\hat{\boldsymbol{M}}_{i,j} = f_{\text{aff}}(\boldsymbol{h}_{1i}^{(1)}, \boldsymbol{h}_{2j}^{(1)}),\ i \in \mathcal{V}_1, j \in \mathcal{V}_2$$
$$\hat{\boldsymbol{S}} = \text{Sinkhorn}(\hat{\boldsymbol{M}})$$

where f_{aff} represents the affinity measure,

$$f_{\text{aff}}(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j}) = \exp\left(\frac{\boldsymbol{h}_{1i}^{\top} \boldsymbol{A} \boldsymbol{h}_{2j}}{\tau}\right)$$

$$\boldsymbol{M}^{(k)\prime} = \boldsymbol{M}^{(k-1)} \oslash (\boldsymbol{M}^{(k-1)} \mathbf{1}\mathbf{1}^{\top})$$
$$\boldsymbol{M}^{(k)} = \boldsymbol{M}^{(k)\prime} \oslash (\mathbf{1}\mathbf{1}^{\top} \boldsymbol{M}^{(k)\prime})$$

where $\oslash$ means element-wise division,

Cross-graph Node Embedding

$$
\begin{aligned}
\boldsymbol{m}_{1i}^{(k)} &= \sum_{j \in \mathcal{V}_2} \hat{\boldsymbol{S}}_{i,j} f_{\text{msg-cross}}(\boldsymbol{h}_{2j}^{(k-1)}) \\
\boldsymbol{n}_{1i}^{(k)} &= f_{\text{node-cross}}(\boldsymbol{h}_{1i}^{(k-1)}) \\
\boldsymbol{h}_{1i}^{(k)} &= f_{\text{update-cross}}(\boldsymbol{m}_{1i}^{(k)}, \boldsymbol{n}_{1i}^{(k)})
\end{aligned}
$$

- Introduction
- Deep Learning of Graph Matching
- Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach
- Scalable Gromov-Wasserstein Learning for Graph Partitioning and Matching
- Baselines

Scalable Gromov-Wasserstein Learning for Graph Partitioning and Matching

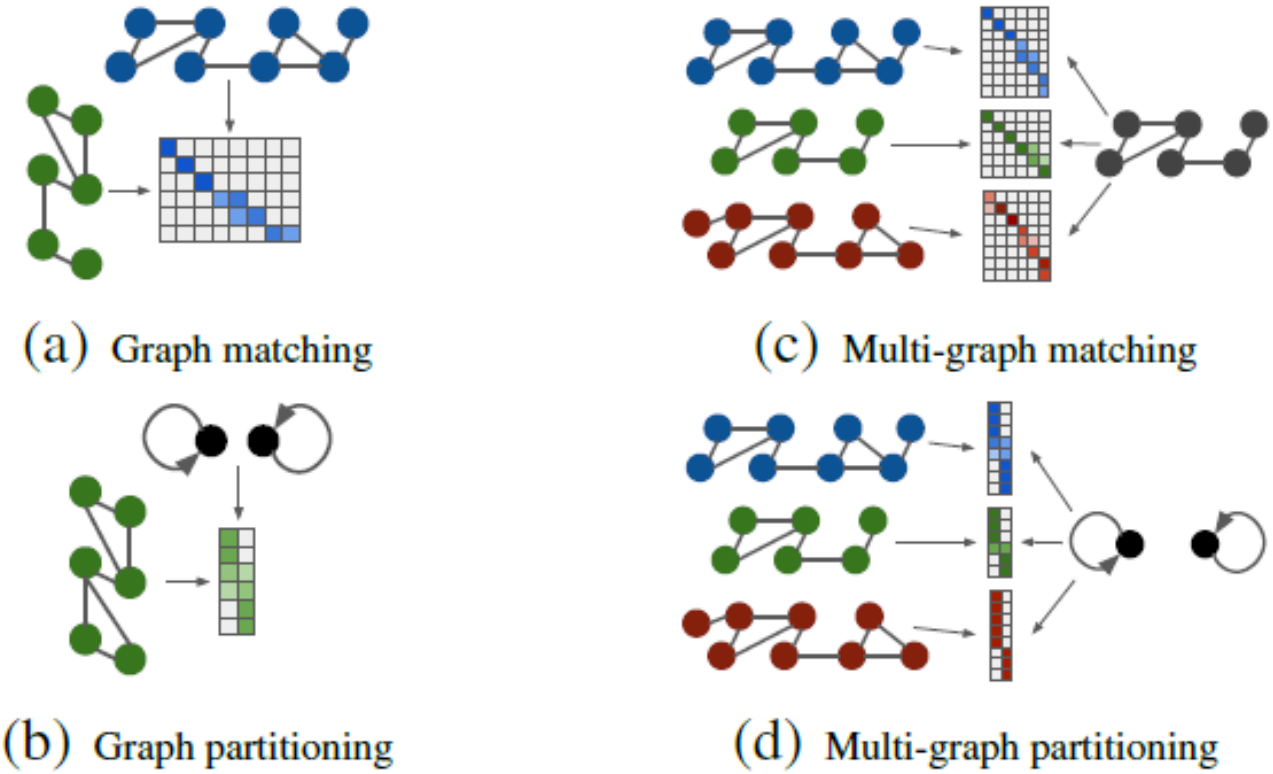

Figure 3: Applying the GWL framework into the graph partitioning and matching

Gromov-Wasserstein discrepancy between graphs

Denote a measure graph as $G(\mathcal{V}, \boldsymbol{C}, \boldsymbol{\mu})$, where $\mathcal{V} = \{v_i\}_{i=1}^{|\mathcal{V}|}$ is the set of nodes, $\boldsymbol{C} = [c_{ij}] \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$is the adjacency matrix, and $\boldsymbol{\mu} = [\mu_i] \in \sum^{|\mathcal{V}|}$ is a Borel probability measure defined on $\mathcal{V}$.

$$d_{gw}(G_s, G_t) := \min_{\boldsymbol{T} \in \prod(\boldsymbol{\mu}_s, \boldsymbol{\mu}_t)} \left(\sum_{i,j \in \mathcal{V}_s} \sum_{i',j' \in \mathcal{V}_t} |c_{ij}^s - c_{i'j'}^t|^p T_{ii'} T_{jj'} \right)^{\frac{1}{p}}$$

where $\prod(\boldsymbol{\mu}_s, \boldsymbol{\mu}_t) = \{\boldsymbol{T} \geq \mathbf{0} | \boldsymbol{T}\mathbf{1}_{|\mathcal{V}_t|} = \boldsymbol{\mu}_s, \boldsymbol{T}^\top \mathbf{1}_{|\mathcal{V}_s|} = \boldsymbol{\mu}_t\}$

Multi-graph matching and partitioning

$$G(\bar{V}, \bar{\boldsymbol{C}}, \bar{\boldsymbol{\mu}}) := \arg\min_{\bar{G}} \sum_{m=1}^{M} \omega_m d_{gw}^p(G_m, \bar{G})$$

where $\boldsymbol{\omega} = [\omega_m] \in \sum^M$ are the predefined weights, and $\bar{G} = G(\bar{\mathcal{V}}, \bar{\boldsymbol{C}} \in \mathbb{R}^{|\bar{\mathcal{V}}| \times |\bar{\mathcal{V}}|}, \bar{\boldsymbol{\mu}} \in \sum^{|\bar{\mathcal{V}}|})$

Scalable Gromov-Wasserstein Learning

$$\begin{aligned}\boldsymbol{T}^{(n+1)} &= \arg\min_{\boldsymbol{T}\in\prod(\boldsymbol{\mu}_s,\boldsymbol{\mu}_t)} \sum_{i,j\in\mathcal{V}_s}\sum_{i',j'\in\mathcal{V}_t} |c^s_{ij} - c^t_{i'j'}|^2 T^{(n)}_{ii'} T_{jj'} + \gamma \mathrm{KL}(\boldsymbol{T}\|\boldsymbol{T}^{(n)}) \\ &= \arg\min_{\boldsymbol{T}\in\prod(\boldsymbol{\mu}_s,\boldsymbol{\mu}_t)} \langle \boldsymbol{L}(\boldsymbol{C}_s,\boldsymbol{C}_t,\boldsymbol{T}^{(n)}),\boldsymbol{T}\rangle + \gamma \mathrm{KL}(\boldsymbol{T}\|\boldsymbol{T}^{(n)})\end{aligned}$$

where $\boldsymbol{L}(\boldsymbol{C}_s,\boldsymbol{C}_t,\boldsymbol{T}) = \boldsymbol{C}_s\boldsymbol{\mu}_s\mathbf{1}^\top_{|\mathcal{V}_t|} + \mathbf{1}_{|\mathcal{V}_s|}\boldsymbol{\mu}_t^\top\boldsymbol{C}_t^\top - 2\boldsymbol{C}_s\boldsymbol{T}\boldsymbol{C}_t^\top$

A recursive K-partition mechanism

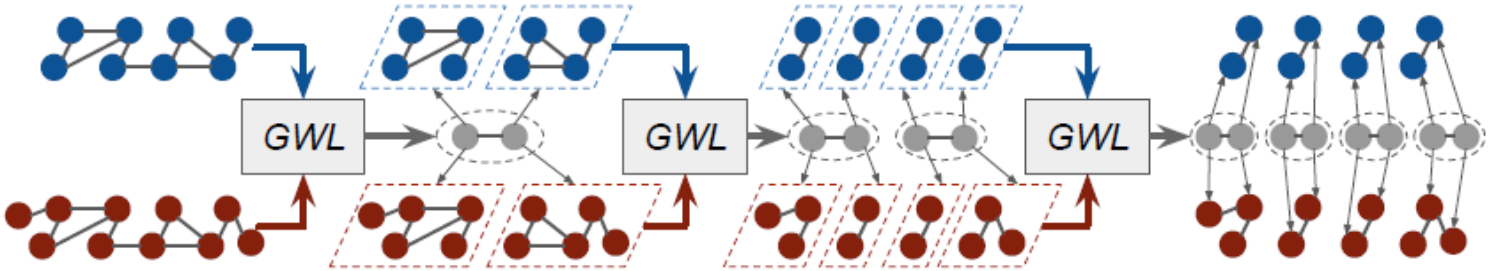

Figure 4: Scheme of S-GWL

- Introduction
- Deep Learning of Graph Matching
- Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach
- Scalable Gromov-Wasserstein Learning for Graph Partitioning and Matching
- Baselines

Baselines

- **Revolver: Vertex-Centric Graph Partitioning Using Reinforcement Learning**(2018)
  –paper: https://ieeexplore.ieee.org/document/8457880
  –code: https://github.com/hmofrad/revolver

- **Graph neural network based coarse-grained mapping prediction**(2020)
  –paper: https://pubs.rsc.org/en/content/articlehtml/2020/sc/d0sc02458a
  –code: https://github.com/rochesterxugroup/DSGPM

- **Graph Partition Neural Networks for Semi-Supervised Classification**(2018)
  –paper: https://arxiv.org/pdf/1803.06272.pdf
  –code: https://github.com/microsoft/graph-partition-neural-network-samples

- **GAP: Generalizable Approximate Graph Partitioning Framework**(2019)
  –paper:https://arxiv.org/pdf/1903.00614.pdf
  –code: https://github.com/saurabhdash/GCN_Partitioning