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

Deep Feature Extractor

In: I1, I2

 $\begin{aligned} \textbf{Out:} & \text{ features matrices } \textbf{F}^1, \textbf{F}^2 \text{ and } \\ \textbf{U}^1, \textbf{U}^2 & \text{ as computed by any CNN,} \\ & \text{at certain levels of the hierarchy,} \\ & \text{e.g. } VGG\text{-}16 \text{ [}29\text{]} \end{aligned}$

Affinity Matrix

In: $\mathbf{F}^1, \mathbf{F}^2, \mathbf{U}^1, \mathbf{U}^2$ Build graph structure: $\mathbf{G}_1, \mathbf{G}_2, \mathbf{H}_1, \mathbf{H}_2$ Computations: build \mathbf{M}_e and \mathbf{M}_p Out: \mathbf{M} as given by eq. (5)

Power Iteration

In: M
Computations: $\mathbf{v}_0 \leftarrow \mathbf{1}$, $\mathbf{v}_{k+1} = \mathbf{M}\mathbf{v}_k / |\mathbf{M}\mathbf{v}_k|$ Out: \mathbf{v}^*

Bi-Stochastic

In: v*

Computations: reshape v* to a matrix and apply eqs. (19)
Out: double-stochastic confidence matrix S

Voting

In: $\mathbf{S} \in \mathbb{R}^{n \times m}$

Computations: softmax(α S)
Parameters: scale α Out: displacement vector d as

given by eq. (22)

Loss

In: $\mathbf{d}, \mathbf{d}^{gt}$ Out: $L(\mathbf{d}) = \sum_{i} \phi(\mathbf{d}_{i} - \mathbf{d}_{i}^{gt})$

 \rightarrow

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

Shen Yuan Graph Matching June 10, 2021 4/17

Affinity Matrix Layer

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

where $A_1 \in \{0,1\}^{n \times n}$, $A_2 \in \{0,1\}^{m \times m}$ are the node-to-node adjacency matrices.

$$oldsymbol{M} = [\operatorname{vec}(oldsymbol{M}_p)] + (oldsymbol{G}_2 \otimes oldsymbol{G}_1)[\operatorname{vec}(oldsymbol{M}_e)](oldsymbol{H}_2 \otimes oldsymbol{H}_1)^{ op}$$

where the matrix $M_p \in \mathbb{R}^{n \times m}$ measures the node-to-node similarities, and $M_e \in \mathbb{R}^{p \times q}$ 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 $S_0 = (v^*)_{n \times m}$,

$$S_{k+1} = S_k[\mathbf{1}_n^{\mathsf{T}} S_k]^{-1}, S_{k+2} = [S_{k+1} \mathbf{1}_m]^{-1} S_{k+1}$$

Voting layer

$$oldsymbol{d}_i = rac{\exp[lpha oldsymbol{S}(i,1\dots m)]}{\sum_i \exp[lpha oldsymbol{S}(i,j)]} oldsymbol{P} - oldsymbol{P}_i$$

where $P \in \mathbb{R}^{m \times 2}$ 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

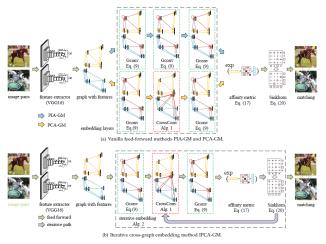


Figure 2: The pipelines.

Node Feature Extraction on Images

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

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

where Interp(P, X) bi-linearly interpolates on point P from the feature map X.

Intra-graph Node Embedding

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

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

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

 $\hat{m{S}} = \mathrm{Sinkhorn}(\hat{m{M}})$

where f_{aff} represents the affinity measure,

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

 $\hat{\boldsymbol{S}} = \text{Sinkhorn}(\hat{\boldsymbol{M}})$

where f_{aff} represents the affinity measure,

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

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

 $\hat{\boldsymbol{S}} = \text{Sinkhorn}(\hat{\boldsymbol{M}})$

where f_{aff} represents the affinity measure,

$$f_{ ext{aff}}(oldsymbol{h}_{1i},oldsymbol{h}_{2j}) = \exp\left(rac{oldsymbol{h}_{1i}^{ op}oldsymbol{A}oldsymbol{h}_{2j}}{ au}
ight)$$

$$oldsymbol{M}^{(k)\prime} = oldsymbol{M}^{(k-1)} \oslash (oldsymbol{M}^{(k-1)} oldsymbol{1} oldsymbol{1}^{ op}) \ oldsymbol{M}^{(k)} = oldsymbol{M}^{(k)\prime} \oslash (oldsymbol{1} oldsymbol{1}^{ op} oldsymbol{M}^{(k)\prime})$$

where \oslash means element-wise division,

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

- 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

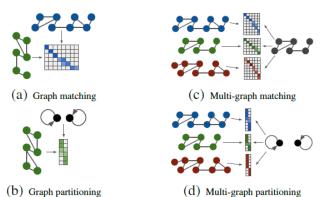


Figure 3: Applying the GWL framework into the graph partitioning and matching

Shen Yuan Graph Matching June 10, 2021 12/17

Gromov-Wasserstein discrepancy between graphs

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

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

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

Shen Yuan Graph Matching June 10, 2021 13/17

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 $\boldsymbol{\omega} = [\omega_m] \in \sum_{m=1}^{M}$ 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

$$\begin{split} \boldsymbol{T}^{(n+1)} &= \arg \min_{\boldsymbol{T} \in \prod(\boldsymbol{\mu}_{s}, \boldsymbol{\mu}_{t})} \sum_{i, j \in \mathcal{V}_{s}} \sum_{i', j' \in \mathcal{V}_{t}} |c_{ij}^{s} - c_{i'j'}^{t}|^{2} T_{ii'}^{(n)} T_{jj'} + \gamma \text{KL}(\boldsymbol{T} \| \boldsymbol{T}^{(n)}) \\ &= \arg \min_{\boldsymbol{T} \in \prod(\boldsymbol{\mu}_{s}, \boldsymbol{\mu}_{t})} \langle \boldsymbol{L}(\boldsymbol{C}_{s}, \boldsymbol{C}_{t}, \boldsymbol{T}^{(n)}), \boldsymbol{T} \rangle + \gamma \text{KL}(\boldsymbol{T} \| \boldsymbol{T}^{(n)}) \end{split}$$
where $\boldsymbol{L}(\boldsymbol{C}_{s}, \boldsymbol{C}_{t}, \boldsymbol{T}) = \boldsymbol{C}_{s} \boldsymbol{\mu}_{s} \mathbf{1}_{|\mathcal{V}_{t}|}^{\top} + \mathbf{1}_{|\mathcal{V}_{s}|} \boldsymbol{\mu}_{t}^{\top} \boldsymbol{C}_{t}^{\top} - 2 \boldsymbol{C}_{s} \boldsymbol{T} \boldsymbol{C}_{t}^{\top} \end{split}$

Shen Yuan Graph Matching June 10, 2021 15/17

A recursive K-partition mechanism

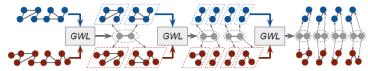


Figure 4: Scheme of S-GWL

- 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