Origin: Non-Rigid Network Alignment

Si Zhang  Hanghang Tong  Jiejun Xu  Yifan Hu  Ross Maciejewski
Network Alignment

- To find node correspondence across networks.
Network Alignment: Applications

- Fraud detection

• Unsuspicious patterns become suspicious!
Network Alignment: Applications

• Other applications

Drug design
[Kazemi et al. 2016]

Friend recommendation
[Yan et al. 2013]

Existing Methods

• Graph matching based methods
  • Koopmans-Beckmann’s quadratic assignment problem (KB-QAP)
    \[
    \max \quad \text{Tr}(S^T A_1 S A_2) + \text{Tr}(H^T S)
    \]
    \[
    s.t. \quad \text{constraints on } S
    \]

• Choices on constraints
  • \(S\) is a permutation matrix (exact constraint)
  • \(S\) is a doubly stochastic matrix (stochastic relaxation)
    \[
    S \in [0,1]^{n_1 \times n_2}, S1_{n_2} \leq 1_{n_1}, S^T 1_{n_1} = 1_{n_2}
    \]
  • \(S\) is an orthogonal matrix (spectral relaxation)
Existing Methods (con’t)

• Embedding based methods
  • Learn representations of nodes in different networks
  • Infer alignment by similarities among embedding vectors

(showcase from Liu et al. 2016)

Limitation #1: Representation Power

• Koopmans-Beckmann’s QAP

\[
\max \text{ Tr}(S^T A_1 S A_2) = \sum_{u,k} (S^T A_1)_{uk} (A_2 S^T)_{uk}
\]

• Node-\(u\) feature vector: \(u\)-th row of linear transformations \(S^T A_1 \& A_2 S^T\)

• Inner product of feature vectors computed from \(A_1\) and \(A_2\)

• Maximizing inner product similarities

• Limitations:
  • Linear transformation based on connections
  • High dimensions

• Question: How to learn better node representations?
Limitation #2: Representation Incomparability

- Single network embedding

- Intra-network node similarities do not change
- Semantically rotation/translation invariance
Limitation #2 (con’t)

- Multiple network embedding
  - Given $u$ is aligned with $v$

- Inter-network node similarities totally changed!

- **Question:** How to address the representation incomparability?
Prob. Def.: Non-Rigid Network Alignment

• Input:
  • (1) undirected networks $G_1 = \{V_1, A_1, X_1^0\}$ and $G_2 = \{V_2, A_2, X_2^0\}$;
  • (2) labeled aligned node pairs $L^+ = \{(u_{li}, v_{li})| i = 1, \ldots, L\}$;
  • (3) (optional) prior cross-network node similarity matrix $H$.

• Output:
  • (1) alignment matrix $S$;
  • (2) node representation matrices $Z, Y$ of $G_1, G_2$
Outline

• Motivations

• Model Overview

• Q1: Multiple Network Representation Learning

• Q2: Non-Rigid Point Set Alignment

• Experiments

• Conclusions
Model Overview

Multi-GCN

$G_1$

$G_2$

$\bar{x}_u \forall u \in G_1$

$\bar{y}_v \forall v \in G_2$

Multi-View Point Set Alignment

Point Set Alignment

Sampling & Normalization

Pseudo-Labeling
Outline

• Motivations
• Model Overview
• Q1: Multiple Network Representation Learning
• Q2: Non-Rigid Point Set Alignment
• Experiments
• Conclusions
Single Network GCN

• Spatial-based GCN formulation (Intra-GCN)

\[
\begin{align*}
\tilde{x}_{N_u}^t &= \text{Aggregate} \left( \{ \tilde{x}_{u'}^{t-1}, \forall u' \in N_u \} \right) \\
\tilde{x}_u^t &= \sigma \left( [\tilde{x}_u^{t-1} | \tilde{x}_{N_u}] \mathbf{W}^t \right)
\end{align*}
\]

• Aggregate hidden representations from neighborhood \( N_u \)
• Combine aggregated representation
• Limitations: only aggregate within a single network

• **Question:** How to aggregate across different networks?
Multi-GCN: Formulation #1

- Cross-network aggregation via alignment

\[
\hat{x}_u = \text{Aggregate}_{\text{cross}}(\tilde{y}_v) = \sum_{v \in V_2} S(u, v)\tilde{y}_v
\]

\[
\hat{y}_v = \text{Aggregate}_{\text{cross}}(\tilde{x}_u) = \sum_{u \in V_1} S(u, v)\tilde{x}_u
\]

- Sample on alignment \(S\) for aggregation localization and efficiency

- Cross-network combination

\[
x_u = \text{Combine}_{\text{cross}}(\tilde{x}_u, \hat{x}_u) = [\tilde{x}_u || \hat{x}_u]W_{\text{cross}} + b_1
\]

\[
y_v = \text{Combine}_{\text{cross}}(\tilde{y}_v, \hat{y}_v) = [\tilde{y}_v || \hat{y}_v]W_{\text{cross}} + b_2
\]
Multi-GCN: Formulation #2

- Multi-GCN loss function
  \[ J_{\text{GCN}} = J_{g_1}(X) + J_{g_2}(Y) + \lambda J_{\text{cross}}(X, Y) \]
- Inter-network loss
- Intra-network loss (e.g., SkipGram)
- Inter-network disagreement loss
  \[ J_{\text{cross}}(X, Y) = \sum_{u \in V_1} \left\| x_u - \sum_{k=1}^{K} S_1(u, v_{q_k}) y_{q_k} \right\|^2 + \sum_{v \in V_1} \left\| y_v - \sum_{k=1}^{K} S_2(u_{p_k}, v) x_{u_{p_k}} \right\|^2 \]

Cross-network
Within-network
by alignment
Outline

• Motivations
• Model Overview
• Q1: Multiple Network Representation Learning
• Q2: Non-Rigid Point Set Alignment
• Experiments
• Conclusions
Non-Rigid Point Set Alignment (NR-PSA)

• **Goal:** to address the representation incomparability

• **Key ideas:**
  • View node representation vectors as points in Euclidean space
  • Displace one point set towards another based on labeled alignment
  • Move coherently in two views (i.e., point view and node view)

![Diagram showing point set of $G_1$ and $G_2$.](image-url)
NR-PSA: Formulation #1

• Intuition: to maximize labeled node-pair overlaps
• Given labeled node alignment \((u_l, v_l), l = 1, \ldots, L\)

\[
\min_f \sum_{i=1}^{L} \left\| x_{ui} + \frac{1}{2} f(x_{ui}) - y_{vi} \right\|_2^2 + \alpha \| f \|_{\mathcal{H}}^2
\]

vector-valued non-rigid displacement function

• Minimize vector distances after displacement
• Functional minimization problem
• \(\| f \|_{\mathcal{H}}^2\) is the RKHS norm for regularization
NR-PSA: Formulation #2

• Intuition: each point $x_{u_l_i}$ has two interpretations (views)
  • Representation vectors in the Euclidean space
  • Nodes of networks in the non-Euclidean graph space
• Divide $\mathcal{H}$ into two RKHS, i.e., $\mathcal{H} = \mathcal{H}_1 \oplus \mathcal{H}_2$, such that
  
  $$\mathcal{H} = \{f | f(x) = f^1(x) + f^2(x), f^1 \in \mathcal{H}_1, f^2 \in \mathcal{H}_2\}$$

• Re-write RKHS norm regularization into
  
  $$\|f\|_{\mathcal{H}}^2 = \min_{f=f^1+f^2} \alpha_1 \|f^1\|_{\mathcal{H}_1}^2 + \alpha_2 \|f^2\|_{\mathcal{H}_2}^2 + \mu \sum_{j=1}^{n_1-L} [f^1(x_{u_{r_j}}) - f^2(x_{u_{r_j}})]^2$$

  displacement consistency in two views on the unlabeled nodes
NR-PSA: Formulation #2 (con’t)

• By representer theorem

\[ f(x_u) = K(u, J)T \]

• Matrix \( K \): kernel matrix computed by reproducing kernels in \( \mathcal{H}_1, \mathcal{H}_2 \)

• \( T \) is the matrix variable and \( J = \{ u_{li} \mid i = 1, \cdots, L \} \)

• Matrix-form objective function

\[
\min_T J_{PSA} = \sum_{i=1}^{L} \left\| x_{u_{li}} + \frac{1}{2} K(u_{li}, J)T - y_{v_{li}} \right\|_2^2 + \alpha \text{Tr}(T^T K_J T)
\]

• \( K_J = K(J, J) \)

• Details in the paper.
Origin: Algorithm

- Alternating between two stages
  - Stage #1: to learn node representations based on current alignment
  - Stage #2: to solve for the displacement function
- Stage #1: mini-batched SGD
- Stage #2: gradient descent

\[
\frac{\partial J_{PSA}}{\partial T} = \frac{1}{2} K_j^T K_j T + K_j^T (X(I,:) - Y(J,:)) + 2\alpha K_j T
\]

- Time complexity: sub-quadratic w.r.t # of nodes
- Outputs displaced node representations \( Z \) of \( G_1 \)
Outline

• Motivations
• Model Overview
• Q1: Multiple Network Representation Learning
• Q2: Non-Rigid Point Set Alignment
• Experiments
• Conclusions
Experiment Setup

• Datasets:
  • Cora-1 & Cora-2 networks (nodes: 2,708 vs. 2,708)
  • Citeseer-1 & Citeseer-2 networks (nodes: 3,327 vs. 3,327)
  • Foursquare & Twitter networks (nodes: 5,313 vs. 5,120)

• Evaluation objectives:
  • Effectiveness: alignment accuracy
  • Efficiency: running time

• Comparison methods:

<table>
<thead>
<tr>
<th>Methods</th>
<th>Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>Representation based</td>
</tr>
<tr>
<td>SageAlign</td>
<td>Representation based</td>
</tr>
<tr>
<td>FINAL-N [1]</td>
<td>KB-QAP based</td>
</tr>
<tr>
<td>FINAL-P [1]</td>
<td>KB-QAP based</td>
</tr>
<tr>
<td>REGAL [2]</td>
<td>Representation based</td>
</tr>
<tr>
<td>IONE [3]</td>
<td>Representation based</td>
</tr>
<tr>
<td>PriorSim</td>
<td>Heuristics</td>
</tr>
</tbody>
</table>

**R1. Effectiveness**

**Observation:** outperforms both QAP-based methods FINAL and other embedding-based methods.
R2. Visualizations

Observations:

- Embeddings $X, Y$ of $G_1, G_2$ are misleading even with cross-network disagreement loss;
- Displaced embeddings $Z, Y$ are more accurate for alignment.
R3. Efficiency

**Observation:** the extra computational cost for Inter-GCN is quite light.
Outline

- Motivations
- Model Overview
- Q1: Multiple Network Representation Learning
- Q2: Non-Rigid Point Set Alignment
- Experiments
- Conclusions
Conclusions

• Problem: Non-rigid network alignment
• Solutions (proposed Origin algorithm):
  • Multi-GCN: node representation learning across networks based on GCN
  • NR-PSA: non-rigid point-set alignment in two views
• Results:
  • Find more accurate node correspondence
  • Learn more meaningful node representations
  • Efficient compared to single-network counterpart
• More details in paper.
Thank You!