Graph Matching Networks for Learning the Similarity of Graph Structured Objects

Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, Pushmeet Kohli

**Introduction**

We want to learn a similarity / distance function between graphs.

Many applications:
- Similarity search in graph databases
- Copy detection for graph structured objects

Motivating problem:
- Binary function similarity search for detecting software vulnerabilities

Challenge:
- Reasoning about both graph structure as well as the graph semantics

Previous approaches:
- Graph hashes
- Graph kernels

**Synthetic Task: Learning Graph Edit Distance**

Learn a similarity metric that correlates with graph edit distance.
- Extreme case: distinguishing graph edit distance of 0 vs non-zero - graph isomorphism test.
- Graph edit distance is NP-hard in general.

Comparing graph matching model vs graph embedding model vs WL-kernel on random graphs to distinguish edit distance of 1s 2s.

<table>
<thead>
<tr>
<th>Graph Spec.</th>
<th>WL kernel</th>
<th>embedding model</th>
<th>matching model</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 20, p = 0.2</td>
<td>89.8 / 83.2</td>
<td>85.0 / 85.6</td>
<td>85.0 / 85.6</td>
</tr>
<tr>
<td>n = 30, p = 0.5</td>
<td>74.9 / 78.0</td>
<td>92.1 / 93.4</td>
<td>96.6 / 98.0</td>
</tr>
<tr>
<td>n = 50, p = 0.2</td>
<td>93.9 / 97.8</td>
<td>95.9 / 97.1</td>
<td>97.4 / 97.6</td>
</tr>
<tr>
<td>n = 50, p = 0.5</td>
<td>82.3 / 89.0</td>
<td>88.5 / 91.0</td>
<td>93.8 / 92.6</td>
</tr>
</tbody>
</table>

Measuring pair classification AUC / triplet accuracy.

**Learning**

Learn to make similar pairs have small distance (high similarity), and dissimilar pairs have high distance (small similarity).

Pairwise training:
\[ L_{pair} = E(G, G_{adj}) \max(0, \gamma - t(1 - d(G, G_{adj}))) \]
\[ t \in \{-1, +1\} : \text{label}, +1 \text{ for similar, otherwise -1}. \quad \gamma : \text{margin} \]

Triplet training:
\[ L_{triplet} = E(G, G, G_{adj}) \max(0, d(G, G_{adj}) - d(G, G_{adj}^*) + \gamma) \]
\( (G, G_{adj}) \text{ is a similar pair; } (G, G_{adj}^*) \text{ is a dissimilar pair} \quad \gamma : \text{margin} \]

**Attention Visualizations**

We never supervise the cross-graph attention, but the model still learns some interesting attention patterns.

**Conclusions, Limitations and Future Work**

Graph similarity can be learned with graph neural networks.

Graph Matching Networks perform better than embedding models.

GMN is more expensive compared to GNN embedding models, requiring \(O(V_1V_2)\) computation at each step.
- This provides us with an accuracy-computation trade-off

GMNs may be used jointly with GNNs embedding models in a retrieval system: GNN for fast filtering, GMN for refinement.

Future directions:
- Larger graphs
- More effective / scalable attention
- Different matching architectures
- Many more!

**The Models**

**GNN embedding model**

Map each graph to a vector representation, through a graph neural net using multiple message passing / graph convolution layers.

**Graph matching networks**

Cross-graph attention & comparison early in the message passing process.

**The Models**

**Graph Similarity Model**

\[ v_j^{(t)} = \text{MLP}_{\text{prop}}(x_j); \quad v_j^{(t)} \in \mathbb{R}^{d} \]

\[ v_j^{(t)} = \text{MLP}_{\text{prop}}(x_j); \quad v_k^{(t)} \in \mathbb{R}^{d} \]

\[ h_j^{(t)} = \text{MLP}_{\text{prop}}(v_j^{(t)}); \quad h_k^{(t)} = \text{MLP}_{\text{prop}}(v_k^{(t)}) \]

\[ h_j^{(t)} = \text{MLP}_{\text{prop}}(v_j^{(t)}); \quad h_k^{(t)} = \text{MLP}_{\text{prop}}(v_k^{(t)}) \]

\[ h_j = \text{fc}(h_j^{(T)}) = \text{MLP}(\sum_{j \in G_1} \text{MLP}_{\text{prop}}(h_j^{(T)}) \otimes \text{MLP}(h_k^{(T)})) \]

Cross-graph attention-based matching:

\[ \mathcal{A}_{jk} = \exp(h_j^{(T)}; h_k^{(T)}) \]

\[ \sum_{j \in V_1} \exp(h_j^{(T)}; h_k^{(T)}) \]

\[ \mathcal{B}_{jk} = \text{fc}(h_j^{(T)}; h_k^{(T)}) \]

\[ \mu_{j,k} = \text{fc}(h_j^{(T)}; h_k^{(T)}) \]

\[ \sum_{j \in V_1} \exp(h_j^{(T)}; h_k^{(T)}) \]

\[ \sum_{j \in V_1} \exp(h_j^{(T)}; h_k^{(T)}) \]

\[ \sum_{j \in V_1} \exp(h_j^{(T)}; h_k^{(T)}) \]