Hierarchical Graph Representation Learning via Differentiable Pooling

Rex Ying, Jiaxuan You, Christopher Morris, William L. Hamilton, Xiang Ren, Jure Leskovec

Stanford University
TU Dortmund University
University of Southern California
Motivation: ML for Graphs

- **Graph classification tasks:**
  - Molecule prediction
    - Classify molecule properties (toxicity, drug-likeness etc.)
  - Social networks
    - Predict social group properties
  - Biological applications
    - Model disease pathways in PPI networks
  - Physical systems
    - Evolving dynamical systems
Graph Neural Networks (GNNs) have revolutionized machine learning with graphs.

But GNNs learn individual node representations and then simply globally aggregate them:

- Mean/max/sum of all node embeddings (e.g. structure2vec)
- Pool by sorting (e.g. DGCNN, PatchySan)

**Problem:** How to aggregate information in a hierarchical way to capture the entire graph.
Pooling for GNNs

**Problem:** Learn a hierarchical pooling strategy that respects graph structure

**Our solution:** **DIFFPOOL**

- Learns hierarchical pooling analogous to CNNs
- Sets of nodes are pooled hierarchically
- Soft assignment of nodes to next-level nodes
A different GNN is learned at every level of abstraction

Our approach: Use two sets of GNNs
- GNN1 to learn how to pool the network
  - Learn cluster assignment matrix
- GNN2 to learn the node embeddings

DIFFPOOL Architecture
Assuming general GNN model:

\[ H^{(k)} = M(A, H^{(k-1)}; \theta^{(k)}) \]

Concretely: \( \text{ReLU}(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(k-1)} W^{(k-1)}) \)

Two-tower architecture

\[ Z^{(l)} = \text{GNN}_{l, \text{embed}}(A^{(l)}, X^{(l)}) \]
\[ S^{(l)} = \text{softmax} \left( \text{GNN}_{l, \text{pool}}(A^{(l)}, X^{(l)}) \right) \]

Aggregate embedding via assignment to generate next-level representations and adjacency
Experimental Results

An average of 6.27% improvement in accuracy for graph classification tasks on biological and social networks.

<table>
<thead>
<tr>
<th>Method</th>
<th>ENZYMES</th>
<th>D&amp;D</th>
<th>REDDIT-MULTI-12K</th>
<th>COLLAB</th>
<th>PROTEINS</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATCHYSAN</td>
<td>–</td>
<td>76.27</td>
<td>41.32</td>
<td>72.60</td>
<td>75.00</td>
<td>4.17</td>
</tr>
<tr>
<td>GRAPH SAGE</td>
<td>54.25</td>
<td>75.42</td>
<td>42.24</td>
<td>68.25</td>
<td>70.48</td>
<td>–</td>
</tr>
<tr>
<td>ECC</td>
<td>53.50</td>
<td>74.10</td>
<td>41.73</td>
<td>67.79</td>
<td>72.65</td>
<td>0.11</td>
</tr>
<tr>
<td>SET2SET</td>
<td>60.15</td>
<td>78.12</td>
<td>43.49</td>
<td>71.75</td>
<td>74.29</td>
<td>3.32</td>
</tr>
<tr>
<td>SORTPOOL</td>
<td>57.12</td>
<td>79.37</td>
<td>41.82</td>
<td>73.76</td>
<td>75.54</td>
<td>3.39</td>
</tr>
<tr>
<td>DIFFPOOL-Det</td>
<td>58.33</td>
<td>75.47</td>
<td>46.18</td>
<td>82.13</td>
<td>75.62</td>
<td>5.42</td>
</tr>
<tr>
<td>DIFFPOOL-NoLP</td>
<td>61.95</td>
<td>79.98</td>
<td>46.65</td>
<td>75.58</td>
<td>76.22</td>
<td>5.95</td>
</tr>
<tr>
<td>DIFFPOOL</td>
<td>62.53</td>
<td>80.64</td>
<td>47.08</td>
<td>75.48</td>
<td><strong>76.25</strong></td>
<td><strong>6.27</strong></td>
</tr>
</tbody>
</table>
Experimental Results

**DIFFPOOL** learns reasonable pooling architectures

Pooling at Layer 1

Pooling at Layer 2
Thank you!

Poster: AB #14

Code: https://github.com/RexYing/diffpool