More Expressive GNNs: Mitigating Oversquashing

S Deepak Narayanan, 22\textsuperscript{nd} March 2022

Mentor: Kenza Amara
Overview

• Introduction to Oversquashing
  • Oversquashing v/s Oversmoothing
  • An example problem and a simple rewiring solution
• More Solutions to Alleviate Oversquashing
  • Geometric GCN
  • Rewiring with Positional Encodings
• A curvature perspective on Oversquashing
What is Oversquashing?

• Aggregation in multi-hop GNNs involve large neighborhoods
• GNNs “compress” this information into a fixed-length vector
• Bottleneck $\rightarrow$ Loss of Information
• Consequence: Difficult to learn long range information


Picture taken from [1]
Oversmoothing v/s Oversquashing

Oversmoothing
• Refers to all node embeddings converging to similar vectors
• Found to occur with increasing number of layers
• Common in short-range tasks

Oversquashing
• Bottleneck caused due to information compression
• Issue more related to graph topology than # of GNN layers
• More relevant to long-range tasks
Demonstrating Oversquashing

• The Neighbors Match Problem
• Multiple Graphs; Labels are a function of the number of immediate blue neighbors
• A single layer GNN can count, but cannot infer the label!

Picture taken from [1]
Tree Neighbors Match

• Suppose the cloud is a binary tree
• We can control the “problem radius” \( r \) – minimum number of GNN layers needed to propagate sufficient information
• This example has Problem Radius 3
Tree Neighbors Match - Results

• Take GNNs of $k = r + 1$ layers
• Even $r = 4$ causes over-squashing
• Extent of Oversquashing depends on the aggregator

Picture taken from [1]
Tree Neighbors Match - Analysis

GIN and GCN suffer from oversquashing “before” GAT!

GAT can potentially *ignore* half the information
Rewiring: Solutions Outlook

Oversquashing Consequences
→ Capturing long-range dependencies are hard

• Rewiring involves modifying the underlying graph structure
• Modification: Changes the graph connectivity by addition or removal of edges to ease information flow
• Solution 1: A Fully Adjacent Layer
Solution 1: A Fully Adjacent (Last) Layer
Fully Adjacent (FA) Layer – Results

Datasets: Quantum Chemistry, Biological, Computer Programs
All of these datasets contain long-range problems

Table 2: Average accuracy (30 runs±stdev) on the biological datasets. † – previously reported by Errica et al. (2020).

Table 3: Average accuracy (5 runs±stdev) on VARMISUSE. † – previously reported by Brockschmidt (2020).
Fully Adjacent Layer - Analysis

• Impressive performance gains over SOTA
• Why not make all layers FA?
• Graph topology
  • Provides Relevant Inductive Bias $\rightarrow$ Regularization Effect
  • Empirically 1500% higher error with all layers FA!
• Pros: Simple, Easy to Implement 😊
• Cons: Computationally expensive for large graphs 😞

FA layer eases information flow and relieves bottleneck while retaining graph topology from previous layers

Results taken from [1]
Rewiring: Solutions Outlook

Oversquashing Consequences

→ Capturing long-range dependencies are hard

• Rewiring involves modifying the underlying graph structure
• Modification: Changes the graph connectivity by addition or removal of edges to ease information flow
• Solution 1: A Fully Adjacent Layer
• Solution 2: Geometric GCN
Geometric GCN

• Neighborhoods are defined by the underlying graph
• Q. Can we bring together nodes that are far apart but structurally similar, and involve them in aggregation?
• Such nodes are especially important in disassortative graphs (biological networks)
• More concretely, can we design modified neighborhoods over which aggregation can capture long-range dependencies?

Geometric GCN – Key Ideas

• Construct a latent space where structurally similar nodes appear together

• Exploit the underlying geometry of the latent space to define new neighborhoods for aggregation
Geometric GCN – Modules and Components

• Node Embedding Module
  • Maps the nodes to a latent continuous space

• Structural Neighborhood:
  • Graph defined neighborhood and Latent space neighborhood

• Bi-Level Aggregation: This module has two levels of aggregation
  • Low-Level Aggregation: Nodes from the same neighborhood and same geometric relationship are aggregated into a virtual node
  • High-Level Aggregation: Features are aggregated from virtual nodes to generate final representations
Geometric GCN – Modules

• Node Embedding Module $f$: For Graph $G = (V, E)$, define $f: v \rightarrow z_v$, $z_v \in \mathbb{R}^d$, $v \in V$

• Neighborhood of a node: $N(v) = (\{N_g(v), N_s(v)\}, \tau)$. Here, $N_g(v), N_s(v)$ are respectively the graph neighborhood and the latent space neighborhood

• $N_s(v) = \{u | u \in V, d(z_u, z_v) < \rho\}$, where $d(z_u, z_v)$ is the distance metric in the latent space

• $\tau$ is a relational operator; $\tau: (z_u, z_v) \rightarrow r \in R$, where $R$ is the set of geometric relations; E.g.: Direction w.r.t target node
Geometric GCN – Aggregation

• Low Level Aggregation:
  \[ e_{i,r}^v = p(\{h_u | u \in N_i(v), \tau(z_v, z_u) = r\}) \]
  \[ \forall i \in \{g, s\}, \forall r \in R \]

• High Level Aggregation:
  \[ m_v = q(e_{i,r}^v, (i, r)) \forall i \in \{g, s\} \forall r \in R \]

• Representation for layer L:
  \[ h_v^L = \sigma(m_v^L; W^L) \]
Advantages of a geometric neighborhood

• Geometric neighborhood informs every edge differently
• Can distinguish different graphs, even with same aggregator!

Picture Taken from [2]
Geometric GCN – Empirical Results

- GeomGCN with GCN aggregator outperforms GAT and GCN significantly; Huge performance gains in disassortative graphs
- Design of Latent Space method is critical for performance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>85.77</td>
<td>73.68</td>
<td>88.13</td>
<td>28.18</td>
<td>23.96</td>
<td>26.86</td>
<td>52.70</td>
<td>52.16</td>
<td>45.88</td>
</tr>
<tr>
<td>GAT</td>
<td>86.37</td>
<td>74.32</td>
<td>87.62</td>
<td>42.93</td>
<td>30.03</td>
<td>28.45</td>
<td>54.32</td>
<td>58.38</td>
<td>49.41</td>
</tr>
<tr>
<td>Geom-GCN-I</td>
<td>85.19</td>
<td>77.99</td>
<td>90.05</td>
<td>60.31</td>
<td>33.32</td>
<td>29.09</td>
<td>56.76</td>
<td>57.58</td>
<td>58.24</td>
</tr>
<tr>
<td>Geom-GCN-P</td>
<td>84.93</td>
<td>75.14</td>
<td>88.09</td>
<td>60.90</td>
<td>38.14</td>
<td>31.63</td>
<td>60.81</td>
<td>67.57</td>
<td>64.12</td>
</tr>
<tr>
<td>Geom-GCN-S</td>
<td>85.27</td>
<td>74.71</td>
<td>84.75</td>
<td>59.96</td>
<td>36.24</td>
<td>30.30</td>
<td>55.68</td>
<td>59.73</td>
<td>56.67</td>
</tr>
</tbody>
</table>

Results taken from [2]
Geometric GCN – Summary and Analysis

• Alleviates Oversquashing by rewiring the graph based on a structural neighborhood 😊

• Requires manual design of geometric relations 😞
  - E.g. Direction from Target Node

• Performance heavily depends on the embedding module
  - Current work uses methods such as Struct2Vec, Poincare, and Iso-map, all of which have strong inductive biases
Rewiring: Solutions Outlook

Oversquashing Consequences

→ Capturing long-range dependencies are hard

• Rewiring involves modifying the underlying graph structure
• Modification: Changes the graph connectivity by addition or removal of edges to ease information flow
• Solution 1: A Fully Adjacent Layer
• Solution 2: Geometric GCN
• Solution 3: Rewiring with Positional Encodings
Rewiring with Positional Encodings - Preliminaries

• Receptive field of a node is its immediate neighbors from which it aggregates information
• To reduce oversquashing, increase receptive field to the entire graph
  • Issues: Poor performance and introduces computational load
• Trade some compute for increasing receptive fields, while reducing oversquashing

Positional Encodings in GNNs

• Additional information about graph topology provided either as node/edge attribute for GNNs
• Examples: Laplacian Spectra, Node Degrees, Shortest Path Lengths
• Increase GNN expressivity
Rewiring with Positional Encodings: Key Ideas

• Increase receptive field to a $k$-hop neighborhood where $k \ll D$ and $D$ is the diameter of the graph → Ease information flow

• Introduce positional encodings → More expressivity and preserve graph topology

• Introduce Virtual Nodes → Ease information flow
Increasing Receptive Fields: Idea I

Expanded Receptive Field

• Given a graph $G = (V, E, f_v, f_e)$, with node attributes $f_v$ and edge attributes $f_e$, add edges between all nodes within $k$-hops of each other to create $G' = (V, E', f'_v, f'_e)$

• Set constant feature $C_e \forall e \in E' \setminus E$

• Virtual Node: Add a virtual node $v_{CLS}$ to $V$, and add an edge between $v_{CLS}$ and every other node

• Set $f'_v(v_{CLS}) = C_v$ for some constant $C_v$
Introducing Positional Encoding: Aims

• Lossless encodings – Be able to recover the original graph

• Discriminative Power: Encodings should improve the discriminative power measured in terms of the 1-WL Test

• Context Range: Global or Local Information
Positional Encoding: Idea II - Options

• Shortest Path Encodings (SPE):
  • Edge positional encoding denoting the shortest path distance between two nodes in the graph
  • Lossless encoding as we can recover $G$ from $G'$ when this encoding is 1
  • Expanded receptive fields + SPE is more powerful than the 1-WL Test

• Spectral Embeddings:
  • Node positional encoding consisting eigenvectors of the Laplacian
  • Not necessarily lossless, but works well in practice
  • Contains Global Information about the entire graph
Positional Encoding: Idea II - Options

• Powers of the Adjacency Matrix:
  • Edge positional encoding generalizing SPE
  • Captures number of paths between a pair of nodes
  • Lossless encoding, since we can recover $G$ from $G'$ using the first power of A
Positional Encoding: Options - Summary

• Shortest Path Encodings – Edge-based, Lossless, Expressive, Local Context
• Spectral Embeddings – Node-based, Lossless*, Global Context
• Powers of the Adjacency Matrix – Edge-based, Lossless, Generalizes SPE, Expressive, Controllable Context
Empirical Results – Benchmark Datasets

• Strong empirical performance with fewer parameters
• Empirically required receptive field has size $k \ll D$
• Encoding primarily depends on the dataset

Table 2: Benchmarking. Higher is better for all but for ZINC where lower is better. All results can be found in (Dwijedi et al., 2020; Corso et al., 2020; Bouritsas et al., 2020) and the leaderboard at this link. The benchmarks and corresponding leaderboard have 100K and 500K parameter entries, with many models only appearing with a subset of datasets or number of parameters; dashes indicate that the result for corresponding model, number of parameters, and dataset was not found in their paper or on the leaderboard.

Results taken from [3]
Empirical Results – Tree Neighbors Match

• In the figure, $r$ is the receptive field, and $r_p$ is the problem radius
• With $r = 1$, and $v_{CLS}$, similar performances of $r = 2$, and $r = 3$

Figure 1. NeighborsMatch (Alon and Yahav, 2021). Benchmarking the extent of over-squashing via the problem radius $r_p$.

Results taken from [3]
Rewiring with Positional Encodings - Summary

• Reduces Oversquashing by increasing receptive fields 😊
• Provides neat incorporation of positional encodings that can theoretically make the GNN more expressive 😊
• Need to manually tune over receptive fields sizes 😞
• Positional Encodings are application specific in practice 😞
Rewiring: Solutions Outlook

Oversquashing Consequences
→ Capturing long-range dependencies are hard

• Rewiring involves modifying the underlying graph structure
• Modification: Changes the graph connectivity by addition or removal of edges to ease information flow
• Solution 1: A Fully Adjacent Layer
• Solution 2: Geometric GCN
• Solution 3: Rewiring with Positional Encodings
• A Curvature Perspective of Oversquashing
A curvature perspective of Oversquashing

Curvature – Sensitivity

- Sensitivity $\left| \frac{\partial h_u}{\partial x_s} \right|$ can quantify how much a node representation is affected by other node representations.
- Sensitivity is bounded by powers of adjacency matrix $\rightarrow$ Graph Topology is responsible!
- Smaller value potentially indicates oversquashing.
Ricci Curvature of a Manifold

Curvature is characterized by “geodesic dispersion”
Geodesic Dispersion on Graphs

Clique (>0)

Grid (=0)

Tree (<0)

Picture Taken from [5]
Curvature for Graphs and Take-Away

• Define a version of curvature for edges based on geometric cues
  • For Spherical Geometry, Count Triangles
  • For Euclidean Geometry, Count 4-cycles
  • For Hyperbolic Geometry, Count # of outgoing edges

• Sensitivity is related to graph specific curvature

• Conclusion: Negatively curved edges $\rightarrow$ Cause oversquashing
Rewiring: Solutions Outlook

Oversquashing Consequences
→ Capturing long-range dependencies are hard

• Rewiring involves modifying the underlying graph structure
• Modification: Changes the graph connectivity by addition or removal of edges to ease information flow
• Solution 1: A Fully Adjacent Layer
• Solution 2: Geometric GCN
• Solution 3: Rewiring with Positional Encodings
• A Curvature Perspective of Oversquashing
  • Solution 4: Curvature Based Rewiring
A Curvature Based Rewiring Solution

Stochastic Discrete Ricci Flow (SDRF)

• Before training a GNN, rewire the graph as follows
  • Consider a most bottlenecked edge
  • Add an edge that can provide a high improvement to curvature
  • Remove the least bottlenecked edge under certain conditions
  • Repeat

• The above procedure greedily increases curvature and reduces over-squashing by providing better connectivity 😊
Empirical Results

<table>
<thead>
<tr>
<th>H(G)</th>
<th>Cornell</th>
<th>Texas</th>
<th>Wisconsin</th>
<th>Chameleon</th>
<th>Squirrel</th>
<th>Actor</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>52.69 ± 0.21</td>
<td>61.19 ± 0.49</td>
<td>54.60 ± 0.86</td>
<td>41.33 ± 0.18</td>
<td>30.32 ± 0.99</td>
<td>23.84 ± 0.43</td>
<td>81.89 ± 0.79</td>
<td>72.31 ± 0.17</td>
<td>78.16 ± 0.23</td>
</tr>
<tr>
<td>Undirected</td>
<td>53.20 ± 0.53</td>
<td>63.38 ± 0.87</td>
<td>51.37 ± 1.15</td>
<td>42.02 ± 0.30</td>
<td>35.53 ± 0.78</td>
<td>21.45 ± 0.47</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+FA</td>
<td>58.29 ± 0.49</td>
<td><strong>64.82 ± 0.29</strong></td>
<td>55.48 ± 0.62</td>
<td>42.67 ± 0.17</td>
<td>36.86 ± 0.44</td>
<td>24.14 ± 0.43</td>
<td>81.65 ± 0.18</td>
<td><strong>70.47 ± 0.18</strong></td>
<td><strong>79.48 ± 0.12</strong></td>
</tr>
<tr>
<td>DIGL (PPR)</td>
<td>58.26 ± 0.50</td>
<td>62.03 ± 0.43</td>
<td>49.53 ± 0.27</td>
<td>42.02 ± 0.13</td>
<td>33.22 ± 0.14</td>
<td>24.77 ± 0.32</td>
<td><strong>83.21 ± 0.27</strong></td>
<td><strong>73.29 ± 0.17</strong></td>
<td>78.84 ± 0.08</td>
</tr>
<tr>
<td>DIGL + Undirected</td>
<td><strong>59.54 ± 0.64</strong></td>
<td>63.54 ± 0.38</td>
<td>52.23 ± 0.54</td>
<td>42.68 ± 0.12</td>
<td>32.48 ± 0.23</td>
<td>25.45 ± 0.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDRF</td>
<td>54.60 ± 0.39</td>
<td>64.46 ± 0.38</td>
<td>55.51 ± 0.27</td>
<td><strong>42.73 ± 0.15</strong></td>
<td><strong>37.05 ± 0.17</strong></td>
<td><strong>28.42 ± 0.75</strong></td>
<td><strong>82.76 ± 0.23</strong></td>
<td><strong>72.58 ± 0.20</strong></td>
<td>79.10 ± 0.11</td>
</tr>
<tr>
<td>SDRF + Undirected</td>
<td>57.54 ± 0.34</td>
<td><strong>70.35 ± 0.60</strong></td>
<td>61.55 ± 0.86</td>
<td>44.46 ± 0.17</td>
<td><strong>37.67 ± 0.23</strong></td>
<td><strong>28.35 ± 0.06</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: Experimental results on common node classification benchmarks. Top two in bold.

Results Taken from [4]
A curvature perspective of Oversquashing - Summary

• Establishes links between geometry and graph topology
• Leads to a very simple algorithm for rewiring
• These links can prove a better understanding using tools from spectral graph theory
• Lots of interesting theoretical directions
Summary: Outlook on Rewiring and Solutions

Oversquashing Consequences

→ Capturing long-range dependencies are hard

• Rewiring involves modifying the underlying graph structure
• Modification: Changes the graph connectivity by addition or removal of edges to ease information flow

• Solution 1: A Fully Adjacent Layer
• Solution 2: Geometric GCN
• Solution 3: Rewiring with Positional Encodings
• A Curvature Perspective of Oversquashing
  • Solution 4: A Curvature Inspired Rewiring Solution