Graph Learning with Topological Regularization

Dilan Karaguler

1 Project Overview

Project Title: Graph Learning with Topological Regularization

GitHub Repository: https://github.com/DilanKaraguler/cmse802$_p$*roject.git*

Brief Project Description: This project explores the integration of topological invariants such as Betti numbers and Mayer Betti numbers into graph learning models using topological regularization. The goal is to incorporate persistent Mayer homology as a regularization term in Graph Neural Networks (GNNs) to preserve topological features during learning. This is particularly useful for applications where the topology of the graph plays a key role, such as in social networks, biological networks, or citation graphs.

2 Project Setup

2.1 Repository Structure

```
project_root/

 data/
    raw/
    processed/
    external/

 notebooks/
    exploratory/
    final/

 experiments/
    exp_001/
    exp_002/

 src/
    data_loading/
       data_read.py
    topology/
       path_complexes.py
```

```
      boundary_maps.py
      betti_numbers.py
   model_architecture/
   evaluation/
   training/

tests/

results/
   figures/
   models/

config/

.gitignore
README.md
requirements.txt
```

2.2 Key Files and Directories

- **src/data_loading/**: Scripts for reading molecule data and generating point cloud datasets.
- **src/topology/**: Contains scripts to compute path complexes and topological invariants like Betti numbers.
- **experiments/**: Contains the experiment directories with configurations for each experiment run.
- **results/**: Stores generated figures, models, and logs from experiments.

2.3 Dependencies and Setup Instructions

2.3.1 Dependencies

The project requires the following libraries:

- **Python 3.8+**
- **NumPy**: Used for numerical computations and array handling.
- **GUDHI**: For computing topological invariants such as Betti numbers.
- **PyTorch Geometric** (optional): Used for building graph-based models like GNNs.

2.3.2 Setup Instructions

To set up the project environment:

1. Clone the repository:

```
git clone https://github.com/DilanKaraguler/cmse802_project.git
cd cmse802_project
```

2. Create and activate a virtual environment:

```
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

3. Install the required dependencies:

```
pip install -r requirements.txt
```

4. Run the code:

```
python src/topology/path_complexes.py
```

3 Completed Tasks

- **Literature Review:** Reviewed three key papers (Persistence Enhanced Graph Neural Network, PersLay, and Graph Filtration Learning) relevant to topological regularization in GNNs.
  - **Work 1:** Persistence Enhanced Graph Neural Network by Qi Zhao, Ze Ye, Chao Chen, Yusu Wang.

    Summary: Propose a new network architecture which learns to use persistent homology information to reweight messages passed between graph nodes during convolution. For node classification tasks, this network outperforms existing ones on a broad spectrum of graph benchmarks.
  - **Work 2:** PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures by Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, Yuhei Umeda

    Summary: Relying on extended persistence theory and the so-called heat kernel signature, we show how graphs can be encoded by (extended) persistence diagrams in a provably stable way. They propose a general and versatile framework for learning vectorizations of persistence diagrams, which encompasses most of the vectorization techniques used in the literature.

– **Work 3:** Graph Filtration Learning by Christoph D. Hofer, Florian Graf,Bastian Rieck, Marc Niethammer, Roland Kwitt

Summary: They propose a readout operation for graph classification using persistent homology with a learnable filter function. They make this topological feature extraction differentiable, and empirical results show it outperforms previous methods, particularly when graph structure is important for learning.

- **Topological Invariants Computation:** Implemented methods to compute Betti numbers and path complexes updating GUDHI.

4 Proposed Approach

Model: A Graph Neural Network (GNN) integrated with topological features (Betti numbers and path complexes). These topological features will be injected either into the input layer or hidden layers of the GNN.

Justification: GNNs are suitable for graph-based data, but they struggle to capture higher-order topological structures like loops. Incorporating topological invariants helps to preserve such structures during learning.

4.1 Evaluation Strategy

Performance Metrics

- **Classification Accuracy**: Measure the percentage of correct predictions.
- **Precision, Recall, F1-Score**: Evaluate the model's performance, especially for imbalanced data.

Specific Tests

- **Cross-validation**: Perform k-fold cross-validation to ensure robust evaluation.
- **Confusion Matrix**: For classification tasks, visualize performance across different classes.
- **Ablation Study**: Compare the model's performance with and without topological features.

5 Preliminary Results

Initial Experiments: The initial experiments focused on testing the effectiveness of incorporating topological invariants, such as Betti numbers and path complexes. These experiments were conducted on different types of path complexes induced by directed graphs .

The results suggest that topological invariants, like Betti numbers, enhanced the ability to capture higher-order structures (such as loops or voids), which focus mainly on local node connectivity.

6 Challenges and Solutions

6.1 Challenges

- **Topological Feature Computation:** Computing Betti numbers for large datasets was computationally intensive.
- **Integrating Topological Features into GNN:** Incorporating topological invariants as regularizers required careful tuning of the regularization term.

6.2 Solutions

- Used GUDHI's optimized routines for efficient computation of Betti numbers.
- Developed a flexible framework for controlling the strength of the topological regularization.

7 Next Steps

- By introducing these topological descriptors as new features or modifying the input layer of the GNN to include these invariants, determine whether the model will show improved performance in classifying graph structures.

7.1 Updated Timeline

Milestone	Date	Details
Data Acquisition & Preprocessing	Week 1	Collect data, preprocess graphs, and extract topological features.
Baseline Model Implementation	Week 2-3	Implement GNN without topological features and run initial tests.
Topology Feature Integration	Week 4-5	Incorporate Betti numbers/path complexes and retrain the model.
Model Evaluation	Week 6	Test the model with cross-validation and analyze results.
Final Report & Presentation	Week 7	Prepare documentation and present findings.

8 Conclusion

The project is progressing well, with initial results showing the promise of integrating topological regularization into GNNs. Key challenges around computational complexity have been addressed, and the next steps focus on optimizing the computation of path complexes.