

RAGNOR: Robust Aggregation Graph Norm for Outlier Recognition

1 Code

Code can be available in <https://anonymous.4open.science/r/RAGNOR-1ED8/Readme.md>

2 Introduction

This repository is a modified version of the official codebase for “**NODESAFE: Bounded and Uniform Energy-based Node-level Out-of-distribution Detection on Graphs**”. The modifications integrate **RAGNOR (Robust Aggregation Graph Norm for Outlier Recognition)**, a post-hoc OOD detection framework that leverages node embedding norms from pre-trained GNNs and refines them through global Z-score normalization, median-based local aggregation, and optional multi-hop blending to provide robust OOD scores.

3 Relationship to NODESAFE

This codebase builds upon the NODESAFE framework, maintaining its experimental setup and evaluation pipeline while adding RAGNOR as an additional method. The original NODESAFE functionality remains intact, allowing for direct comparison between RAGNOR, NODESAFE, and other baseline methods within the same framework.

4 Overview

RAGNOR is a post-hoc OOD detection method that:

- Works with any pre-trained GNN backbone
- Uses node embedding norms as the foundation for OOD detection
- Applies global Z-score normalization for consistent scoring
- Employs median-based local aggregation for robustness
- Supports multi-hop reference blending for enhanced detection

5 Installation

5.1 Requirements

- Python 3.8+
- CUDA 12.1 (optional, for GPU support)
- PyTorch 1.12.0

- PyTorch Geometric 2.1.0.post1
- OGB 1.3.6
- scikit-learn 1.0.2
- SciPy 1.7.3
- NumPy 1.21.5

5.2 Setup

1. Clone the repository:

```
1 git clone <repository-url>
2 cd RAGNOR
```

2. Install dependencies:

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

6 How RAGNOR Works

RAGNOR operates in two phases:

1. **Training Phase:** Train a standard GNN backbone for node classification
2. **Detection Phase:** Apply RAGNOR's post-hoc OOD detection on the trained model

6.1 Key Components

- **Global Z-score Normalization:** Standardizes embedding norms across the graph
- **Median-based Local Aggregation:** Uses neighborhood information for robust scoring
- **Multi-hop Blending:** Optional feature to incorporate extended neighborhood context

7 Running RAGNOR

7.1 Basic Usage

Run RAGNOR for node-level OOD detection:

```
1 python main.py \
2     --method ragnor \
3     --backbone gcn \
4     --dataset cora \
5     --ood_type structure \
6     --mode detect \
7     --use_bn \
8     --device 0 \
9     --epochs 200 \
10    --lr 0.01 \
11    --hidden_channels 64 \
12    --num_layers 2 \
13    --dropout 0.5 \
14    --ragnor_epsilon 1e-8 \
15    --ragnor_lambda_blend 1.0
```

7.2 Key Parameters

7.2.1 RAGNOR-specific Arguments

- `-method ragnor`: Activates RAGNOR method
- `-ragnor_epsilon`: Small constant for numerical stability in Z-score normalization (default: $1e-8$)
- `-ragnor_lambda_blend`: Blending factor for multi-hop reference (default: 1.0)
 - 1.0 = 1-hop only
 - 0.5 = equal weighting of 1-hop and 2-hop
 - 0.0 = 2-hop only

7.2.2 General Arguments

- `-backbone`: GNN architecture (`gcn`, `gat`, `sage`)
- `-dataset`: Dataset name (`cora`, `citeseer`, `pubmed`, `ogbn-proteins`)
- `-ood_type`: Type of OOD nodes (`structure`, `feature`, `label`)
- `-mode`: Running mode (`detect` for OOD detection)
- `-hidden_channels`: Hidden dimension size
- `-num_layers`: Number of GNN layers
- `-epochs`: Training epochs
- `-lr`: Learning rate