EFLEC: Efficient Feature-LEakage Correction in GNN-based Recommendation Systems

Ishaan Kumar[12], Yaochen Hu[1], Yingxue Zhang
Huawei Technologies Canada

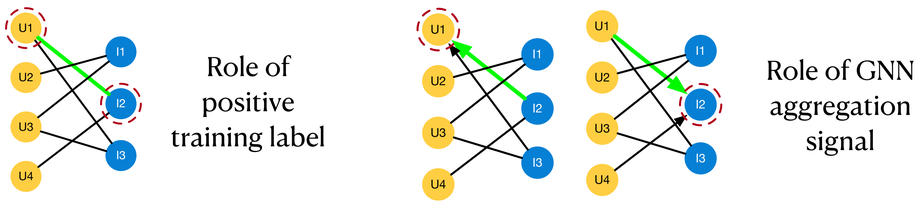
1 Both authors contributed equally to this work; 2 work was done when Ishaan Kumar was working at Huawei Technologies Canada.

Introduction

Graph Convolutional Neural Networks (GNN) based recommender systems are state-of-the-art. However, they suffer from the feature leakage problem since label information determined by edges can be leaked into node embeddings through the GNN aggregation procedure guided by the same set of edges, leading to poor generalization.

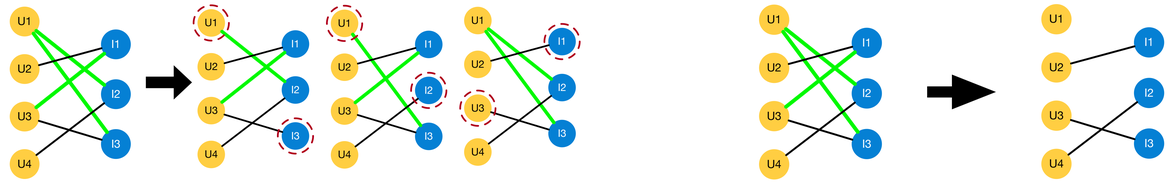
We propose an efficient accurate removal algorithm to generate the final embedding to solve the feature leakage problem.

Feature Leakage Problem



The proxy of positive labels appears as an implicit feature in the training procedure. The main reason is that each edge (the green edge in this example) performs dual roles in the training procedure. The green edge serves a role to set one of the positive training labels. On the other hand, it also provides a signal for GNN aggregation, which will reinforce the similarity between the embeddings of the two end nodes of the green edge. This reinforcement never appears in the inferencing phase between two potentially similar nodes, leading to degraded generalization.

Naive Solutions



Accurate removal

Generate a separate graph for each edge when it acts as a training label and removes it.

- Pros: perfectly removes the leakage
- Cons: computational complex

Sample and Removal

In each min batch, generate a graph with all the edges as training labels removed.

- Pros: efficient computation
- Cons: aggressive removal ignores a substantial amount of information

Is it possible to use the graph information as an accurate removal method while keeping an acceptable computation complexity as the sample and removal method?

Our Algorithm and Experiment Results

We found the relation between the node embeddings from the original graph and those after the accurate removal method, and propose a dynamic programming method to efficiently evaluate the embedding from accurate removal. Empirical results demonstrate that our algorithm can improve the performance on sparse datasets while the computation time is close to the vanilla algorithm without correction.

Table 1: Mean results of recall@20, nDCG@20, and time per epoch (T) in seconds. Bold represent the best and underline represents the second best. Vanila is not considered in the ranking for time.

	Method	Instant			Instrument			Yelp			Gowalla		
		Recall	nDCG	T(s)	Recall	nDCG	T(s)	Recall	nDCG	T(s)	Recall	nDCG	T(s)
2 Layers	Vanilla	0.1698	0.0805	2.96	0.0392	0.0187	2.75	0.0577	0.0467	110.13	0.1623	0.1375	82.65
	DropEdge	0.1656	0.0776	4.43	0.0465	0.0216	4.57	0.0581	**0.0469**	259.93	0.1622	0.1375	130.01
	S&R	**0.2202**	**0.1047**	5.23	**0.0541**	**0.0257**	4.62	0.0576	0.0466	444.94	0.1628	**0.1379**	259.49
	EFLEC	**0.2207**	**0.1029**	**3.11**	**0.0546**	**0.0260**	**3.06**	**0.0583**	**0.0469**	**122.75**	0.1630	**0.1382**	**72.88**
3 Layers	Vanilla	0.1776	0.0874	4.52	0.0471	0.0216	3.55	**0.0604**	**0.0489**	136.23	0.1677	0.1414	67.15
	DropEdge	0.1806	0.0825	**3.65**	0.0521	0.0241	**3.70**	**0.0603**	**0.0487**	219.73	**0.1690**	**0.1422**	105.28
	S&R	**0.2160**	**0.1059**	5.72	**0.0574**	**0.0270**	5.18	0.0600	0.0485	465.03	**0.1687**	**0.1420**	190.42
	EFLEC	**0.2155**	**0.1046**	4.56	**0.0573**	**0.0271**	4.15	0.0602	0.0485	**145.11**	**0.1689**	**0.1422**	**71.06**