Adversarial Graph Perturbations for Recommendations at Scale

Huiyuan Chen
Visa Research
hchen@visa.com

Kaixiong Zhou
Rice University
Kaixiong.Zhou@rice.edu

Kwei-Herng Lai
Rice University
khlai@rice.edu

Xia Hu
Rice University
xia.hu@rice.edu

Fei Wang
Visa Research
feiwang@visa.com

Hao Yang
Visa Research
haoyang@visa.com

ABSTRACT

Graph Neural Networks (GNNs) provide a class of powerful architectures that are effective for graph-based collaborative filtering. Nevertheless, GNNs are known to be vulnerable to adversarial perturbations. Adversarial training is a simple yet effective way to improve the robustness of neural models. For example, many prior studies inject adversarial perturbations into either node features or hidden layers of GNNs. However, perturbing graph structures has been far less studied in recommendations.

To bridge this gap, we propose AdvGraph to model adversarial graph perturbations during the training of GNNs. Our AdvGraph is mainly based on min-max robust optimization, where an universal graph perturbation is obtained through an inner maximization while the outer optimization aims to compute the model parameters of GNNs. However, direct optimizing the inner problem is challenging due to discrete nature of the graph perturbations. To address this issue, an unbiased gradient estimator is further proposed to compute the gradients of discrete variables. Extensive experiments demonstrate that our AdvGraph is able to enhance the generalization performance of GNN-based recommenders.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Theory of computation** → **Adversarial learning**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Collaborative Filtering; Graph Neural Networks; Adversarial Training; Discrete Optimization

ACM Reference Format:
Huiyuan Chen, Kaixiong Zhou, Kwei-Herng Lai, Xia Hu, Fei Wang, and Hao Yang. 2022. Adversarial Graph Perturbations for Recommendations at Scale. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3477495.3531763

SIGIR '22, July 11–15, 2022, Madrid, Spain

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531763

1 INTRODUCTION

Graph Neural Networks (GNNs) are widely used for reommender systems, owing to their ability of exploiting high-order structural relations between users and items [13, 21, 27, 31]. Given a user-item bipartite graph, GNNs perform graph convolutions to propagate messages over the graph, and refine the representation of each node with its neighbors [18]. This recursive message passing schema has fueled the development of GNN-based recommenders, like PinSage [31], NGCF [27], LightGCN [13], SGCN [4] and UltraGCN [21], which have achieved state-of-the-art performance.

However, GNNs are notoriously vulnerable to adversarial attacks [5, 35]. This implies that the performance of GNNs can significantly degrade by slightly injecting adversarial perturbations on either node's features or graph structures [35]. The lack of robustness of GNNs open the door for attackers to exploit their vulnerabilities [10, 36]. Several robust networks have been proposed to defend against adversarial attacks, including Gaussian graph convolution [34] and low-rank approximation [17]. Even so, most of architectures may fail to counteract complex global attacks [11].

Adversarial training has empirically shown great potential in boosting the robustness and generalization ability of neural networks [3, 14, 19, 20, 22]. Unlike its great success in image data [22], adversarial training cannot be directly applied to graph data due to their discrete adjacency structures. An alternative strategy is to generate perturbations on continuous spaces, such as node features [10, 19] and hidden layers [16], during the training stage of GNNs. Meanwhile, several recent studies aim to conduct discrete graph perturbations using meta-learning [36] or dropping edges [5]. However, these algorithms are often greedy, leading to sub-optimal results. It remains unclear how to efficiently reap the benefits of adversarial training for GNNs, especially for large-scale graphs.

Present Work: Here we propose a general framework, named AdvGraph, to model adversarial graph perturbations for training GNNs. Our AdvGraph is mainly based on min-max robust optimization [22, 25], where an *universal* graph perturbation is obtained through an inner maximization while the outer optimization aims to compute the model parameters of GNNs. Although the inner problem involves discrete variables, we show that their gradients can still be efficiently approximated by using discrete Antithetic REINFORCE-based solvers [7, 8, 30]. Moreover, our AdvGraph is naturally compatible to any GNNs without changing their architectures, which makes AdvGraph a general technique for enhancing the generalization performance of GNN-based recommenders. Extensive experiments demonstrate the effectiveness of the AdvGraph.

2 RELATED WORK

2.1 Graph Neural Recommendation

GNNs aim to learn node representations by iteratively aggregating neighbor messages which can capture structural information of graphs [18]. This message passing schema has been successfully applied to user-item interaction graphs [13, 18, 21, 27, 31]. Especially, Pinsage [31] integrates random walks and graph convolutions to compute node embeddings for recommendation at Pinterest. NGCF [27] designs an interaction layer to exploit the collaborative signals among multi-hop node community. LightGCN [13] further simplifies the aggregation modules by omitting feature transformations and nonlinear activations. Recently, UltraGCN [21] skips infinite layers of explicit messages passing for efficient recommendations. Though GNNs perform well, they are sensitive to graph perturbations, *e.g.,* adding or deleting edges [5, 35]. The vulnerability of GNNs may limit their deployments in industry.

2.2 Graph Adversarial Training

Adversarial training is a simple yet effective way to improve the robustness of neural networks [2, 14, 20, 23]. Nevertheless, adversarial training on graphs is non-trivial due to their discrete structures. Thus, many studies focus on perturbations on continuous spaces. For example, GraphAT [10], BVAT [6] and Flag [19] perform adversarial perturbations on nodes' features and boost system performance at test time. LAT-GCN [16] injects the perturbations on the first latent embeddings of GNNs, which blends the information of both node features and graphs. However, node features of bipartite graphs are often not available in recommender systems. On the other hand, several work have attempted to conduct graph perturbations using meta-learning [36], randomely dropping edges [5], and convex approximation with bisection method [29]. Nevertheless, these algorithms are often greedy, which may lead to sub-optimal performance. In contrast, we inject universal perturbations to discrete graphs, and directly solve a discrete optimization via Antithetic REINFORCE-based gradient estimators [7–9, 30].

3 METHODOLOGY

3.1 Preliminaries

3.1.1 ***Problem Setup.*** We first introduce the common paradigm of GNN-based recommenders [13, 27, 31]. Let $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ be the user-item interaction matrix, when $|\mathcal{U}|$ and $|\mathcal{I}|$ denote the number of users and items, respectively, and $\mathbf{R}_{ui} = 1$ if user u has interacted with item i before, 0 otherwise. Its corresponding adjacency matrix of the user-item graph can be obtained as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{bmatrix},$$

where the matrix $\mathbf{A}$ can be used for GNNs later. We aim to recommend users a ranked list of items that are unobserved before, in the same sense that performing link prediction on the bipartite graph.

3.1.2 ***Recap GNNs.*** The core of GNNs is to update the representation of each node by aggregating messages from its neighbors [18]. This message-passing schema can be expressed as:

$$\mathbf{H}^{(l)} = f_{\text{agg}}(\mathbf{H}^{(l-1)}, \mathbf{A}), \tag{1}$$

where $\mathbf{H}^{(l)}$ is the nodes' embeddings at the l-th layer, and $\mathbf{H}^{(0)}$ can be initialized with ID embeddings via lookup tables; $f_{\text{agg}}(\cdot)$ can be any differentiable aggregation functions [13, 18, 27, 31]. After L layers, one may adopt a readout function to generate the final embedding for each node v, *i.e.*, $\mathbf{h}_v = f_{\text{readout}}(\{\mathbf{h}_v^{(l)}, 0 \le l \le L\})$. The common choices of $f_{\text{readout}}(\cdot)$ include concatenation [27] and weighted sum [13]. Finally, we can use inner product to predict how likely user u would interact with item i as: $\hat{y}_{ui} = \mathbf{h}_u^\top \mathbf{h}_i$.

To optimize model parameters, we employ the pairwise Bayesian Personalized Ranking (BPR) loss [24] that encourages the prediction score of an observed entry to be higher than its unobserved counterparts:

$$\mathcal{L}_{BPR}(\mathbf{\Theta}) = \sum_{(u,i,j)\in\mathcal{D}} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \alpha\|\mathbf{\Theta}\|_F^2, \tag{2}$$

where $\mathcal{D} = \{(u,i,j)|\mathbf{R}_{ui} = 1, \mathbf{R}_{uj} = 0\}$ is the set of pairwise training instances; $\sigma(\cdot)$ is the sigmoid function; $\mathbf{\Theta}$ denotes the model parameters and α is the regularized parameter to prevent over-fitting.

3.2 The AdvGraph Model

3.2.1 ***Min-max Robust Optimization.*** GNNs are sensitive to small perturbations of graph structures [5, 17, 35, 36]. Inspired by adversarial training [20], we present AdvGraph, a general adversarial training framework that defends against perturbations on graph topology. To be specific, we inject perturbations to the original graph when training GNNs. As such, Eq. (1) becomes

$$\mathbf{H}^{(l)} = f_{\text{agg}}(\mathbf{H}^{(l-1)}, \mathbf{A} + \mathbf{\Delta}), \tag{3}$$

where $\mathbf{\Delta} = \{\Delta_{ij} \in \{0,1\} \wedge \Delta_{ij} \notin \mathbf{A}\}$ denotes the *universal* graph perturbations, where universal perturbations have been successfully used to improve both robustness and generalization performance in image classification [22, 25]. Moreover, the goal of adversarial perturbations is to cause largest influence (*e.g.*, worst-case perturbations). Therefore, we find the optimal adversarial perturbations $\mathbf{\Delta}_{adv}$ by maximizing the BPR loss:

$$\mathbf{\Delta}_{adv} = \arg\max_{\mathbf{\Delta}} \mathcal{L}_{\text{BPR}}(\hat{\mathbf{\Theta}}, \mathbf{A} + \mathbf{\Delta}), \quad \text{s.t.} \quad \|\mathbf{\Delta}\|_0 \le \delta, \tag{4}$$

where $\hat{\mathbf{\Theta}}$ is a constant denoting the current model parameters. In addition, the adversarial perturbations are expected to be small, unnoticeable and should not corrupt the majority of graph structures. We thus penalize the number of non-zero entries of $\mathbf{\Delta}$ via L_0 norm with an upper bound $\delta \ge 0$.

To this end, we aim to design a new objective function that is both suitable for personalized ranking and robust to adversarial graph perturbations. Formally, we minimize the adversarial BPR loss by combining Eq. (2) and Eq. (4) as:

$$\begin{aligned} \mathcal{L}(\mathbf{\Theta}) &= \mathcal{L}_{\text{BPR}}(\mathbf{\Theta}) + \beta \mathcal{L}_{\text{BPR}}(\mathbf{\Theta}, \mathbf{A} + \mathbf{\Delta}_{adv}), \\ \text{where} \quad \mathbf{\Delta}_{adv} &= \arg\max_{\mathbf{\Delta}, \|\mathbf{\Delta}\|_0 \le \delta} \mathcal{L}_{\text{BPR}}(\hat{\mathbf{\Theta}}, \mathbf{A} + \mathbf{\Delta}), \end{aligned} \tag{5}$$

where β controls the impact of the adversarial perturbations on the BPR loss. In the extreme case (e.g., $\beta = 0$), Eq. (5) boils down to the standard BPR loss in Eq. (2). Therefore, our AdvGraph can be viewed as a generalization of GNN models with adversarial robustness as a regularized term.

The objective Eq. (5) consists of two steps: 1) an inner maximization that computes the universal perturbations $\mathbf{\Delta}^*$ by fixing the model parameters $\mathbf{\Theta}$, and 2) an outer minimization that computes

the model parameters Θ^* by fixing the perturbations Δ. The outer minimization is a typical GNN optimization problem and we can learn Θ^* via Stochastic Gradient Descent. However, the inner problem is intractable due to discrete nature of Δ. We next introduce continuous relaxation of Δ via probabilistic reparameterization [15].

3.2.2 ***Antithetic Gradient Estimator.*** The Δ in Eq. (4) contains $2^{|\mathbf{A}|}$ possible candidates of edge perturbations, which becomes challenging for large-scale graphs. Inspired by recent work [26, 32], we introduce mask techniques to generate sub-graph perturbations that can be scalable to large graphs.

Given the bipartite graph $\mathbf{A}$, its complementary graph is $\bar{\mathbf{A}}$ that contains all unobserved edges, we can sample a sub-graph $\mathbf{T} \sim \mathcal{P}(\bar{\mathbf{A}})$ with a fixed budget of edges B. Then, we generate the universal perturbations as: $\Delta = \mathbf{T} \odot \mathbf{M}$, where $\odot$ denotes the element-wise product and $\mathbf{M} \in \{0, 1\}$ is a learnable binary mask (*e.g.,* 1 is kept and 0 is dropped). The inner optimization Eq. (4) can be approximated as:

$$\max_{\mathbf{M}} \quad \mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbf{M}) - \lambda \|\mathbf{M}\|_0, \tag{6}$$

where the sparsity of Δ (*e.g.,* $\|\Delta\|_0 \leq \delta$) can be achieved by adding L_0 regularizer on $\mathbf{M}$ with hyper-parameter $\lambda \geq 0$. The Eq. (6) is still intractable since the discrete variable $\mathbf{M}$ and the L_0 norm are non-differentiable. The most widely applicable approach is to compute the gradient via REINFORCE [5, 28], but with high variance. To reduce the variance, we propose an Antithetic REINFORCE-based gradient estimator to solve the inner optimization [7–9, 30].

We consider each $\mathbf{M}_{ij}$ to be drawn from a Bernoulli distribution parameterized by $\mathbf{\Pi}_{ij} \in [0, 1]$ such that $\mathbf{M}_{ij} \sim \text{Bern}(\mathbf{M}_{ij}; \mathbf{\Pi}_{ij})$ [26, 32]. Then, we can rewrite Eq. (6) by its expectation:

$$\max_{\mathbf{\Pi}} \quad \mathbb{E}_{\mathbf{M} \sim \text{Bernoulli}(\mathbf{M}; \mathbf{\Pi})} \left[\mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbf{M}) \right] - \lambda \sum_{i,j} \mathbf{\Pi}_{ij}. \tag{7}$$

To efficiently compute gradients of discrete variable $\mathbf{\Pi}$, we use the *reparameterization trick* [15] to reparameterize $\mathbf{\Pi}_{ij} \in [0, 1]$ with a deterministic function $g(\cdot)$ with parameter Φ_{ij}: $\mathbf{\Pi}_{ij} = g(\Phi_{ij})$. Since the $g(\cdot)$ should be bounded within $[0, 1]$, we simply choose the sigmoid function $\sigma(\cdot)$ as our deterministic function. Also, we have the following probability proprieties [33]:

(1) Given a Bernoulli random variable $z \sim \text{Bernoulli}(z; \sigma(\phi))$, it can be represented by comparing *two augmented exponential random variables*: $z = \mathbb{I}[xe^{-\phi} < y]$, $x, y \overset{i.i.d}{\sim} \text{Exp}(1)$, where $\mathbb{I}[\cdot]$ is the indicator function having the value 1 if condition is true and 0 otherwise.
(2) For a random variable $z = xe^{-\phi}, x \sim \text{Exp}(1)$, it is equal in distribution to $z \sim \text{Exp}(e^{\phi})$.
(3) For two variables $x, y \overset{i.i.d}{\sim} \text{Exp}(1)$, they are the same in distribution as $x = \epsilon u$, $y = \epsilon(1 - u)$, where $u \sim \text{Uniform}(0, 1)$, $\epsilon \sim \text{Gamma}(2, 1)$.

With proprieties (1-2), Eq. (7) is equivalent to

$$\begin{aligned} f(\Phi) =& \mathbb{E}_{\mathbf{M} \sim \text{Bernoulli}(\mathbf{M}; \sigma(\Phi))} \left[\mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbf{M}) \right] - \lambda \sum_{i,j} \sigma(\Phi_{ij}) \\ =& \mathbb{E}_{\mathbf{X}, \mathbf{Y} \overset{i.i.d}{\sim} \text{Exp}(\mathbf{1})} \left[\mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbb{I}[\mathbf{X} \odot e^{-\Phi} < \mathbf{Y}]) \right] - \lambda \sum_{i,j} \sigma(\Phi_{ij}) \\ =& \mathbb{E}_{\mathbf{Z} \sim \text{Exp}(e^{\Phi}), \mathbf{Y} \sim \text{Exp}(\mathbf{1})} \left[\mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbb{I}[\mathbf{Z} < \mathbf{Y}]) \right] - \lambda \sum_{i,j} \sigma(\Phi_{ij}). \end{aligned} \tag{8}$$

We next compute the gradients *w.r.t.* Φ. The second term is now differentiable and its gradients is: $c = \lambda \sum_{i,j} \nabla_{\Phi_{ij}} \sigma(\Phi_{ij})$. Applying REINFORCE to first term in Eq. (8), we have:

$$\begin{aligned} & \nabla_{\Phi} f(\Phi) \\ =& \mathbb{E}_{\mathbf{Z} \sim \text{Exp}(e^{\Phi}), \mathbf{Y} \sim \text{Exp}(\mathbf{1})} \left[\mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbb{I}[\mathbf{Z} < \mathbf{Y}]) \nabla_{\Phi} \log \text{Exp}(\mathbf{Z}; e^{\Phi}) \right] - c \\ =& \mathbb{E}_{\mathbf{Z} \sim \text{Exp}(e^{\Phi}), \mathbf{Y} \sim \text{Exp}(\mathbf{1})} \left[\mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbb{I}[\mathbf{Z} < \mathbf{Y}]) (1 - \mathbf{Z} \odot e^{\Phi}) \right] - c \\ =& \mathbb{E}_{\mathbf{X}, \mathbf{Y} \overset{i.i.d}{\sim} \text{Exp}(\mathbf{1})} \left[\mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbb{I}[\mathbf{X} \odot e^{-\Phi} < \mathbf{Y}]) (1 - \mathbf{X}) \right] - c. \end{aligned} \tag{9}$$

With proprieties (3), we can express $\mathbf{X}, \mathbf{Y} \overset{i.i.d}{\sim} \text{Exp}(\mathbf{1})$ as:

$$\mathbf{X} = \mathbf{E} \odot \mathbf{U}, \qquad \mathbf{Y} = \mathbf{E} \odot (1 - \mathbf{U}),$$

where $\mathbf{U}$ and $\mathbf{E}$ can be drawn from $\text{Uniform}(0, 1)$ and $\text{Gamma}(2, 1)$, respectively. The inequality in Eq. (9) becomes:

$$\mathbf{X} \odot e^{-\Phi} < \mathbf{Y} \Leftrightarrow \mathbf{U} < \sigma(\Phi).$$

As such, Eq. (9) can be expressed as:

$$\nabla_{\Phi} f(\Phi) = \mathbb{E}_{\substack{\mathbf{U} \sim \text{Uniform}(0,1), \\ \mathbf{E} \sim \text{Gamma}(2,1)}} \left[\mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbb{I}[\mathbf{U} < \sigma(\Phi)]) (1 - \mathbf{E} \odot \mathbf{U}) \right] - c.$$

Using Rao-Blackwellization [1], we further simplify gradients as:

$$\nabla_{\Phi} f(\Phi) = \mathbb{E}_{\mathbf{U} \sim \text{Uniform}(0,1)} \left[\mathcal{L}_{\text{BPR}}(\hat{\Theta}, \mathbf{A} + \mathbf{T} \odot \mathbb{I}[\mathbf{U} < \sigma(\Phi)]) (1 - 2\mathbf{U}) \right] - c. \tag{10}$$

As such, we can efficiently solve the discrete optimization Eq. (6) via probabilistic reparameterization. The gradient estimator shown above is also known as the Antithetic REINFORCE-based (AR) gradient estimator [7–9, 30], which has following benefits: 1) Applying antithetic sampling over augmented variables yields an unbiased and low-variance gradients. 2) It has low computational complexity: sampling from a Bernoulli distribution is simply replaced by a non-parametric Uniform distribution, and the gradients in Eq. (10) only involves one forward-pass of networks. These appealing properties make AR estimator popular in deep neural networks that contain non-differentiable operators, such as Bayesian GNNs [4, 12].

3.2.3 ***Complexity Analysis.*** To this end, we use an alternative optimization schema to iteratively update Δ and Θ, which is summarized in Algorithm 1. Also, it is worth mentioning that the proposed AdvGraph can be naturally plugged into any GNN-based recommenders without changing their architectures.

The time complexity of AdvGraph mainly comes from its min-max optimization. The outer optimization involves the standard training process of vanilla GNNs while the inner optimization only requires one forward-pass of GNNs. Therefore, the complexity of AdvGraph keeps the same order as the original GNNs.

4 EXPERIMENTS

4.1 Experimental Settings

Dataset. We conduct experiments on three benchmark datasets[1]: *Gowalla*, *Yelp-2018*, and *Amazon-Book*. The statistics of the datasets are summarized in Table 1. We follow the settings described in [13, 27] to split the datasets and adopt two common Top-N metrics [13, 27]: Recall@N and NDCG@N ($N = 20$ by default). Both evaluation metrics are computed by the all-ranking protocol [13].

[1] https://github.com/kuandeng/LightGCN/tree/master/Data

Algorithm 1: AdvGraph

Input: GNN f_{gnn}, bipartite graph $\mathbf{A}$, perturbations budget B, regularized parameters α, β for Eq. (5), and λ for Eq. (7)
Output: Model parameters Θ;

Initialize Θ by solving original BPR in Eq. (2);
Initialize Φ randomly;
Sampling a sub-graph $\mathbf{T} \sim \mathcal{P}(\bar{\mathbf{A}})$ with budget B;
repeat
  Draw $\mathbf{U} \sim \text{Uniform}(0, 1)$;
  Compute universal perturbations: $\mathbf{\Delta} = \mathbf{T} \odot \mathbb{I}[\mathbf{U} < \sigma(\Phi)]$;
  Feed $(\mathbf{A}, \mathbf{\Delta})$ to the GNN f_{gnn} and compute the loss in Eq. (5);
  Update Φ with stochastic gradient ascent via Eq. (10)
    $\Phi \leftarrow \Phi + \eta_1 \cdot \nabla_{\Phi} f(\Phi)$;
  Update Θ with stochastic gradient descent in Eq. (5)
    $\Theta \leftarrow \Theta - \eta_2 \cdot \nabla_{\Theta} \mathcal{L}(\Theta)$;
until *Convergence*
return Θ

Table 1: Statistics of the experimental datasets.

Dataset	|User|	|Items|	|Interactions|	Sparsity
Gowalla	29,858	40,981	1, 027, 370	0.084%
Yelp-2018	31, 668	38, 048	1, 561, 406	0.130%
Amazon-Book	52, 643	91, 599	2, 984, 108	0.062%

Compared Methods. We compare with the following models: 1) **BPR-MF** [24]: a matrix factorization model with Bayesian Personalized Ranking loss; 2) **APR-MF** [14]: a matrix factorization model with adversarial training; 3) **Pro-GNN** [17]: a robust GNN model with low-rank and sparse assumptions; 4) **LAT-GNN** [16] : an adversarial model that imposing perturbations on latent embeddings of GNNs. Note that our AdvGraph does not compete against GraphAT [10], BVAT [6] and Flag [19], because the node features of users/items are often not available in recommendations.

Parameter Settings. For Pro-GNN, LAT-GNN, and AdvGraph, we choose two recent models: NGCF [27] and LightGCN [13], as their GNN backbones. Also, we use the same hyper-parameters as the original backbones, such as batch size, stopping criteria, learning rate, *etc.* The hyper-parameters of all baselines are carefully tuned to achieve the optimal performance. For AdvGraph, we tune both adversarial regularizer β and L_0 regularizer λ (Eq. (6)) within $\{0.001, 0.01, 0.1, 1, 10\}$. For perturbations budget B, we vary the ratios r in $\{0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ such that $B = r \cdot |\mathbf{A}|$.

4.2 Experimental Results

4.2.1 Effect of Adversarial Learning. The results of different models in terms of Recall@20 and NDCG@20 are summarized in Table 2. Note that we simply omit the results of Pro-GNN for *Amazon-book* since it involves Singular Value Decomposition that is expensive for large graphs. We have the following observations:

- The recommender models, *e.g.*, both MF and GNN, generally get benefits from adversarial training, which demonstrates the effectiveness of adversarial perturbations during the training.
- AdvGraph yields the best performance on all datasets. For example, by comparing the LightGCN and LightGCN+AdvGraph, LightGCN+AdvGraph has on average 7.62% improvement with respect to Recall@20 and over 7.37% improvements in terms of NDCG@20. Figure 1 further shows the training curves of LightGCN and LightGCN+AdvGraph on the first two datasets.
- AdvGraph is able to generalize the LAT that only injects adversarial perturbations at the first layer of GNNs, leading to better performance. The reason is that injecting graph perturbations is equivalent to injecting perturbations to every layer of GNNs based on the message passing schema in Eq. (1).

In summary, the experimental results demonstrate the superiority of our proposed AdvGraph in the tasks of recommendations.

4.2.2 Hyper-parameter Studies. We further investigate sensitivity of AdvGraph for the hyper-parameters: the adversarial regularizer β, L_0 regularizer λ, and the ratio of edge perturbations r. We conduct experiments on *Gowalla*. Figure 2(a)-(b) show the results of β and λ. As can be seen, the non-zero choices of β and λ generally

Table 2: The performance of different methods.

	Gowalla		Yelp-2018		Amazon-Book	
Models	Recall	NDCG	Recall	NDCG	Recall	NDCG
BPR-MF [24]	0.1298	0.1112	0.0431	0.0352	0.0253	0.0197
APR-MF [14]	0.1416	0.1297	0.0467	0.0386	0.0281	0.0218
NGCF [27]	0.1563	0.1325	0.0580	0.0478	0.0337	0.0263
+Pro [17]	0.1598 (+2.12%)	0.1351 (+1.94%)	0.0592 (+2.13%)	0.0487 (+1.84%)	-	-
+LAT [16]	0.1612 (+3.13%)	0.1384 (+4.42%)	0.0605 (+4.31%)	0.0496 (+3.84%)	0.0351 (+4.03%)	0.0273 (+3.85%)
+AdvGraph	**0.1659** (+6.13%)	**0.1410** (+6.44%)	**0.0621** (+7.02%)	**0.0506** (+5.92%)	**0.0361** (+7.11%)	**0.0281** (+6.82%)
LightGCN [13]	0.1826	0.1549	0.0649	0.0540	0.0417	0.0322
+Pro [17]	0.1850 (+1.33%)	0.1574 (+1.64%)	0.0662 (+2.02%)	0.0551 (+1.93%)	-	-
+LAT [16]	0.1891 (+3.52%)	0.1593 (+2.81%)	0.0675 (+4.02%)	0.0559 (+3.47%)	0.0431 (+3.01%)	0.0335 (+4.06%)
+AdvGraph	**0.1976** (+8.21%)	**0.1659** (+7.07%)	**0.0697** (+7.41%)	**0.0577** (+6.92%)	**0.0447** (+7.24%)	**0.0348** (+8.13%)

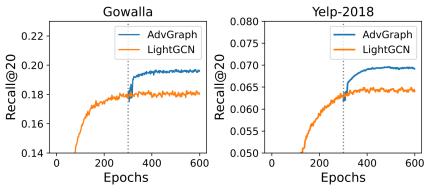

Figure 1: Performance curves of LightGCN and AdvGraph, where AdvGraph starts adversarial training after 300 epochs.

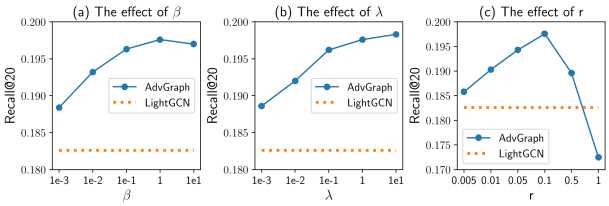

Figure 2: Sensitive analysis of our AdvGraph on Gowalla.

improve the performance. It is reasonable to set $\beta = \lambda = 1$ in our experiments. Figure 2(c) also shows the impact of the noisy level of edge perturbations . We observe that the performance increases with a larger ratio when $r \leq 0.1$, but the performance drops dramatically when r is too large. This indicates that injecting too many edge perturbations will impair the model parameters of GNNs.

5 CONCLUSION

GNN-based recommenders can be easily fooled by graph perturbations. To address this issue, we present a general framework AdvGraph, to model adversarial graph perturbations during the training of GNNs. The idea of AdvGraph is to compute graph universal perturbations and model parameters by solving a min-max optimization. Extensive experiments demonstrate the positive impact of our graph adversarial learning for personalized ranking.

REFERENCES

[1] George Casella and Christian P Robert. 1996. Rao-Blackwellisation of sampling schemes. *Biometrika* (1996), 81–94.

[2] Huiyuan Chen and Jing Li. 2019. Adversarial tensor factorization for context-aware recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 363–367.

[3] Huiyuan Chen and Jing Li. 2019. Data poisoning attacks on cross-domain recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2177–2180.

[4] Huiyuan Chen, Lan Wang, Yusan Lin, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 2021. Structured graph convolutional networks with stochastic masks for recommender systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 614–623.

[5] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. 1115–1124.

[6] Zhijie Deng, Yinpeng Dong, and Jun Zhu. 2019. Batch virtual adversarial training for graph convolutional networks. *arXiv preprint arXiv:1902.09192* (2019).

[7] Alek Dimitriev and Mingyuan Zhou. 2021. ARMS: Antithetic-REINFORCE-Multi-Sample Gradient for Binary Variables. In *International Conference on Machine Learning*. 2717–2727.

[8] Zhe Dong, Andriy Mnih, and George Tucker. 2020. DisARM: An Antithetic Gradient Estimator for Binary Latent Variables. In *Advances in Neural Information Processing Systems*.

[9] Zhe Dong, Andriy Mnih, and George Tucker. 2021. Coupled Gradient Estimators for Discrete Latent Variables. In *Advances in Neural Information Processing Systems*.

[10] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. 2019. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[11] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. 2021. Robustness of Graph Neural Networks at Scale. *Advances in Neural Information Processing Systems* (2021).

[12] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. 2020. Bayesian graph neural networks with adaptive connection sampling. In *International conference on machine learning*.

[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[14] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 355–364.

[15] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

[16] Hongwei Jin and Xinhua Zhang. 2019. Latent adversarial training of graph convolution networks. In *ICML Workshop on Learning and Reasoning with Graph-Structured Representations*.

[17] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 66–74.

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[19] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. 2020. Flag: Adversarial data augmentation for graph neural networks. *arXiv preprint arXiv:2010.09891* (2020).

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

[21] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1253–1262.

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1765–1773.

[23] Liang Qu, Huaisheng Zhu, Ruiqi Zheng, Yuhui Shi, and Hongzhi Yin. 2021. ImGAGN: Imbalanced Network Embedding via Generative Adversarial Graph Networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery amp; Data Mining*. 1390–1398.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.

[25] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. 2020. Universal adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 5636–5643.

[26] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. 2021. Adversarial Graph Augmentation to Improve Graph Contrastive Learning. In *Advances in Neural Information Processing Systems*.

[27] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[28] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.

[29] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 3961–3967.

[30] Mingzhang Yin and Mingyuan Zhou. 2019. ARM: Augment-REINFORCE-Merge Gradient for Stochastic Binary Networks. In *International Conference on Learning Representations*.

[31] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[32] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. In *Advances in Neural Information Processing Systems*.

[33] Mingyuan Zhou and Lawrence Carin. 2013. Negative binomial process count and mixture modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013), 307–320.

[34] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1399–1407.

[35] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.

[36] Daniel Zügner and Stephan Günnemann. 2018. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations*.