Interpretable and Effective Reinforcement Learning for Attacking against Graph-based Rumor Detection

Yuefei Lyu*, Xiaoyu Yang*, Jiaxin Liu†, Sihong Xie†, Philip Yu‡ and Xi Zhang*§
*Key Laboratory of Trustworthy Distributed Computing and Service (BUPT)
Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, China
†Lehigh University, Bethlehem, USA
‡University of Illinois at Chicago, Chicago, USA
lvyuefei@bupt.edu.cn, littlehaes@bupt.cn, jilb17@lehigh.edu, sxie@cse.lehigh.edu, psyu@uic.edu, zhangx@bupt.edu.cn
§Corresponding Author

***Abstract*—Social networks are frequently polluted by rumors, which can be detected by advanced models such as graph neural networks. However, the models are vulnerable to attacks, and discovering and understanding the vulnerabilities is critical to robust rumor detection. To discover subtle vulnerabilities, we design a attacking algorithm based on reinforcement learning to camouflage rumors against black-box detectors. We address exponentially large state spaces, high-order graph dependencies, and ranking dependencies, which are unique to the problem setting but fundamentally challenging for the state-of-the-art end-to-end approaches. We design domain-specific features that have causal effect on the reward, so that even a linear policy can arrive at powerful attacks with additional interpretability. To speed up policy optimization, we devise: (i) a credit assignment method that proportionally decomposes delayed and aggregated rewards to atomic attacking actions for enhance feature-reward associations; (ii) a time-dependent control variate to reduce prediction variance due to large state-action spaces and long attack horizon, based on reward variance analysis and a Bayesian analysis of the prediction distribution. On two real world datasets of rumor detection tasks, we demonstrate: (i) the effectiveness of the learned attacking policy on a wide spectrum of target models compared to both rule-based and end-to-end attacking approaches; (ii) the usefulness of the proposed credit assignment strategy and variance reduction components; (iii) the interpretability of the attacking policy.**

***Index Terms*—graph adversarial attack, reinforcement learning, graph convolutional network, rumor detection**

I. Introduction

Social networks, such as Twitter and Weibo, help propagate useful information. However, they are also exploited to spread misinformation, such as rumors, to manipulate opinions in a large scale. Detecting misinformation is important to trustworthy social networks. Graph Convolutional Networks (GCN) [1] can aggregate neighborhood information to deliver high detection accuracy [2]–[5]. However, GCN is fragile to graph adversarial attacks. For example, [6] showed that GCN are vulnerable to edges or features flipping that degrade node classification accuracy. Focusing on rumor detection, [2] restricts rumor producers to controlled accounts to camouflage rumors to be less suspicious through re-posting, following and commenting, to deceive a GCN detector.

To re-design more robust GCN-based rumor detectors, it is critical to discover and understand the vulnerabilities using an attacking model to simulate camouflage actions of various level of sophistication under realistic constraints. However, existing attacking models, especially those based on deep neural networks and reinforcement learning, are too complicated to help humans understand how attacks are generated and what detector vulnerabilities are exploited. We argue that the simplicity of the attacking models should be as important as their effectiveness to be useful to the detector designers.

Threat model. To capture how rumors can be spread, we use a heterogeneous graph consisting of nodes representing user accounts, messages, and comments, and edges representing posting, re-posting, and commenting activities. An attacker produces adversarial samples by controlling some accounts to post messages and follow other accounts [7]. These operations can be represented as adding edges to the graph. For each message node v_i, a trained GCN f outputs $f(v_i)$ as the ranking of v_i based on its suspiciousness, and messages with high ranking will be removed as rumors. During a time period, only a few high influence messages can receive the most attention [8]. A high influence rumor is more useful for spreading misinformation, worth to be camouflaged, as it has zero influence once detected. Inspired by the ranking metric Normalized Discounted Cumulative Gain (NDCG), we assume that an attacker aims to minimize the objective function:

$$J = \frac{1}{Z}\sum_{i=1}^{n} \frac{w_i}{\log(f(v_i)+1)} \mathbb{1}[f(v_i) > \kappa], \tag{1}$$

where Z normalizes the sum to $[0, 1]$ and n is the number of target rumors (not including other rumors that are not controlled by this attacker). w_i is the influence or weight of the i-th target rumor v_i estimated in various ways [9]. The indicator function $\mathbb{1}[f(v_i) > \kappa]$ truncates the contribution of target rumors whose suspiciousness ranking is greater than κ. Prior graph adversarial attacks assumed the target detectors are white-box [10]–[12] so that gradient-based attacks can be crafted. In contrast, we assume that the architecture or model parameters of f is unknown and only allow attackers having access to and manipulate part of the nodes in a social network.

Challenges. Reinforcement learning (RL) has been adopted to learn from a sequence of attack actions and the black-box detector's output as rewards. Prior RL-based attacks [6]

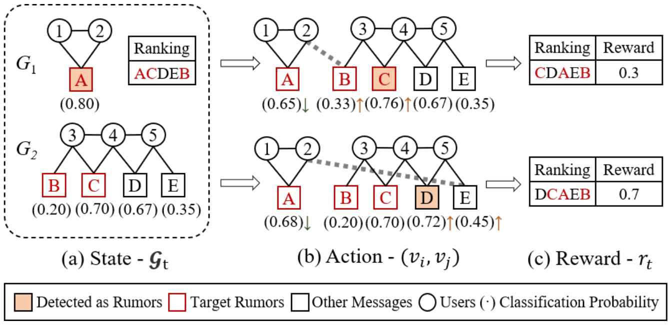


Fig. 1: The state, action and reward in RL step t. The state and action vectors describe the current graph $\mathcal{G}_t$ and the edge (v_i, v_j) to add for camouflaging, respectively. It produces the reward r_t according to the suspiciousness ranking of target rumors A, B and C according to (1). **Sensitive state-action reward mapping**: two similar actions (v_2, v_B) and (v_2, v_E) under the same state in the dashed box are structurally symmetric but lead to different rewards after more attack steps. We obverse two dependencies: i) **Graph dependencies**. When connecting G_1 and G_2 with (v_2, v_B), v_A in G_1 becomes less suspicious and v_B and v_C in G_2 become more suspicious due to information propagation. The more remote v_D and v_E is not affected. The up/down arrows in the right of classification probability means probability increasing/decreasing. ii) **Ranking dependencies** occur due to relative suspiciousness ranking of target rumors. Pulling a target rumor down the suspicious list might push another target rumor up into the top of the list. Assuming the top-1 message is detected as the rumor, *top*: after an attack, rumor A escapes the detection while rumor C is detected. *Bottom*: after a better attack, rumors A and C both escape the detection.

train neural networks as policies in an end-to-end manner. The neural RL policies are not interpretable [13]. Furthermore, the attack performance depends heavily on whether the graph representation can capture long-term reward, and the typical GCN is limited in expressiveness [14]–[16] and has difficulties in learning a representation of global inter-node influence and ranking that contribute to aggregated reward of camouflaging. For example, in Fig. 1, the dependencies among the messages on the graph and their relative suspiciousness ranking positions in objective (1) are exploitable vulnerabilities that are hard for the end-to-end approaches to learn from delayed feedbacks. Lastly, RL policies for manipulating graphs is highly sample inefficient: structural similar actions under the same state can lead to significantly different returns as shown in Fig. 1. With the large action spaces derived from graphs and long horizon, there are exponentially many possible future trajectories, each of which can lead very different rewards. Thus, the current state-action representation can hardly predict the final reward [17], [18], and the common state/action dependent reward baseline [19]–[22] work with poor effects to reduce the prediction variance.

Proposed solution. We propose AdRumor-RL to generate interpretable and effective evasion attacks to camouflage high influence rumors from a GCN rumor detector. We formulate an episodic Markov Decision Process (MDP) and a hierarchical RL algorithm to attack the detector. An action adds an edge in two sub-steps: the agent first selects two graphs (possibly identical) and then two nodes from the selected graphs to connect. A return (cumulative rewards) at the end of an episode represents how well the sequence of added edges reduces the objective (1). We have two inventions:

First, to learn a strong and interpretable attacking policy, we use domain knowledge about rumor spread to design inherent but hard-to-learn features to train a linear policy for more interpretability and data efficiency. In particular, we include two kinds of features that capture the dependencies due to the graphs and the ranking-based objective. As shown in Fig. 1, the graph makes rumors depend on each other so that linking v_2 and v_B can make the connected target rumor v_C more detectable and thus reduce attack effectiveness. The relative suspiciousness ranking of rumors creates another type of dependencies: as shown in the top of Fig. 1, the ranking drop of target rumor A leads to the ranking rising of target rumor C, making camouflaging less effective. Both types of dependencies are hard for a data-driven learning agent to capture and we design features to capture such dependencies to train powerful and yet interpretable linear policies.

Second, we propose a time-dependent credit assignment and reward baseline to address the large variance in delayed reward due to the large action space/horizon and sensitive graph state-action representation. We decompose the returns achieved by a sequence of edges manipulations and proportionally assign the due effects to individual actions to speed up learning. The step-wise rewards exhibit a dependency on the time steps, providing a time-dependent baseline to reduce the high variance of the reward. Intuitively, it "clusters" the rewards based on the time steps and find the cluster means achieve to a smaller prediction variance, based on a reward variance analysis and a Bayesian analysis of prediction distribution.

II. Related Work

Rumor Detection with GCNs. Many rumor detection methods make use of GCNs to mine social relationship networks. CGAT [2] constructs a user-tweet-comment graph and proposes a graph adversarial learning framework, which help GCN better learn malicious rumor camouflage behaviors. FANG [3] enhances the node representation ability of GraphSAGE [23] based on user-news-source graphs. GCNs are also used to detect rumors with message propagation trees [4] and potential user interaction graphs [24], and here we attack against GCN detector with heterogeneous social networks.

Adversarial Attack on GCNs. Adversarial attack can be categorized based on poisoning [25] and evasion attacks [12]; untargeted [11] and targeted attacks [6]; white-box and black-box attacks [6], [7]. We realize evasion attacks against a black-box pre-trained rumor detector, targeting at a set of rumors.

Reward baseline of reinforcement learning. The reward baseline (control variate) can reduce variance of Monte Carlo estimation effectively [26], [27]. The basic method is to use the constant baseline, such as the difference between the reward and the average reward [28], [29]. A common method is to use state-value functions as state-dependent control variates [19],

[20]. Recently, some action-dependent [21], [22] and input-based [30] methods are proposed. In this work, we propose a time-dependent control variate as reward baseline.

III. Preliminaries and Problem Definition

A. Rumor Detection on Social Networks

We construct an undirected heterogeneous graph $\mathcal{G}$=$(\mathcal{V}, \mathcal{E})$. The node set is $\mathcal{V} = \{v \mid v \in \mathcal{M} \cup \mathcal{U} \cup \mathcal{C}\}$, where $\mathcal{M}$, $\mathcal{U}$ and $\mathcal{C}$ are the sets of messages, users, and comments, respectively. The edge set is $\mathcal{E} = \{(v_i, v_j) \mid v_i \in \mathcal{V}, v_j \in \mathcal{V}\}$. There is a relation mapping function $\psi : \mathcal{E} \to \mathcal{L}$ and $\mathcal{L} = \{l_1, l_2, l_3\}$ is the set of three particular relation types: user-message l_1, message-comment l_2, and user-user l_3. l_1 indicates a user posts or re-posts a message, l_2 means that a comment is appended to a message, and l_3 means that a user is connected to the author of a message when the user re-posts the message, or when two users re-post the same message. There are many communities, which correspond to a set of connected components $\{G_1, G_2, ..., G_m\}$ in $\mathcal{G}$, where each connected component $G_i = (\mathcal{V}_i, \mathcal{E}_i)$ is called a *subgraph* in the sequel.

We attack against the GCN-based rumor detectors f that is a trained model, such as GCN [1], GAT [31], GraphSAGE [23] and RGCN [32]. The model output probabilities are used to rank the messages, with ranking position $f(v_i) \in \{1, 2, \ldots, M\}$, where M is the number of messages.

B. Influence Calculation

We calculate user influence using PageRank on $\mathcal{G}_{user}$ containing user nodes and user-user relations

$$w_j = \text{PageRank}(\mathcal{G}_{user}, v_j), v_j \in \mathcal{U}, \tag{2}$$

and the influence of the message $v_i \in \mathcal{M}$ is calculated as

$$w_i = \max_{u \in \mathcal{N}_i^1} \text{PageRank}(\mathcal{G}_{user}, u) + \frac{\mid \mathcal{N}_i^1 \mid -1}{z_1} + \frac{\mid \mathcal{N}_i^2 \mid}{z_2}, \tag{3}$$

where $\mathcal{N}_i^1$ and $\mathcal{N}_i^2$ are the user and comment neighbors of v_i. z_1 and z_2 indicate the normalization factors. Multiple reposting and comments, and linking with high influence users make the message influential.

C. Reinforcement Learning

An MDP consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a reward function $r(S_t, A_t)$, a state transition probability distribution $\Pr(S_{t+1}|S_t, A_t)$. Since we focus on finite horizon, a discounting factor is not needed. In our application, at any time t, a state S_t is the graph $\mathcal{G}_t$ and an action A_t is a pair of nodes (v_i, v_j) or a pair of subgraphs. The goal of the attacker is to train an attacking policy $\pi_{\boldsymbol{\theta}}(A|S)$ that connects nodes in T steps to minimize (1). Since $f(v_i)$ is a black-box, (1) cannot be minimized via gradient-based approaches. The trajectory is denoted by $(S_0, A_0, S_1, \ldots, A_{T-1}, S_T)$. From samples of T-step trajectories by interacting with the environment, RL uses the reduction in the objective as a reward to learn a policy π that maximizes the reduction

$$\Delta NDCG = J(0) - J(T) \tag{4}$$

where $J(T)$ is the NDCG value (1) at the end of step T and $J(0)$ is the NDCG value before attack. We will learn the action-value function $Q_\pi(S_t, A_t)$ so that

$$Q^*(s, a) = \max_\pi \mathbb{E}[R_T | S_t = s, A_t = a], \tag{5}$$

where R_T is the random variable representing the reduction in (1) at the end of the T steps, starting from the t-th step with state s and action a. If $t < T$, R_T is a delayed reward. For interpretability, we let the Q function be linear: $Q(s, a|\boldsymbol{\theta}) = \mathbf{x}(s, a)^\top \boldsymbol{\theta}$, with $\mathbf{x}(s, a)$ being a vector representing (s, a) pair. We sample triples (s, a, r) to train Q by solving

$$\min_{\boldsymbol{\theta}} \sum_{(s,a,r) \sim \mathcal{D}} (Q(s, a|\boldsymbol{\theta}) - r)^2. \tag{6}$$

The loss function is similar to that in DQN [33], but we use the Monte Carlo estimation of the reward rather than bootstrapping with a Q function. Following the LinUCB algorithm [34], in the end of each episode e, we update the policy with state-action vectors and rewards of T steps as

$$\begin{aligned} \mathbf{A}_e &= \mathbf{A}_{e-1} + \sum\nolimits_{t=1}^{T} \mathbf{x}_t \mathbf{x}_t^\top \\ \boldsymbol{b}_e &= \boldsymbol{b}_{e-1} + \sum\nolimits_{t=1}^{T} r_t \mathbf{x}_t \\ \boldsymbol{\theta}_e &= \mathbf{A}_e{}^{-1} \mathbf{b}_e, \end{aligned} \tag{7}$$

with initial values $\mathbf{A}_0 = \mathbf{I}_d$ and $\mathbf{b}_0 = \mathbf{0}_{d*1}$, where d is the feature dimension. At each step t of episode e, the policy $\pi_{\boldsymbol{\theta}}$ chooses the action a_t from action space $\mathcal{A}(t)$ at time t as

$$a_t = \arg\max_{a \in \mathcal{A}(t)} \mathbf{x}(s_t, a)^\top \boldsymbol{\theta}_{e-1} + \alpha \sqrt{\mathbf{x}(s_t, a)^\top \mathbf{A}_{e-1}^{-1} \mathbf{x}(s_t, a)}, \tag{8}$$

where α is a hyper-parameter to control the exploitation and exploration trade-off.

IV. Method

A. The Attack Framework

The attacker can only have access to controllable node and add edges of specified type. The controllable node set $\mathcal{V}' \subset \mathcal{V}$ contains controllable users and their messages. The modifiable edge set is $\mathcal{E}' = \{(v_i, v_j) \mid v_i \in \mathcal{V}' \cap \mathcal{U}, v_j \in \mathcal{V}' \cap \mathcal{M}, (v_i, v_j) \notin \mathcal{E}\}$, which allows to connect controllable users with controllable messages. The target rumor set $\mathcal{O}$ is the set of all rumors in controllable node set $\mathcal{V}'$.

Due to the multi-communities characteristic of social networks, we design a hierarchical contextual bandit to decompose the action to the subgraph and node levels. The decomposition reduces the action spaces to speed up reinforcement learning. The AdRumor-RL algorithm is shown in Fig. 2.

On the subgraph level, we focus on the subgraphs $\{G_i = (\mathcal{V}_i, \mathcal{E}_i)\}$ and extract their features with feature extraction method Φ^g. The feature vector of G_i is denoted by $\Phi^g(G_i) = \boldsymbol{h}_i^g = [h_{i,1}^g, h_{i,2}^g, ..., h_{i,d}^g]$, d is the subgraph feature dimension. Two subgraphs are combined as an action, with the action space $\mathcal{A}_1 = \{(G_i, G_j) \mid \mathcal{O} \cap \mathcal{V}_i \neq \emptyset, \mathcal{V}' \cap \mathcal{V}_j \neq \emptyset\}$, which means subgraph G_i and G_j must contain target rumors and

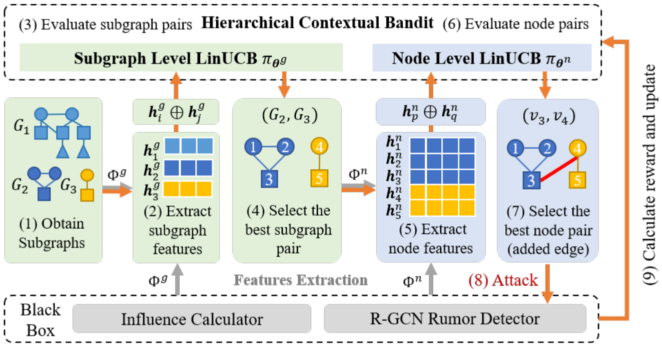

(3) Evaluate subgraph pairs
Hierarchical Contextual Bandit
(6) Evaluate node pairs
Subgraph Level LinUCB $\pi_{\boldsymbol{\theta}^g}$
Node Level LinUCB $\pi_{\boldsymbol{\theta}^n}$
G_1
G_2
G_3
(1) Obtain Subgraphs
Φ^g
$\boldsymbol{h}_i^g \oplus \boldsymbol{h}_j^g$
$\boldsymbol{h}_1^g$
$\boldsymbol{h}_2^g$
$\boldsymbol{h}_3^g$
(2) Extract subgraph features
(G_2, G_3)
1
2
3
4
5
(4) Select the best subgraph pair
Φ^n
$\boldsymbol{h}_p^n \oplus \boldsymbol{h}_q^n$
$\boldsymbol{h}_1^n$
$\boldsymbol{h}_2^n$
$\boldsymbol{h}_3^n$
$\boldsymbol{h}_4^n$
$\boldsymbol{h}_5^n$
(5) Extract node features
(v_3, v_4)
(7) Select the best node pair (added edge)
Φ^g
Features Extraction
Φ^n
(8) Attack
(9) Calculate reward and update
Black Box
Influence Calculator
R-GCN Rumor Detector

Fig. 2: AdRumor-RL. Following the orange arrows, there are 9 phases. It first extracts and concatenates subgraph features. and then the subgraph-level policy selects the best subgraph pair using (8). Phases 5-7 select and connect a pair of nodes from the selected subgraphs. The selected edge is added and the reward of the attack is calculated based on (14), which is used to update parameters with (7) after T steps attack. The grey arrows indicates the feature extraction process.

controllable nodes respectively, to attack the target rumor using a controllable node. Features of G_i and G_j are concatenated to $\mathbf{x}^g_{(i,j)} = \boldsymbol{h}^g_i \oplus \boldsymbol{h}^g_j = [h^g_{i,1}, ..., h^g_{i,d}, h^g_{j,1}, ..., h^g_{j,d}]$ based on which the subgraph-level policy $\pi_{\boldsymbol{\theta}^g}$ selects the best subgraph pair as shown in (8).

On the node level, for each node in the selected subgraph pair (G_i, G_j), we extract node features $\boldsymbol{h}^n$ with Φ^n. Any two nodes from the two selected subgraphs are paired to construct an action space $\mathcal{A}_2 = \{(v_p, v_q) \mid v_p \in \mathcal{V}_i, v_q \in \mathcal{V}_j, (v_p, v_q) \in \mathcal{E}'\}$. Similar to the subgraph level, the node-level policy $\pi_{\boldsymbol{\theta}^n}$ evaluates all node pairs with concatenated features $\mathbf{x}^n_{(p,q)}$ and decides the edge to be added for attack. After T edges are added in an episode, we calculate the reward with (14) and update $\boldsymbol{\theta}^g$ and $\boldsymbol{\theta}^n$ with (7).

B. Interpretable Attacking Feature

There is a trade-off between feature design, interpretability, and sample efficiency. With features learned end-to-end, there is less design effort but poor interpretability and sample efficiency (more data needed to learn the feature representation beyond the attacking policy). On the other hand, with hand-crafted features, simpler model can be used to enhance interpretability and allow less samples to learn a powerful policy. We design interpretable features to capture mechanism of rumor detection. With a linear model $\boldsymbol{\theta}^\top \mathbf{x}$, each element of $\boldsymbol{\theta}$ represents the importance of the corresponding feature for predicting future attacking rewards. The feature importance helps the detector designers to understand the attacking policies and detector vulnerabilities. In particular, these features describe the social network on the subgraph and node level, and include structural, social, influence, attack potential and camouflaging features. Structural features describe characteristics of graph and node, such as the number of nodes and edges, degree etc. Social features describe node type (rumor, non-rumor, user and comment) and ratios for different types of nodes. Influence features summarize user and message influence calculated in (2) and (3). We detail the attack potential and camouflaging features that respectively describe graph and ranking dependencies, as exemplified in Fig. 1.

Ranking dependencies means that two rumors' ranking positions are related as they appear in the same ranking list. The drop in ranking of one message can lead to the rise in ranking of other messages. For example, at the top of Fig. 1, attacking the target rumor A can lower its ranking, but also rises the ranking of another target rumor C. An effective attack must not rise the ranking of other targets when pushing down a target rumor. Therefore, the non-target messages are expected to exchange ranking positions with the target rumors to camouflage the targets (such as message D in Fig. 1), because they don't affect the attack objective function in (1). These non-target messages to help camouflage rumors are named camouflaging messages. We use the classification probability of the camouflaging messages in the selected subgraph or around the selected node to capture ranking dependencies. It is more likely for a camouflaging message to exchange with a target rumor if their probabilities are similar. The feature can help attackers identify whether a camouflaging message has a chance to exchange ranking positions with target rumors to help camouflage them, named camouflaging feature.

Graph dependencies occur due to information propagations along the links in a graph. For example, when connecting the edge (v_2, v_B) in Fig. 1, v_A propagates the suspiciousness to v_B, which makes v_A less suspicious and v_B more suspicious. v_A increases the attack performance and v_B does the opposite. In addition, when v_B is attacked directly, v_C is also made to be suspicious indirectly. Therefore, we design the attack potential features to measure the effects when a target rumor is attacked. i) Suspiciousness. We query the classification probability of target rumors in the selected subgraph or around the selected node before attack. Attacking suspicious rumors could change the NDCG more due to small $f(v_i)$ in (1). ii) Attack degree. We record the number of previous added edges within the subgraph and node neighbor. The object that has been attacked repeatedly will have less potential for changing the objective. iii) The condition of nearby target rumors. It concerns the number of targets within the selected node k-hop insides and their averaged distance to the selected node. It might have greater effects when attacking the target connected to more other targets in a shorter distance. Furthermore, camouflaging messages also play a role in graph dependencies because connecting the camouflaging messages with low probability to the target rumor could make the target less suspicious.

C. Credit Assignment

To learn to minimize the objective function in (1), prior work [35]–[37] shows that it is important to assign a proper reward as the feedback signal to individual action or state-action tuple that deserves the reward. Otherwise, the policy will be trained to visit undesirable states or state-action tuples more frequently since the policy is unable to distinguish high and low-valued actions. However, as shown at the top of Fig. 3, similar state-action vectors can lead to different rewards. It

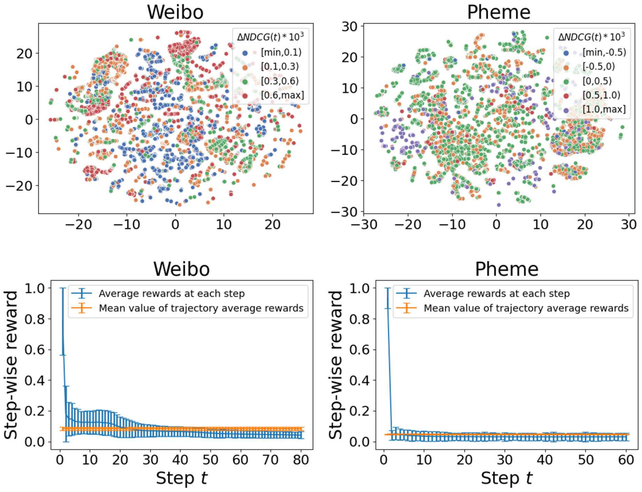


Fig. 3: The motivation of credit assignment and baseline design. *Top*: The scatter plots of 2-D state-action vector processed by tSNE in Weibo($T = 80$) and Pheme($T = 60$) datasets. Scatters are colored according to the assigned credit $\Delta NDCG(t)$ as (9) associated with each state-action pair. Observe that vector representation of state-action cannot predict the credit well. *Bottom*: The average step-wise rewards at each step t as (13) and the mean of average rewards of each trajectory ($\mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}}[\frac{1}{T}\sum_t r_t]$). The former decreases as more edges are added. Vertical lines denote standard deviations. The time-dependent baseline can better predict step-wise rewards and leads to more variance reduction.

means that the prior state-dependent or state-action-dependent assigned rewards are not indicative of step action effects.

Thus, we propose a time-dependent credit assignment method. For a trajectory $(S_0, A_0, S_1, \dots, A_{T-1}, S_T)$, we define the following step-wise NDCG change:

$$\Delta NDCG(t) = J(t-1) - J(t), t = 1, 2, ...T. \quad (9)$$

We assign $\Delta NDCG(t)$ to each step t as

$$r_t = r(S_{t-1}, A_{t-1}) = o(\Delta NDCG(t)), t = 1, 2, ...T, \quad (10)$$

where o is Min-Max normalization function and the maximum and minimum value can be estimated by rule-based method described in section V-C.

Compared with representing states of multiple graphs, the time has a simple representation, and the assigned credits are highly correlated with the time steps. At the bottom of Fig. 3, we observe a significant difference in the step-wise rewards, especially between early and late steps. Alternatively, if we assign the delayed return R_T as a single reward to each (S_t, A_t) of a trajectory for updating the policy as in (7), the values of late/early attacking steps are overestimated/underestimated, so that the agent won't learn to take high-value actions early on to maximize the overall return.

D. Variance Reduction in Rewards

Reward baseline. We estimate the optimal $\boldsymbol{\theta}$ with Monte Carlo techniques and update the policy in (7), and this estimation tends to have a high variance [26], [27]. To reduce the variance, a common practice is to subtract a baseline, or control variate, that highly correlates with the rewards [21], [22], [27], [28]. The reward at step t becomes

$$\tilde{r}_t = r_t - b(S_{t-1}, A_{t-1}), \quad (11)$$

where $b(S_{t-1}, A_{t-1})$ is the baseline that can depend on S_{t-1} [20], A_{t-1} [22], or some external input process [30]. With baselines to reduce policy optimization variance, the policy converges to the excellent level quickly, and then predicts the Q value $\mathbf{x}^\top \boldsymbol{\theta}$ more accurately to choose an effective action in (8). In other words, the baseline could help reduce the prediction variance. A simple baseline can be the average reward from a trajectory that is a constant:

$$\bar{b} = \frac{1}{T}\sum_{t=1}^{T} r_t. \quad (12)$$

The above constant baseline is not too much correlated with r_t and thus too coarse to control the variance, as shown in the bottom of Fig. 3. As for state-dependent or action-dependent baselines, they learn a function of state or action vector to correlate with r_t better. However, it is difficult to distinguish the rewards of similar state-action vectors in graph data with the long horizon as shown in the top of Fig. 3, and the mapping from state or action vector to the reward is sensitive, which makes such baselines estimation accompanied by risks of learning failure.

A time-dependent control variate. To address these difficulties, we propose the following time-dependent baseline

$$V^\pi(t) = \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}}[r_t], \quad (13)$$

which is the expectation of rewards r_t collected at step t across each trajectory τ when executing the policy $\pi_{\boldsymbol{\theta}}$. It is an unbiased estimation with simple time representation and avoids complex learning. Fig. 3 (bottom) plots $V^\pi(t)$. Compared with the constant baseline $\bar{b}$ in (12), the time-dependent baseline $V^\pi(t)$ can predict r_t more accurately over time. As a result, we use $V^\pi(t)$ as a control variate to reduce the prediction variance, leading to the following reward:

$$\tilde{r}_t = r_t - V^\pi(t). \quad (14)$$

$\tilde{r}_t$ is used to replace r_t to update policy in (7).

Reward sample variance analysis. $V^\pi(t)$ is defined for each t across trajectories, rather than depending on rewards from other steps in the same trajectory. The implicit assumption is that the rewards from different time steps are conditional independent. Thus, we could analyze the reward variance with sample independence. The time-dependent baseline could be seen as "clustering" the rewards according to the time t. It reduces the differences among reward clusters through subtracting the cluster center to shift the clusters to the close positions, and then reduces the reward variance.

Theorem 1. *Given the sample matrix $\boldsymbol{R} = \{r_{e,t}\} \in \mathbb{R}^{E\times T}$, where $r_{e,t}$ indicates the element in the e-th row and the t-th column, and $b_t = \mathbb{E}_e[r_{e,t}] = \frac{1}{E}\sum_e r_{e,t}$ is the mean of the t-th* *column of* $\boldsymbol{R}$*. The variance of random variate is denoted by* $\mathrm{Var}(\cdot)$*. Then,* $\mathrm{Var}(r_{e,t}) \geq \mathrm{Var}(r_{e,t} - b_t)$*.*

Proof.

$$\begin{aligned}
&\mathrm{Var}(r_{e,t}) - \mathrm{Var}(r_{e,t} - b_t) \\
&= \frac{1}{ET}\sum\nolimits_e\sum\nolimits_t\left(r_{e,t} - \frac{1}{ET}\sum\nolimits_e\sum\nolimits_t r_{e,t}\right)^2 \\
&\quad - \frac{1}{ET}\sum\nolimits_e\sum\nolimits_t\left(r_{e,t} - b_t - \frac{1}{ET}\sum\nolimits_e\sum\nolimits_t (r_{e,t} - b_t)\right)^2 \\
&= \frac{1}{T}\sum\nolimits_t\left(b_t - \frac{1}{T}\sum\nolimits_t b_t\right)^2 \geq 0.
\end{aligned}$$

Thus, $\mathrm{Var}(r_{e,t}) \geq \mathrm{Var}(r_{e,t} - b_t)$. □

We collect r_t over each trajectory and step to form a matrix, each row and column of which corresponds to a trajectory and a step respectively. Then, $V^\pi(t)$ is the mean of the t-th column of this matrix. As $\bar{b}$ is a constant, $\mathrm{Var}(r_t - \bar{b}) = \mathrm{Var}(r_t)$. According to Theorem 1, we have $\mathrm{Var}(r_t - \bar{b}) = \mathrm{Var}(r_t) \geq \mathrm{Var}(r_t - V^\pi(t))$.

Reduced variance in the predicted rewards. The linear function $\mathbf{x}^\top\boldsymbol{\theta}$ is used to predict the future return of state-action pair (s, a) by following the policy parameterized by $\boldsymbol{\theta}$. The target variate is $\tilde{r} = \mathbf{x}^\top\boldsymbol{\theta} + \epsilon$ and ϵ is the noise on the data. Assuming that ϵ is zero-mean Gaussian noise with the precision (inverse variance) β, we show that the reduced variance in the rewards can be transferred to reduced variance in future predictions. Using Bayesian ridge regression, according to (3.57) and (3.58) of [38], the predictive distribution $p(r|\mathbf{x}, \boldsymbol{r}, \boldsymbol{S}, \beta)$ at a new input $\mathbf{x}$ has variance

$$\eta^2(\mathbf{x}) = \frac{1}{\beta} + \mathbf{x}^\top\boldsymbol{S}\mathbf{x} \tag{15}$$

where $\boldsymbol{S}$ is the variance of the posterior distribution of N observed data. The posterior distribution becomes narrower when new data are observed so the second term goes to zero as $N \to \infty$ [39]. A lower $1/\beta$ therefore leads to a smaller linear regression variance. Using maximum likelihood estimation, we can estimate the β as in (3.21) of [38]:

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left(\tilde{r}_n - \boldsymbol{\theta}^\top\mathbf{x}\right)^2 = \mathrm{Var}(\tilde{r}_t) \tag{16}$$

since $\mathbb{E}[\tilde{r}_t] = \boldsymbol{\theta}^\top\mathbf{x}$. A reduced $\mathrm{Var}(\tilde{r}_t)$ directly leads to a reduced variance in the predicted reward in (15).

V. Experiments

A. Datasets

We conduct experiments on two real-world datasets: Weibo [40] and Pheme [41]. They contain rumors and non-rumors, as well as user, re-posting and comment information. We split the datasets for rumor detector training and testing using a ratio of 7:3. RL is to learn from experience so we perform attacks and focus on results in training set. We randomly select 20% of the authors and their messages in the training set as controllable nodes. There are 213 and 180 target rumors in Weibo and Pheme respectively. Table I shows the dataset statistics.

TABLE I: Dataset statistics. *Subgraphs* are connected components. *Rep&Com* means Re-post and Comment.

	Weibo	Pheme		Weibo	Pheme
Nodes	10280	11950	Rumors	1538	1972
Edges	16412	14737	Non-rumors	1849	3830
Subgraphs	2392	2450	Authors	2440	2837
Relations	3	3	Rep&Com	4453	3311

TABLE II: The detection accuracy(ACC) and precision(PRE) of GCN, GAT, GraphSAGE, RGCN, CGAT and RGCN-G.

	Weibo		Pheme	
Train	ACC	PRE	ACC	PRE
GCN	97.05	96.57	79.56	65.26
GAT	98.10	98.59	83.77	70.61
GraphSAGE	100.00	100.00	98.99	97.11
RGCN	98.02	99.52	84.91	71.39
CGAT	74.30	71.10	77.39	76.23
RGCN-G	97.85	98.33	79.12	79.36
Test	ACC	PRE	ACC	PRE
GCN	71.58	68.60	67.32	51.56
GAT	69.22	65.05	67.49	51.75
GraphSAGE	70.70	68.98	71.17	59.49
RGCN	72.07	74.72	69.33	54.32
CGAT	68.73	62.50	66.80	50.89
RGCN-G	66.47	57.85	70.53	60.21

B. Detector Effectiveness

To verify the effectiveness and generality of the proposed attack algorithm, we attack a wide spectrum of graph neural network models, including GCN [1], GAT [31], GraphSAGE [23], CGAT [2], RGCN [32] and RGCN-G. CGAT learns a robust GCN-based rumor detector with adversarial learning and we use its graph channel as target model. RGCN-G is the RGCN model with only graph structure for eliminating interference of text contents and simulating the zero-text situations. The target models represents message node embedding with a text embedding layer except RGCN-G. The hidden dimension of embedding and hidden layer is 64 and the learning rate is 0.01. The detection performance of target models in training and testing sets of two datasets are shown in Table II.

C. The Performance of Attack

We measure the attack performance with $\Delta NDCG$ in (4) and no truncation in ranking ($\kappa = 0$ in (1)). The larger $\Delta NDCG$ corresponds to the better attack performance. We compare our method to the following four rule-based attacking strategies and two state-of-the-art black-box attack methods:

- **Random and Random+**. It connects edges between users and messages randomly, denoted by Random. Inspired by [2], we propose two variants GU-R and BU-N, denoted by Random+. GU-R connects edges between good users and target rumors randomly and BU-N connects edges between bad users (who post target rumors) and non-rumors

TABLE III: The attack performance results in the metric $\Delta NDCG(\times 10^{-2})$. The horizon T of Weibo and Pheme is 80 and 60 respectively. **Boldfaced font** and $*$ mean the best performance and the runner-up among all methods respectively.

Threat Model	GCN		GAT		GraphSAGE		RGCN		CGAT		RGCN-G	
Dataset	Weibo	Pheme	Weibo	Pheme	Weibo	Pheme	Weibo	Pheme	Weibo	Pheme	Weibo	Pheme
Original	62.43	40.71	63.23	39.46	64.25	44.54	56.03	40.01	64.96	38.67	58.70	45.14
Ramdom	-0.21	0.00	3.24	0.09	1.27	0.52	0.95	0.67	0.80	-0.34	1.45	1.08
Random+	1.20	0.36	4.53	0.25	3.45	1.49	$*$2.11	1.63	1.35	-0.07	2.73	3.88
Degree	1.86	0.35	4.84	0.31	1.07	0.94	1.41	2.42	1.71	0.06	1.86	4.84
Influence	2.01	0.45	5.60	0.11	6.78	1.86	1.61	2.30	$*$2.05	0.04	2.20	7.32
DCG	$*$2.13	$*$0.64	$*$5.87	$*$0.45	$*$7.58	$*$3.05	2.07	$*$2.72	1.45	$*$0.68	$*$4.04	$*$7.38
GC-RWCS	0.00	0.29	4.92	0.27	2.98	0.81	1.46	1.87	2.00	0.26	1.90	2.63
RL-S2V	0.18	0.26	2.92	0.04	2.92	1.50	0.56	1.66	1.23	-0.18	2.34	3.28
AdRumor-RL	**8.95**	**2.45**	**10.13**	**1.34**	**9.49**	**3.36**	**4.83**	**4.52**	**11.68**	**0.72**	**5.48**	**10.79**

randomly. The main idea is that adding edges as above helps the attacker camouflage rumors.

- **Degree**. It selects the target rumor with the highest degree. There are also two variants GU-R and BU-N. The former connects the selected rumor with a random good user, and the latter connects the author of the selected rumor with a random non-rumor. Nodes with high degrees usually have high influence. It aims to attack high influence rumor because the target rumor with large influence value w_i would sharply change the NDCG value in (1).
- **Influence**. It selects the target rumor with the highest influence value defined in (3). GU-R and BU-N are also two variants and work similarly with Degree.
- **DCG**. It calculates rumor DCG value $w_i / \log(f(v_i)+1)$ of target rumor v_i and selects the rumor with the highest DCG value. GU-R and BU-N are variants. DCG is closely related to the objective function in (1) and is a strong baseline.
- **GC-RWCS** [6]. It proposes a black-box node selection strategy with a greedy procedure to calculate the node importance score. We use it to select candidate target rumors and apply GU-R and BU-N variants.
- **RL-S2V** [7]. It is a RL-based graph adversarial attack method. It represents the nodes with deep graph models and makes use of two DQN to choose two nodes respectively within 2-hops of the target node, and then flips the edge between two nodes. To compare with our method, we modify the targeted attack to untargeted attack. we select the top-T rumor with the higher DCG value as the target node for RL-S2V, and then conduct T-times attack.

There are 1000 episodes in total with α=1.0. We average the results of the last 100 episodes. Results are shown in Table III and we show the better one for the variants GU-R and BU-N. We find that, i) AdRumor-RL has the best performance in all situations, even for CGAT that is robust. ii) Some rule-based methods, especially DCG, are strong baselines as they are superior to classical graph attacking models. We also attempt different number of modified edges (horizon T) in Fig. 4. AdRumor-RL is effective with different length of horizon.

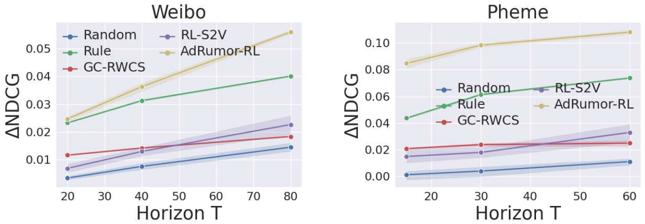


Fig. 4: Attack performance with different horizon on RGCN-G. *Rule* indicates the best performing rule-based method.

D. *Effectiveness of Feature Design*

Superior to deep features. We verify that our designed features are better than end-to-end feature extraction with deep graph models. We replace our designed feature extraction method Φ with another graph neural network model of the same type as the detector in phrases 2 and 5 of Fig. 2. In details, we use a graph neural network model similar with the detector, which outputs 64-dim node embedding in the last convolution layer, followed by a linear classification layer. It is pre-trained using labeled messages and is fixed during RL. The last convolution layer outputs node embedding as node features and averages node embeddings as graph features. The curve Deepfeature in Fig. 5 shows the results and it is far less effective than other methods with designed features.

Feature ablation experiments. We show the effectiveness of different type of features as shown in Fig. 6.

Graph and ranking dependencies. We take the camouflaging features as an example to show its role on two dependencies. Camouflaging messages help lower the ranking of target rumors or exchange with target rumors in ranking, which corresponds to graph and ranking dependencies respectively. We show the effects with two metrics: i) Target rumor ranking drops (TDrop): the total drop in ranking positions of target rumors in the selected subgraphs of each step. ii) Camouflage message rises (CRise): the total risen ranking positions of camouflaging messages in the selected subgraphs of each step and these camouflaging messages must reduce the target rumor rises. TDrop and CRise reflect the role of camouflaging

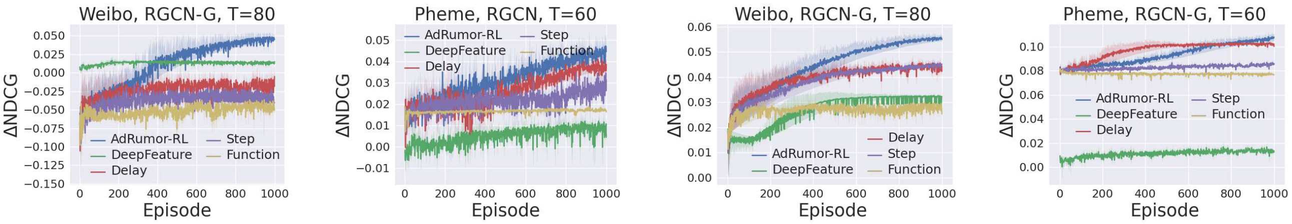
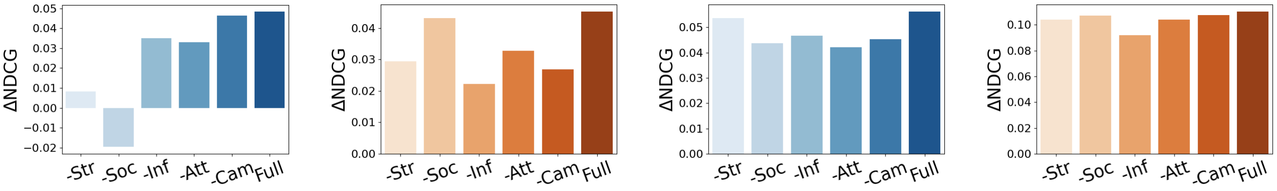
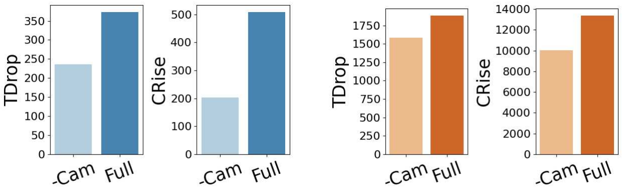


Fig. 5: Comparison experiments on RGCN and RGCN-G. Curves are smoothed and shadows show the standard variances. *AdRumor-RL* is our method. *DeepFeature* extracts deep features. *Delay* regards the delayed return as reward. *Step* uses the mean value of all rewards as the constant baseline. *Function* uses the state-dependent reward baseline.



(a) Weibo, RGCN, T=80 (b) Pheme, RGCN, T=60 (c) Weibo, RGCN-G, T=80 (d) Pheme, RGCN-G, T=60

Fig. 6: Feature ablation experiments on RGCN and RGCN-G. Elimination of structural, social, influence, attack potential, and camouflaging features are named *-Str*, *-Soc*, *-Inf*, *-Att*, and *-Cam*. *Full* indicates no elimination.



(a) Weibo, T=80 (b) Pheme, T=60

Fig. 7: The average TDrop and CRise values for all steps in best 100 episodes on RGCN-G.

messages on graph and ranking dependencies respectively. In Fig. 7, we can see higher TDrop and CRise with camouflaging features.

E. Effectiveness of Credit Assignment and Baseline Design

Credit assignment. We use delayed return for comparison. It calculates $\Delta NDCG$ in (4) and normalizes it as each step reward. The curve Delay in Fig. 5 shows the results.

Reward baseline. To show the effectiveness of the time-dependent baseline, we compare our work with two common baselines and results are shown in Fig. 5.

- **Step** is the common constant baseline [28] as (12). It averages all history $r(S_t, A_t)$ as the baseline.
- **Function** is the common state-dependent method [19], [20], which learns the linear state-value function $V(\mathbf{x}) = W\mathbf{x}+b$ to predict the baseline, where $\mathbf{x}$ is the subgraph/node pair features. It updates parameters W and b by minimizing the mean square error loss with $r(S_t, A_t)$ at each episode. Two functions are used on subgraph and node level, respectively.

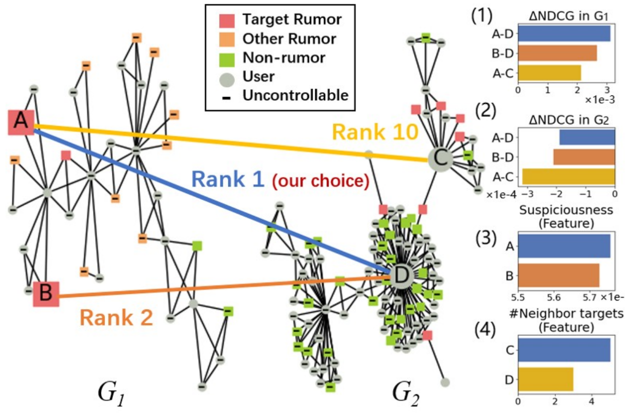


Fig. 8: Case study for graph dependencies. *Left*: The selected subgraph pair (G_1, G_2) in Weibo and three edges with their performance ranking: A-D (rank 1), B-D (rank 2) and A-C (rank 10). *Right*: (1) $\Delta NDCG$ of targets in G_1 (see target rumors in G_1 as targets in (1)). (2)$\Delta NDCG$ of targets in G_2. (3) The rumor probability of target rumor A and B. (4) The number of neighbor target rumors for the user C and D.

F. Case Study

We shows that AdRumor-RL captures the graph dependencies with designed features. In Figure 8, we pick the subgraph pair (G_1, G_2) of a step that performs well in the experiment and traverse all node level actions for (G_1, G_2), and then display 3 edges and their $\Delta NDCG$ ranking. We can see that AdRumor-RL chooses the best pair A-D. Suspiciousness drives the agent to choose the target rumor A in G_1 instead of B and it achieves the highest positive effects. The number of neighbor targets helps the agent choose the user D in G_2 instead of C, which reduces the lowest negative effects the most. These features help balance positive and negative effects under the situation of graph dependencies.

VI. Conclusion

In this paper, we propose AdRumor-RL, an interpretable and effective hierarchical attack framework against GCN-based rumor detector. We define a practical attack object with realistic constraints and use RL to realize black-box attacks. Interpretable attacking features are designed to capture graph dependencies and ranking dependencies. We design a credit assignment method to speed up learning and a time-dependent baseline to reduce variance. With specific feature design, this attack framework can be extended to different types of graphs and more applications in social networks.

Acknowledgment

Xi Zhang, Yuefei Lyu and Xiaoyu Yang are supported by the Natural Science Foundation of China (No.61976026) and the 111 Project (B18008). Sihong Xie and Jiaxin Liu are supported in part by NSF under grants CNS-1931042. Philip Yu is supported in part by NSF under grant SaTC-1930941.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[2] X. Yang, Y. Lyu, T. Tian, Y. Liu, Y. Liu, and X. Zhang, "Rumor detection on social media with graph structured adversarial learning," in *IJCAI*, 7 2020, pp. 1417–1423.

[3] V. Nguyen, K. Sugiyama, P. Nakov, and M. Kan, "FANG: leveraging social context for fake news detection using graph representation," *Commun. ACM*, vol. 65, no. 4, pp. 124–132, 2022.

[4] T. Bian, X. Xiao, T. Xu, P. Zhao, W. Huang, Y. Rong, and J. Huang, "Rumor detection on social media with bi-directional graph convolutional networks," in *AAAI*, 2020, pp. 549–556.

[5] Y.-J. Lu and C.-T. Li, "GCAN: Graph-aware co-attention networks for explainable fake news detection on social media," in *ACL*, 2020, pp. 505–514.

[6] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80, 2018, pp. 1123–1132.

[7] J. Ma, S. Ding, and Q. Mei, "Towards more practical adversarial attacks on graph neural networks," in *NeurIPS*, vol. 33, 2020, pp. 4756–4766.

[8] N. O. Hodas and K. Lerman, "How visibility and divided attention constrain social contagion," in *SocialCom/PASSAT*, 2012, pp. 249–257.

[9] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: Quantifying influence on twitter," in *WSDM*, 2011, p. 65–74.

[10] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *KDD*, 2018, p. 2847–2856.

[11] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *ICLR*, 2019.

[12] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial examples for graph data: Deep insights into attack and defense," in *IJCAI*, 7 2019, pp. 4816–4823.

[13] A. Alharin, T.-N. Doan, and M. Sartipi, "Reinforcement Learning Interpretation Methods: A Survey," *IEEE Access*, vol. 8, pp. 171 058–171 077, 2020.

[14] V. K. Garg, S. Jegelka, and T. S. Jaakkola, "Generalization and representational limits of graph neural networks," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 119, 2020, pp. 3419–3430.

[15] J. B. Lee, R. A. Rossi, X. Kong, S. Kim, E. Koh, and A. Rao, "Graph convolutional networks with motif-based attention," in *CIKM*, 2019, pp. 499–508.

[16] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *ICLR*, 2020.

[17] A. K. Agogino and K. Tumer, "Unifying temporal and structural credit assignment problems," in *AAMAS*, 2004, pp. 980–987.

[18] A. Harutyunyan and et al., "Hindsight credit assignment," in *NeurIPS*, 2019, pp. 12 467–12 476.

[19] R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *NeurIPS*, 1999, pp. 1057–1063.

[20] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *NeurIPS*, 1999, pp. 1008–1014.

[21] G. Tucker, S. Bhupatiraju, S. Gu, R. E. Turner, Z. Ghahramani, and S. Levine, "The mirage of action-dependent baselines in reinforcement learning," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80, 2018, pp. 5022–5031.

[22] C. Wu, A. Rajeswaran, Y. Duan, V. Kumar, A. M. Bayen, S. M. Kakade, I. Mordatch, and P. Abbeel, "Variance reduction for policy gradient with action-dependent factorized baselines," in *ICLR*, 2018.

[23] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, pp. 1024–1034.

[24] Z. He, C. Li, F. Zhou, and Y. Yang, "Rumor detection on social media with event augmentations," in *SIGIR*, 2021, pp. 2020–2024.

[25] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 97, 2019, pp. 695–704.

[26] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[27] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," in *J. Mach. Learn. Res.*, 2004.

[28] L. Weaver and N. Tao, "The optimal reward baseline for gradient-based reinforcement learning," in *UAI*, 2001.

[29] P. Marbach and J. N. Tsitsiklis, "Simulation-based optimization of markov reward processes," *IEEE Transactions on Automatic Control*, vol. 46, no. 2, pp. 191–209, 2001.

[30] H. Mao, S. B. Venkatakrishnan, M. Schwarzkopf, and M. Alizadeh, "Variance Reduction for Reinforcement Learning in Input-Driven Environments," *ICLR*, 2019.

[31] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *stat*, vol. 1050, p. 20, 2017.

[32] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web*, 2018, pp. 593–607.

[33] V. Mnih and et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, feb 2015.

[34] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *WWW*, 2010, pp. 661–670.

[35] M. Seo, L. F. Vecchietti, S. Lee, and D. Har, "Rewards prediction-based credit assignment for reinforcement learning with sparse binary rewards," *IEEE Access*, vol. 7, pp. 118 776–118 791, 2019.

[36] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *AAAI*, 2018, pp. 2974–2982.

[37] H. van Seijen, M. Fatemi, R. Laroche, J. Romoff, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," in *NeurIPS*, 2017, pp. 5392–5402.

[38] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, 2006, vol. 4, no. 4.

[39] C. S. Qazaz, C. K. I. Williams, and C. M. Bishop, *An Upper Bound on the Bayesian Error Bars for Generalized Linear Regression*, Boston, MA, 1997, pp. 295–299.

[40] C. Song, C. Yang, H. Chen, C. Tu, Z. Liu, and M. Sun, "Ced: Credible early detection of social media rumors," *TKDE*, vol. 33, no. 8, pp. 3035–3047, 2021.

[41] A. Zubiaga, M. Liakata, and R. Procter, "Learning reporting dynamics during breaking news for rumour detection in social media," *CoRR*, vol. abs/1610.07363, 2016.