Blocking Influence at Collective Level with Hard Constraints (Student Abstract)

Zonghan Zhang[1], Subhodip Biswas[2], Fanglan Chen[2], Kaiqun Fu[3], Taoran Ji[2], Chang-Tien Lu[2], Naren Ramakrishnan[2], Zhiqian Chen[1]

[1]Mississippi State University
[2]Virginia Tech
[3]South Dakota State University
zz239@msstate.edu, subhodip@cs.vt.edu, fanglanc@vt.edu, kaiqun.fu@sdstate.edu, jtr@vt.edu, ctlu@vt.edu, naren@cs.vt.edu, zchen@cse.msstate.edu

Abstract

Influence blocking maximization (IBM) is crucial in many critical real-world problems such as rumors prevention and epidemic containment. The existing work suffers from: (1) concentrating on uniform costs at the individual level, (2) mostly utilizing greedy approaches to approximate optimization, (3) lacking a proper graph representation for influence estimates. To address these issues, this research introduces a neural network model dubbed Neural Influence Blocking (NIB) for improved approximation and enhanced influence blocking effectiveness. The code is available at https://github.com/oates9895/NIB.

Introduction

As transportation and communication technology bring people closer together and shrink the world, the influence on graphs grows more complicated, attracting increasing attention from researchers and practitioners. Influence blocking maximization (IBM) (He et al. 2012), one of the most studied topics, aims to discover effective ways for preventing harmful influence from propagating throughout the network.

One of the most effective ways to stop the spread is node removal (Tong et al. 2010). It often comes with hard constraints, turning the IBM problem into a constrained combinatorial optimization problem (Tong et al. 2017). Most previous research tackle this problem from individual level perspective assuming submodularity and uniform cost for removing nodes (Ibrahim, Hefny, and Hassanien 2018). Therefore, the greedy approach is the most common approach to approximate optimization. However, these presumptions does not hold when nodes in graphs represent collective concepts. For instance, cities in a national transportation network have different population, and isolating cities from the network should have different costs. As a result, this issue resembles a 0-1 knapsack problem, where the greedy method can only guarantee a $\frac{1}{2}$-optimization (Martello 1990).

To better characterize the collective-level influence, this paper presents a new approach dubbed Neural Influence Blocking (NIB) that enforces hard constraints and optimizes nodes selection to maximize the total scores with a neural networks. The main contributions of this paper are: **(1)** formulate influence blocking as a combinatorial optimization on graph with hard constraints; **(2)** design an effective and efficient neural network to handle this problem; and **(3)** conduct extensive evaluations on synthetic networks.

Neural Influence Blocking

Problem Setup

On a weighted and undirected graph $G = (V, E, C, A)$, where $V = \{v_1, v_2, \ldots, v_i, \ldots, v_N\}$ and $E = \{e_{ij}\}^{N \times N}$ stand for vertices and edges, respectively. Each node is weighted by the cost $c_i \in C \in \mathbb{R}^N$, and each edge is weighted by the probability $a_{ij} \in A \in \mathbb{R}^{N \times N}$ where A denotes the weighted adjacency matrix.

Given G and a hard constraint $k \in \mathbb{R}^+$, our goal is to find a node selection $y_i = \{0, 1\} \in Y^N$ that solves:

$$\underset{Y}{\text{maximize}} \sum_{y_i=1} s_i \cdot y_i, \quad \text{s.t.} \sum_{y_i=1} c_i \cdot y_i \leq k, \tag{1}$$

where $s_i \in S \in \mathbb{R}^N$ represents the score of node i calculated by an approximation algorithm based on the graph structure.

Node-level Proxy Function

The proxy function is critical to nodes selection such that it generates a score of blocking effect for each node. A state-of-the-art proxy function ranks the nodes with a vector $\phi = I + AI + ... + A^r I$ (Yan et al. 2019).

However, when node v belongs to the i_{th}-order neighborhood and the j_{th}-order neighborhood of node u at the same time given $i < r$ and $j < r$, the expected number of nodes activated by node u within r time steps is not calculated correctly. To fix the flaw, assuming that the influence of each time step is independent of all the other steps, we define a matrix $Pr = I - \prod_{r=1}(1 - A^r)$, where $\prod$ represents the element-wise product of the matrices. Here $1 - A^r$ denotes the probabilities of the nodes not activating each other in the r_{th} time step. Pr computes the probabilities of each node activating all other nodes respectively within r time steps. The multiplication of a unit column vector with matrix Pr will provide us a column vector in which each score is the expectation of the number of nodes activated in r time steps if the corresponding node is initially active.

Optimization with Hard Constraints

After scoring, the network influence blocking transfers into a 0-1 knapsack problem. Dynamic programming is the most

efficient algorithm to optimize such problems with a time complexity of $\mathcal{O}(NW)$ where N is the number of items and W denotes the cost constraint (Martello, Pisinger, and Toth 1999). As an efficient method to approximate the optimization, the greedy algorithm only guarantees a $\frac{1}{2}$-optimization (Martello 1990). To achieve a better approximation with an acceptable time cost, NIB leverages a neural network to incorporate the hard constraint into a cost function. NIB takes the cost-score $(c_i, s_i)^N$ pair as input and uses a hidden-layer to turn the pair into one single value, which is further compressed into range $(0, 1)$ with a sigmoid function. The optimal selection can be calculated through the function below:

$$y^* = \arg\min_y [\overbrace{\max(0, \textstyle\sum_{i=1}^{N}(c_i \cdot y_i) - k)}^{\text{Hard Constraint}} - \overbrace{\lambda \textstyle\sum_{i=1}^{N} s_i \cdot y_i}^{\text{Scores in Total}}]. \tag{2}$$

The first term applies the hard cost constraint to the learning process while the second term defines the total score of the selected node subset. λ is a parameter determining the relative importance of the total score comparing to that of enforcing the constraint. A smaller λ will make the neural network more cautious with node selection.

Empirical Evaluation

Experiment Settings

All experiments are conducted on a computer equipped with Ubuntu 20, 16-core CPU (3.0GHz), and 32GB memory. Ten power-law graphs are generated randomly for the evaluation. Simulations are run ten times on each graph to reduce the uncertainty brought by the simulation process. We compare NIB with four baselines: randomly selection (RD), greedy algorithm with a traditional score function (GD), greedy algorithm with the improved score function (IGD), and dynamic programming (DP). We evaluate the methods on the Independent Cascade (IC) model (Kempe, Kleinberg, and Tardos 2003) which is a classic propagation model.

Results and Analysis

Extra Protection We measure the extra protection effect by taking the number of initially selected individuals away from the number of individuals not activated as a result of the intervention. Figure 1 shows that NIB achieves the highest extra protection rate when the selection rate is higher than 30%. Our score function also consistently outperforms the traditional score function.

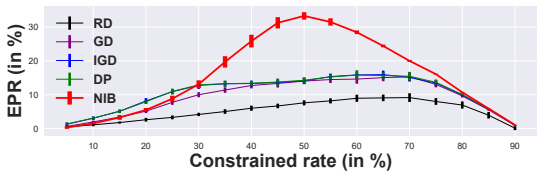

Figure 1: Extra protection rate (EPR) of the methods.

Total Protection We also notice that NIB achieves 90% total protection rate as early as when 70% of the population are strategically blocked from the network. This finding suggests that NIB is not only capable of achieving the highest protection rate but also the most resource-efficient among the methods.

Time Efficiency NIB requires extra time to learn the probability of each node being selected. Thus, it is slightly slower than the greedy algorithms, but still reasonably fast. Different from the dynamic programming algorithm, NIB and the greedy methods spend most of the time on shuffling and calculating the scores. Thus, the time spent by those methods does not depend on the cost constraint.

Conclusion

This paper proposes a collective-level network influence blocking problem and approached it as a combinatorial optimization with hard constraints. To better model this problem, a new score function is designed. Neural network is adopted to approximately optimize the node selection. Extensive empirical experiments demonstrate that our approach achieves promising performance beyond the state-of-the-art methods by a significant margin. In the future, a more fine-grained learning method can be identified to further improve the efficiency and the effectiveness of the node selection.

References

He, X.; Song, G.; Chen, W.; and Jiang, Q. 2012. Influence blocking maximization in social networks under the competitive linear threshold model. In *Proceedings of the 2012 siam international conference on data mining*, 463–474. SIAM.

Ibrahim, R. A.; Hefny, H. A.; and Hassanien, A. E. 2018. Controlling social information cascade: a survey. *Big Data Analytics*, 196–212.

Kempe, D.; Kleinberg, J.; and Tardos, É. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 137–146.

Martello, S. 1990. Knapsack problems: algorithms and computer implementations. *Wiley-Interscience series in discrete mathematics and optimiza tion*.

Martello, S.; Pisinger, D.; and Toth, P. 1999. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management science*, 45(3): 414–424.

Tong, G.; Wu, W.; Guo, L.; Li, D.; Liu, C.; Liu, B.; and Du, D.-Z. 2017. An efficient randomized algorithm for rumor blocking in online social networks. *IEEE Transactions on Network Science and Engineering*, 7(2): 845–854.

Tong, H.; Prakash, B. A.; Tsourakakis, C.; Eliassi-Rad, T.; Faloutsos, C.; and Chau, D. H. 2010. On the vulnerability of large graphs. In *2010 IEEE International Conference on Data Mining*, 1091–1096. IEEE.

Yan, R.; Li, D.; Wu, W.; Du, D.-Z.; and Wang, Y. 2019. Minimizing influence of rumors by blockers on social networks: algorithms and analysis. *IEEE Transactions on Network Science and Engineering*, 7(3): 1067–1078.