Query-Decision Regression for Misinformation Prevention in Social Networks

Siqi Wang, Jiahao Xie, Yifan Wang, and Guangmo Tong(✉)

University of Delaware, Newark De 19711, USA
{wsqbit,jiahaox,yifanw,amotong}@udel.edu

Abstract. Misinformation prevention is a crucial research topic in the social network community, which aims to identify, mitigate, and ultimately prevent the spread of false or misleading information across the social network. Traditional methods typically rely on detailed information on underlying networks and propagation mechanisms. However, these approaches can be impractical due to privacy concerns and restrictions on data accessibility. To overcome this challenge, we propose MetaLearner, a novel framework that leverages historical pairs containing queries and their high-quality decisions. This approach enables us to derive effective decisions for new queries without requiring complete network and diffusion information. We evaluate the performance of MetaLearner by comparing it with existing methods. The results show significant improvements in both accuracy and computational efficiency, demonstrating MetaLearner as a scalable and effective solution for misinformation prevention.

Keywords: Misinformation Prevention · Social Network · Data-driven Learning

1 Introduction

Social networks can connect people worldwide and enable rapid information exchange, but they also facilitate the spread of misinformation, which refers to false, inaccurate, or misleading information [4,16]. Preventing misinformation on social media is essential to avoid distorting perceptions, public opinion impact, and harmful consequences [1]. Additionally, disseminating accurate information is important for informed decision-making [11], public safety [2], and maintaining trust in institutions [14].

An effective strategy for preventing misinformation is to select influential users in social networks to disseminate accurate information before others encounter false content [3,13,23]. This approach selects key individuals to spread truthful information, thereby minimizing the number of users affected by misinformation. The spread of information within networks can be captured by various diffusion models [7,9,12], which indicate that the number of users

J. Kertesz et al. (Eds.): CSoNet 2024, LNCS 15417, pp. 48–60, 2025.
https://doi.org/10.1007/978-981-96-6389-7_5

unaffected by misinformation can be represented by a monotone-increasing submodular function [3]. Previous works leverage this property to design greedy algorithms for misinformation prevention in social networks [3,6,8]. However, these methods typically require comprehensive information of social networks and diffusion models, which is usually inaccessible in real-world scenarios due to privacy concerns [17]. This constraint leads to an unknown objective function for misinformation prevention, limiting the practical application of such methods.

Recently, Tong proposed StratLearner [18], an effective method for estimating the objective function using randomly sampled subgraphs and historical query-decision pairs, without requiring complete diffusion model information. In each query-decision pair, the query represents users initially affected by misinformation and the decision refers to users selected to minimize misinformation spread. However, StratLearner has two main limitations: 1) Its performance depends on the number of sampled subgraphs, with more subgraphs improving accuracy but increasing training time and memory cost; 2) It shows limited performance gains even with larger training datasets, similar to the findings in Model Soup [22], often converging to the same sub-optimal region with identical hyperparameters.

To address above limitations, we propose MetaLearner, a two-step framework for misinformation prevention in social networks with restricted access to the underlying diffusion model. In the first step, we refine the sampling method by leveraging historical query-decision pairs. Multiple models are trained on distinct subsets of training data and randomly sampled subgraphs, producing estimations of the objective function. Since we cannot directly select the most accurate model due to limited access to the true objective function, we draw inspiration from Multi-head Attention [21] and Model Soup [22], considering each estimation as a representation of the social network graph. Ideally, we would train numerous models, with each providing a representation to replace a randomly sampled subgraph, thereby enhancing the accuracy of objective function estimation in the second step. However, this approach is computationally infeasible. To address this, in the second step, we introduce an importance sampling method that reduces the number of required models while still generating high-quality subgraphs for objective function estimation. Once the objective function is estimated, traditional misinformation prevention algorithms can be employed to optimize it to generate solutions for new queries.

Overall, our contributions are summarized: 1) We propose MetaLearner, a novel framework inspired by model ensemble techniques, enhancing the accuracy of objective function estimation by a two-step learning process; 2) We introduce an importance sampling strategy to reduce the number of required models to $\frac{1}{10}$, $\frac{1}{40}$, $\frac{1}{80}$, and $\frac{1}{160}$ of the original amount. This reduction significantly decreases computational cost and memory usage, making the approach practical for applications with limited data access; 3) We conduct extensive experiments, showing that MetaLearner consistently outperforms the state-of-the-art methods in accuracy and computational efficiency across various social network structures, demonstrating its robustness and scalability for large-scale applications.

The rest of the paper is organized as follows. Section 2 introduces the problem, Sect. 3 describes the proposed method, Sect. 4 presents the experimental results, and Sect. 5 concludes the paper.

2 Problem Setting

A social network can be modeled as a directed graph $G = (V, E, T)$, where V represents the user set and E represents the connections between users. Specifically, a directed edge $(u, v) \in E$ with $u, v \in V$, indicates that the user u can spread information to user v. T is a function that maps each edge (u, v) to a real value $T(u, v) \in (0, +\infty)$ which represents the time cost for spreading information from u to v. In addition, for each node $v \in V$, its out-neighbor set and in-neighbor set are denoted as $N_{out}(v) \subseteq V$ and $N_{in}(v) \subseteq V$, respectively.

The considered diffusion model follows the mechanism in triggering model [12]. Given a social graph G, two types of cascades are disseminated simultaneously: the misinformation cascade $\mathcal{M}$ and the positive information cascade $\mathcal{P}$. Each node in the graph can be in one of three activation states: $\mathcal{M}$-active state, $\mathcal{P}$-active state, and inactive state. Initially, a set of nodes $M \subseteq V$ is $\mathcal{M}$-active, while another set of nodes $P \subseteq V$ is selected to be $\mathcal{P}$-active. The remaining nodes are inactive. The diffusion process can be defined as follows.

- **Timestep** 0: Nodes in the $\mathcal{M}$-active and $\mathcal{P}$-active states begin attempting to activate their neighbors.
- **Timestep** t: When a node u becomes $\mathcal{M}$-active at time t, it attempts to activate all its inactive out-neighbors $N_{out}(u)$. Each neighbor $v \in N_{out}(u)$ will become $\mathcal{M}$-active at time $t + T(u, v)$. Similarly, if a node u becomes $\mathcal{P}$-active at time t, it attempts to activate all its inactive out-neighbors to become $\mathcal{P}$-active, following the same process. A node can only be activated by the first attempt from any in-neighbors and will not switch states afterward. If a node is activated by multiple in-neighbors in different states simultaneously, it will become $\mathcal{M}$-active.
- **Termination**: The diffusion process terminates when no more nodes can be activated.

Given a social network $G = (V, E, T)$, a diffusion model described above, a node set $M \subseteq V$, and a budget $r \in \mathbb{Z}^+$, the objective of misinformation prevention problem is to select a set of nodes $P \subseteq V$ with $|P| \leq r$ and $P \cap M = \emptyset$ such that the number of nodes that are not $\mathcal{M}$-active after the diffusion process can be maximized. Let $F(M, P, G)$ represent the number of nodes that are not $\mathcal{M}$-active after the diffusion process. The objective function $f(M, P, G)$ is defined as follows.

$$f(M, P, G) = F(M, P, G) - F(M, \emptyset, G)$$

where $F(M, \emptyset, G)$ represents the number of nodes that are not $\mathcal{M}$-active when no positive nodes are initially selected. Formally, the misinformation prevention problem aims to find a node set P such that the objective function is maximized.

$$\underset{P \subseteq V, P \cap M = \emptyset, |P| \leq r}{\arg\max} f(M, P, G). \tag{1}$$

Algorithm 1. MetaLearner

1: Input: $N, n, K \in \mathbb{Z}^+, \phi, D$;
2: Update the sampling method ϕ to a set of sampling methods ϕ' with Algorithm 2;
3: Estimate the objective function through Algorithm 3;
4: Return an estimation of the objective function $f^*(M, P, G)$;

In this paper, the collected data is represented as $D = \{M_i, P_i\}_{i=1}^m$ where each P_i is an approximate decision for $M_i \in \mathcal{X}$ with its corresponding objective function using some traditional misinformation prevention methods, where $\mathcal{X}$ denotes the query space. Our goal is to learn a framework $\mathcal{A}$ for the misinformation prevention problem based on the historical query-decision data such that it can obtain a good decision $\hat{P} = \mathcal{A}(M)$ for a new given query $M \in \mathcal{X}$. We use l to measure the performance of $\mathcal{A}$.

$$l(M, \mathcal{A}(M)) = 1 - \frac{f(M, \hat{P}, G)}{f(M, P', G)},$$

where P' is an approximate decision for the new query M, obtained using the same method applied to compute the decisions in the collected data. We prefer lower values of l, as they indicate higher quality in the predicted decision $\hat{P}$. Formally, given the training data D, we aim to develop a framework $\mathcal{A}$ such that the following loss function is minimized.

$$\mathcal{L}(\mathcal{A}) = \mathbb{E}_{M \in \mathcal{X}}[l(M, \mathcal{A}(M))].$$

3 Method

In this section, we introduce our proposed framework MetaLearner. The main challenge stems from the unknown objective function $f(M, P, G)$ due to the lack of information about the underlying diffusion models. To overcome this challenge, we follow the idea in [18] that the objective function can be approximated by using the subgraphs randomly drawn from some fixed distribution. We assume that the objective function lies in the following space parameterized by some learnable $w \in \mathbb{R}$.

$$\left\{ \sum_{i=1}^{K} w_i \cdot f(M, P, g_i) \,\middle|\, w_i \in \mathbb{R}, g_i \in S, k \in \mathbb{Z}^+ \right\}, \tag{2}$$

where S indicates a set of subgraphs with size K that are sampled from the true graph structure $G = (V, E)$. Specifically, $S = \{g_1, \cdots, g_K\}$ and $g_i = (V_i, E_i, T_i)$ where $V_i \subseteq V$, $E_i \subseteq E$, and T_i is a function that assigns each edge in g a non-negative weight. Therefore, our approach focuses on accurately estimating $f(M, P, G)$ via searching from the above space by using the historical query-decision data. We can then apply traditional optimization algorithms to infer a

Algorithm 2. Sampling modification

Input: $N, n, K \in \mathbb{Z}^+, \phi, D$;
for $j = 1, 2, \cdots, N$ **do**:
  Sample K subgraphs with the sampling method ϕ to construct $S^j = \{g_1^j, \cdots, g_K^j\}, g_i^j = G(V_i^j, E_i^j, T_i^j)$;
  Sample n training pairs from the dataset D to construct D^j;
  Calculate $\mathbf{w}^j = \{w_1^j, \cdots, w_K^j\}$ for D^j, S^j by applying cutting plane algorithm [10] to solve Equation 3;
  Construct an edge set $E^j = \bigcup_{i=1}^K E_i^j$, where E_i^j is the edge set of subgraph g_i^j;
  for $e \in E_i^j$ **do**:
    Calculate the weighted weight $T^j(e) = \sum_{i=1}^K w_i^j \cdot T_i^j(e)$, $T_i^j(e) = 0$ if $e \notin g_i^j$;
    Sample a from a uniform distribution $[A, B]$, with positive integers A, B;
    Calculate b by solving $\mathbb{E}[\text{Weibull}(a, b)] = T^j(e)$;
    $\phi^j(e) = \text{Weibull}(a, b)$;
  end for
  $\phi^j = \{\phi^j(e)\}, e \in E^j$;
end for
Return a set of sampling method $\phi' = \{\phi^1, \cdots, \phi^N\}$;

decision P when given a new query M by solving Eq. 1. In general, the overall framework is composed of two main steps: in the sampling modification step, we train $N \in \mathbb{Z}^+$ different models using $n \in \mathbb{Z}^+$ training pairs to learn an effective sampling method that approximates the true graph structure. In objective function estimation step, we estimate the objective function using the learned sampling method, yielding the final approximation $f^*(M, P, G)$. The overall algorithm is summarized in Algorithm 1.

Sampling Modification. We summarize this step in Algorithm 2. Following the idea of the multi-head attention mechanism, this approach estimates the social network graph by analyzing different groups of weighted subgraphs across various subsets of training data, resulting in a diverse set of sampling methods. Specifically, we use ϕ to denote the initial sampling method, which is a function that samples a weighted subgraph $g = (V', E', T')$ from a graph structure $G = (V, E)$, where $V' \subseteq V, E' \subseteq E$, and T' is a function that assigns a non-negative weight to each edge in g. Given an initial sampling method ϕ and the training dataset D, MetaLearner samples $N \in \mathbb{Z}^+$ groups of different weighted subgraphs $\{S^1, \cdots, S^N\}$ using ϕ and N subsets of training datasets $\{D^1, \cdots, D^N\}$ from D. The size of each subgraph group is denoted by $K \in \mathbb{Z}^+$, and the size of each training data subset by $n \in \mathbb{Z}^+$. Using each dataset D^j and subgraph group S^j, we obtain estimated objective functions from the search space following the margin-based maximization strategy by training N Structured Support Vector

Machines (SSVM) [20] as follows.

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{w}^j\|_2^2 + \frac{C}{2n}\sum_{i=1}^{n}\xi_i \\ &\text{s.t.} \sum_{k=1}^{K} w_k^j \cdot f(M_i, P_i, g_k) - \sum_{k=1}^{K} w_k^j \cdot f(M_i, P^*, g_k) \geq \alpha \cdot H(P_i, P^*) - \xi_i, \\ &\forall i \in [n], j \in [N], \forall (M_i, P_i) \in D^j, \forall P^* : |P^*| < r, P^* \neq P_i, \end{aligned} \tag{3}$$

where $C, \alpha \in \mathbb{R}^+$ are hyperparameters. The function $H(P_i, P^*)$ measures the similarity between two decisions. Specifically, $H(P_i, P^*) = h(P_i, P_i) - h(P_i, P^*)$, where $h(P_i, P^*) = |N_{P_i} \cap N_{P^*}|$ with N_P denoting the set of nodes directly connected to any of the nodes in P, including P itself.

After training the N SSVMs, MetaLearner obtain a group of weight vectors $\{\mathbf{w}^1, \cdots, \mathbf{w}^N\}$ with size N, where $\mathbf{w}^j = \{w_1^j, \cdots, w_K^j\}$ and $j \in [N]$. Each SSVM searches for an objective function that is most suitable for the given dataset D^j from the search space in Eq. 2. After obtaining the weights $\mathbf{w}^j$, MetaLearner estimates the objective function as follows.

$$f^*(M, P, G) = \sum_{i=1}^{K} w_i^j \cdot f(M, P, g_i^j), w_i^j \in \mathbf{w}^j, g_i^j \in S^j.$$

MetaLearner then updates the sampling method based on each trained SSVM. For each group of subgraphs S^j, a corresponding edge set $E^j = \bigcup_{i=1}^{K} E_i^j$ is generated, where E_i^j is the edge set of subgraph g_i^j. The diffusion time $T^j(e)$ for each edge $e \in E^j$ is then calculated as follows.

$$T^j(e) = \sum_{i=1}^{K} w_i^j \cdot T_i^j(e),$$

where $T_i^j(e)$ represents the diffusion time of edge e in subgraph g_i^j. If edge e is absent from subgraph g_i^j, then $T_i^j(e) = 0$. Given these $T^j(e)$, MetaLearner updates the sampling methods. The distribution of the weight on each edge is modeled as a Weibull distribution, denoted by $\phi^j(e) = \text{Weibull}(a, b)$, where $a, b \in \mathbb{R}$ are the distribution parameters. By setting the expectation equal to $T^j(e)$, a new sampling method $\phi^j = \{\phi^j(e)\}, e \in E^j$, is obtained.

Objective Function Estimation. Given the N learned sampling methods $\{\phi^1, \ldots, \phi^N\}$ and the dataset D, MetaLearner proceeds to estimate the final objective function. We outline the objective function estimation process in Algorithm 3. The key challenge is effectively combining the outputs of the multiple SSVMs. Given that each SSVM provides an estimated objective function along with a corresponding social network graph, we explore two strategies for integration. One approach is to uniformly average the learned social network graphs, similar to the uniform soup. The other approach involves training a final SSVM

Algorithm 3. Function estimation

1: Input: $n, K \in \mathbb{Z}^+$, $\{\phi^1, \cdots, \phi^N\}$, D;
2: Sample $\frac{K}{N}$ subgraphs from each new sampling method and combine them as a new group of subgraphs S';
3: Sample n training data from D as training dataset D';
4: Calculate $\mathbf{w}' = \{w'_1, \cdots, w'_K\}$ for D', S' by applying cutting plane algorithm [10] to solve problem 3;
5: Return the final estimation of objective function as $f^*(M, P, G) = \sum_{i=1}^{K} w'_i \cdot f(M, P, g'_i), w'_i \in \mathbf{w}', g'_i \in S'$;

based on the learned social network graphs from the previous SSVMs. Specifically, we can train $N = K$ different SSVMs, and the resulting K subgraphs are used to train the final SSVM, which can estimate the final objective function. However, as the value of K increases, the computational resources and time required also grow, making it infeasible to train a large number of SSVMs.

To address the challenge of scaling, we employ importance sampling, which allows us to balance effectiveness and efficiency by estimating a sampling method from the learned social network graphs and sampling subgraphs accordingly. MetaLearner samples $\frac{K}{N}$ subgraphs from each sampling method ϕ^j to obtain a group of subgraphs S'. Then, another SSVM is trained on a new dataset D', sampled from D, to optimize the final weight vector $\mathbf{w}'$. Altogether, the final objective function is calculated by the following equation.

$$f^*(M, P, G) = \sum_{i=1}^{K} w'_i \cdot f(M, P, g'_i), w'_i \in \mathbf{w}', g'_i \in S'.$$

4 Experiment

In this section, we present empirical studies on our proposed method for the misinformation prevention task across various graph datasets.

4.1 Experiment Setting

Graph Data. Our approach is evaluated on three graph datasets which are generated from classic graph models, i.e., Erdős-Rényi graph (ER) [5], power-law graph (Pow) and Kronecker graph (Kro) [15]. For each generated graph $G(V, E, T)$, we assume that each edge weight follows a Weibull distribution Weibull(a, b) with parameters a and b sampled from the set $\{1, \cdots, 10\}$ independently, and the weight set to the expected value of this distribution.

Sample Generation. The size of the set M is determined by random sampling from a power-law distribution with a parameter of 2.5. Subsequently, we randomly select $|M|$ nodes from the set V to form M. For each query M, we employ the HMP algorithm [19] to compute the approximated decision P on the graph $G(V, E, T)$, with the budget $r = |M|$. A sample pool with a total of 2,500 such pairs (M, P) is generated for each graph dataset.

Baseline Method. We select the state-of-the-art method, StratLearner, as the baseline for our comparisons. We also compare our method with a random algorithm that randomly selects the decision P.

Training Settings. In the experiment evaluating StratLearner, we use $n' = \{270, 540, 1080, 2160\}$ samples for training and 270 samples for testing, reporting the ratio $\frac{f(M,\hat{P},G)}{f(M,P,G)}$, where $\hat{P}$ is the decision made by StratLearner. Each $f(M, P, G)$ is calculated through $10,000$ simulations. The number of weighted graphs K' is set to $\{100, 400, 800, 1600\}$. In the experiment of MetaLearner, we set $N = 10$ and use $n = \{27, 54, 108, 216\}$ samples for training each model and 270 samples for testing, also reporting the ratio $\frac{f(M,\hat{P},G)}{f(M,P,G)}$. The number of weighted graphs K is set to $\{10, 40, 80, 160\}$. This ensures that the total number of training samples and weighted graphs remain consistent across experiments, such that $N \times n = n'$, $N \times K = K'$. For each (n, K) pair, we repeat the experiment five times and report the mean ratio and corresponding variance. The hyperparameter α in Eq. 3 is set to 1,000, and C is set to 0.01. In the StratLearner, each subgraph is generated by selecting each edge independently at random with a probability of 0.01 and assigning a fixed weight of 1.0 to each selected edge. We denote such distribution as $\phi_{1.0}^{0.01}$. In the MetaLearner approach, subgraphs are also generated with a 0.01 probability for selecting each edge, but the edge weights are sampled from a different Weibull(c, d) distribution, with c and d independently sampled from $\{1, \cdots, 10\}$ for each edge during each sampling process. We denote such distribution as $\phi_{WR}^{0.01}$.

4.2 Analysis

Table 1. Results for the Misinformation Prevention on Kro Graph. Each cell shows the average ratio compared with the sample decision, together with the standard deviation (std). The total number of sampled graphs K' are 100, 400, 800 and 1600 respectively.

	Training	Number of sampled grpah			
	Size n'	100	400	800	1600
Strat-Learner	270	0.699 (8E−3)	0.759 (7E−3)	0.785 (1E−2)	0.810 (9E−3)
	540	0.707 (5E−3)	0.743 (8E−3)	0.780 (9E−3)	0.813 (7E−3)
	1080	0.708 (2E−2)	0.760 (1E−2)	0.782 (8E−3)	0.817 (5E−3)
	2160	0.701 (1E−2)	0.756 (1E−2)	0.792 (5E−3)	0.821 (8E−3)
Meta-Learner	270	0.698 (5E−3)	0.777 (8E−3)	0.807 (1E−2)	0.819 (9E−2)
	540	0.694 (1E−2)	0.772 (8E−3)	0.805 (5E−3)	0.827 (8E−2)
	1080	0.704 (8E−3)	0.773 (6E−3)	0.807 (9E−3)	0.831 (8E−2)
	2160	0.710 (1E−2)	0.773 (1E−2)	0.811 (9E−3)	0.834 (9E−2)
Other Method		**Random**	0.190 (5E−3)		

Table 2. Results for the Misinformation Prevention on ER Graph. Each cell shows the average ratio compared with the sample decision, together with the standard deviation (std). The total number of sampled graphs K' are 100, 400, 800 and 1600 respectively.

	Training	Number of sampled grpah			
	Size n'	100	400	800	1600
Strat-Learner	270	0.661 (2E−2)	0.853 (6E−3)	0.873 (1E−2)	0.892 (3E−3)
	540	0.673 (2E−2)	0.861 (1E−2)	0.876 (6E−3)	0.897 (1E−2)
	1080	0.688 (3E−2)	0.844 (9E−3)	0.870 (1E−2)	0.899 (8E−3)
	2160	0.674 (2E−2)	0.857 (2E−2)	0.873 (5E−3)	0.903 (3E−3)
Meta-Learner	270	0.678 (2E−2)	0.853 (2E−2)	0.913 (7E−4)	0.941 (3E−3)
	540	0.645 (2E−2)	0.862 (6E−3)	0.917 (1E−3)	0.938 (5E−3)
	1080	0.669 (3E−2)	0.858 (2E−2)	0.913 (1E−2)	0.936 (3E−3)
	2160	0.660 (3E−2)	0.868 (1E−2)	0.910 (6E−3)	0.937 (4E−3)
Other Method		**Random**	0.052 (2E−2)		

The main results of the StratLearner and the MetaLearner are shown in Table 1. We observe that when the number of sampled graphs is small, the MetaLearner achieves a similar performance compared to StratLearner. As the number of sampled graphs increases, MetaLearner consistently outperforms StratLearner across all three graph datasets. The magnitude of the performance gap between StratLearner and MetaLearner varies across different social network graphs, with additional results provided in Table 2. We believe this is because the function estimation step requires a sufficient number of subgraphs to gain adequate prior knowledge about the social network.

Table 3. Results for Memory and Time Costs on ER Graph. The memory cost is measured in MB, and the time cost is measured in seconds. The total number of sampled graphs K' are 100, 400, 800 and 1600 respectively.

	Training	Memory Cost (MB)				Time Cost (S)			
	Size n'	100	400	800	1600	100	400	800	1600
Strat-Learner	270	51.1	195.5	387.2	772.0	111.0	435.31	987.3	2070.4
	540	53.1	197.9	390.1	776.4	665.2	197.9	1378.1	2677.3
	1080	56.8	202.7	396.4	784.9	294.4	1154.5	2331.5	3848.5
	2160	64.4	212.2	408.5	802.2	518.5	2177.4	4356.2	6819.1
Meta-Learner	270	8.9	26.5	52.8	144.9	94.1	364.9	691.6	1493.9
	540	9.0	26.6	52.9	145.4	154.4	479.1	954.1	2069.4
	1080	9.4	27.1	53.2	145.4	222.8	609.6	1437.6	3260.7
	2160	10.1	27.7	54.2	146.0	358.5	1153.1	2301.4	4764.6

We also report the time and memory cost of the StratLearner and MetaLearner in Table 3. It is not surprising that the memory cost of MetaLearner is less than that of StratLearner since MetaLearner trains models with less training data and fewer sampled subgraphs. However, we are pleased to find that the time cost of MetaLearner is also less than that of StratLearner, even though MetaLearner trains multiple small models while StratLearner trains a single model. These results demonstrate that MetaLearner is more efficient than StratLearner. Additionally, we observe that the number of sampled graphs has a greater impact on time cost than the amount of training data. Increasing the number of sampled graphs significantly improves performance but also increases time cost. Therefore, a trade-off between effectiveness and efficiency must be considered.

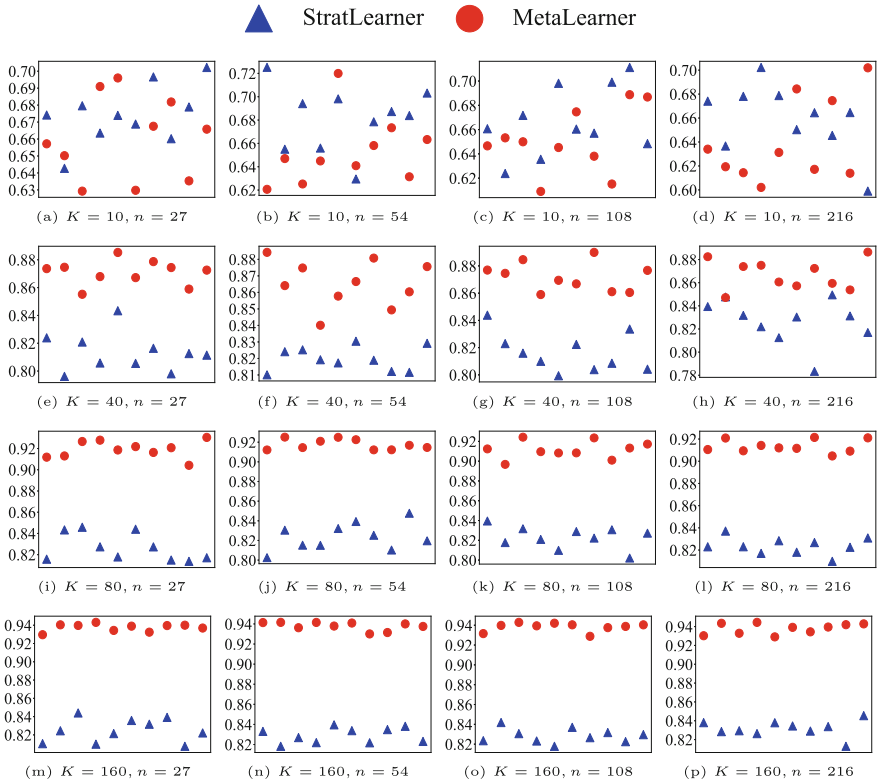

Fig. 1. The results on the ER graph without training. Each subgraph is labeled by the value of K and n and displays the results of 10 repeated experiments without training for StratLearner and MetaLearner. The result indicates the ratio $\frac{f(M,\hat{P},G)}{f(M,P,G)}$.

To evaluate the effectiveness of function estimation step, we conduct experiments to test the performance of StratLearner and MetaLearner without training using the sampling method $\phi_{WR}^{0.01}$ on the ER graph. The results are displayed in

Fig. 1. It is clear that when the number of sampled graphs is insufficient, the sampling method refined by MetaLearner does not necessarily perform better than the initial sampling method $\phi_{WR}^{0.01}$. However, with an increased number of sampled graphs, the quality of the sampling method refined by MetaLearner is improved significantly over the initial sampling method. Moreover, as the number of sampled graphs increases, the performance gap between MetaLearner and the initial method becomes larger. Another interesting observation is that increasing the number of sampled graphs from 100 to 400 significantly improves StratLearner's performance without training. However, further increasing the number of sampled graphs beyond this point results in less noticeable performance gains. This suggests that both StratLearner and MetaLearner require a sufficient number of samples to reach optimal performance levels.

It is worth noting that although we train the models sequentially in the function estimation step of MetaLearner, these models can be trained concurrently. This parallel training approach would significantly reduce the training time but would also increase memory cost. Thus, MetaLearner provides an option to balance between memory cost and time efficiency.

5 Conclusion

In this paper, we introduce MetaLearner, a novel two-step framework designed to enhance the accuracy and efficiency of misinformation prevention in social networks. MetaLearner addresses the challenge of limited access to the complete diffusion model by leveraging historical query-decision pairs and employing refined sampling methods. Our approach combines the strengths of the StratLearner framework with the robustness of ensemble methods, specifically incorporating the concepts of Model Soup and importance sampling. Extensive experiments on various graph structures demonstrate that MetaLearner consistently outperforms the state-of-the-art methods in terms of accuracy. Furthermore, MetaLearner exhibits significant improvements in computational efficiency, reducing both memory usage and training time compared to those of the state-of-the-art methods. This makes MetaLearner a more practical solution for large-scale social network applications. Overall, MetaLearner represents a significant advancement in misinformation prevention techniques for social networks, providing a scalable and efficient solution that can adapt to various network structures and constraints.

Acknowledgments. This project is supported in part by the National Science Foundation under IIS-2414308 and Career IIS-214428.

References

1. Aïmeur, E., Amri, S., Brassard, G.: Fake news, disinformation and misinformation in social media: a review. Soc. Netw. Anal. Min. **13**(1), 30 (2023)

2. Amoruso, M., Anello, D., Auletta, V., Cerulli, R., Ferraioli, D., Raiconi, A.: Contrasting the spread of misinformation in online social networks. J. Artif. Intell. Res. **69**, 847–879 (2020)
3. Budak, C., Agrawal, D., El Abbadi, A.: Limiting the spread of misinformation in social networks. In: Proceedings of the 20th International Conference on World Wide Web, pp. 665–674 (2011)
4. Del Vicario, M., et al.: The spreading of misinformation online. Proc. Natl. Acad. Sci. **113**(3), 554–559 (2016)
5. Erdos, P., Renyi, A.: On random graphs I. Publ. Math. Debrecen **6**, 290–297 (1959)
6. Fan, L., Lu, Z., Wu, W., Thuraisingham, B., Ma, H., Bi, Y.: Least cost rumor blocking in social networks. In: 2013 IEEE 33rd International Conference on Distributed Computing Systems, pp. 540–549. IEEE (2013)
7. Gruhl, D., Guha, R., Liben-Nowell, D., Tomkins, A.: Information diffusion through blogspace. In: Proceedings of the 13th International Conference on World Wide Web, pp. 491–501 (2004)
8. He, X., Song, G., Chen, W., Jiang, Q.: Influence blocking maximization in social networks under the competitive linear threshold model. In: Proceedings of the 2012 SIAM International Conference on Data Mining, pp. 463–474. SIAM (2012)
9. Jiang, Y., Jiang, J.: Diffusion in social networks: a multiagent perspective. IEEE Trans. Syst. Man Cybern. Syst. **45**(2), 198–213 (2014)
10. Joachims, T., Finley, T., Yu, C.: Cutting-plane training of structural SVMs. Mach. Learn. **77**, 27–59 (2009)
11. Karduni, A., et al.: Can you verifi this? Studying uncertainty and decision-making about misinformation using visual analytics. In: Proceedings of AAAI, vol. 12 (2018)
12. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 137–146 (2003)
13. Kumar, K., Geethakumari, G.: Detecting misinformation in online social networks using cognitive psychology. HCIS **4**(1), 1–22 (2014). https://doi.org/10.1186/s13673-014-0014-x
14. Lee, S.J., Lee, C.J., Hwang, H.: The impact of COVID-19 misinformation and trust in institutions on preventive behaviors. Health Educ. Res. **38**(1), 95–105 (2023)
15. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., Ghahramani, Z.: Kronecker graphs: an approach to modeling networks. J. Mach. Learn. Res. **11**(2) (2010)
16. Lewandowsky, S., Ecker, U.K., Cook, J.: Beyond misinformation: understanding and coping with the "post-truth" era. J. Appl. Res. Mem. Cogn. **6**(4), 353–369 (2017)
17. Pham, V., Yu, S., Sood, K., Cui, L.: Privacy issues in social networks and analysis: a comprehensive survey. IET Networks **7**(2), 74–84 (2018)
18. Tong, G.: Stratlearner: learning a strategy for misinformation prevention in social networks. Adv. Neural. Inf. Process. Syst. **33**, 15546–15555 (2020)
19. Tong, G.A., Du, D.Z.: Beyond uniform reverse sampling: a hybrid sampling technique for misinformation prevention. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 1711–1719. IEEE (2019)
20. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y., Singer, Y.: Large margin methods for structured and interdependent output variables. J. Mach. Learn. Res. **6**(9) (2005)
21. Vaswani, A., : Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)

22. Wortsman, M., et al.: Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In: International Conference on Machine Learning, pp. 23965–23998. PMLR (2022)
23. Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M., Procter, R.: Detection and resolution of Rumours in social media: a survey. ACM Comput. Surv. (CSUR) **51**(2), 1–36 (2018)