Bandits for Influence Maximization in Independent Cascade and Topic Aware Independent Cascade

Siddhesh Khandelwal, Mohit Bajaj, Gursimran Singh Muhar
{skhandel, mbajaj01, msimar}@cs.ubc.ca

Abstract

Influence Maximization is the problem of identifying the set of 'seed' users which maximize the spread of influence. This is an important and well studied problem. However, most of the work in this area assumes that we know the probability for each pair of users, or we have the past propagation data. This information is not easy to obtain in a real world scenario. In this project we adopt an online approach to simultaneously learn influence probabilities and maximize the spread. We propose a CMAB (combinatorial multi-armed bandits) framework and show its utility in Independent Cascade (IC) and Topic-aware Independent Cascade (TIC) models. In order to make the framework scalable to large networks, we propose an computationally efficient version of the update step using online expectation maximization. We evaluate the efficacy and feasibility our approach on synthetic and real world datasets.

1 Introduction

In the recent years, the proliferation of internet has led to new ways of communication, viz. social networks. In social networks, users share information which propagates in the news-feeds of their followers. The followers can further react by either endorsing (liking or sharing) or ignoring the news. The endorsed information diffuses further to the second-level followers, and so on. The diffusion of information is of prime interest to many and has been studied widely in various domains like agriculture [8], viral marketing [9, 11, 22, 16], game-theoretic settings [3] and power systems [1].

Motivated by applications to marketing, [15] proposed two information-diffusion models to study the spread of information. One of the models, the independent cascade (IC), allows the advertisers select an initial set of users which act as *seeds* for a given campaign. This information propagates in discrete time-steps and the seeds are given one change to influence its followers with a certain probability known as influence probability. The influenced followers again get one change to influence their followers, and so on. The diffusion stops when the no new nodes can be activated.

The information diffusion models are used by viral marketers who pay a small number of influential seed users to achieve maximum spread. The problem of influence maximization deals with choosing an optimal set of initial *seeds* such that the spread is maximized. Existing work in this domain proposes approximation algorithms for seed selection. However, these algorithms either assume the knowledge of influence probabilities [10, 25, 14] or past propagation data [23, 13] from which these probabilities can be learned. In real life social networks, this information is often not available or difficult to obtain.

Also, the influence probabilities learned are often specific to the information which is being propagated. Hence, when a new item (or product) is encountered, we need to re-learn the influence probabilities using cascade data, which might be limited. [2] proposed a topic-aware IC model (TIC), which postulates that the diffusion of information is topic dependent and different users may have different interests in a topic. It defines latent variables which can assume k states (or topics) to model an abstract interest or pattern. They propose an Expectation Maximization (EM) framework that jointly learns item characteristics and user interests. However, they assume the availability of cascade data, which is often not available.

To solve the problem of limited cascade information, [7, 26] proposed an application of the combinatorial multi-armed bandi (CMAB) framework to the domain of influence maximization (IM). This allowed marketers to start with just the graph structure and little or no knowledge about the influence probabilities, and learn these values over time. The bandit framework involves running the IM campaign for multiple rounds. In each round, the CMAB algorithm incurs a regret in the influence spread due to the lack of knowledge in influence probabilities. The diffusion information obtained in each round is used as feedback to improve the estimates. The aim is to minimize the regret incurred over time. [5] assumed that the diffusion in each round provides information regarding how influence propagated through each edge in the network. [26] extended this approach to a more realistic scenario where we know the status of each node as opposed to the status of each edge. Both [7, 26] proposed methods for just the IC model. Also, [26] used gradient descent in their approach.

In this paper, we propose a combinatorial multi-armed bandit (CMAB) framework for the IC and TIC models for both "edge level" and "node level" feedbacks. We adopt the EM algorithm into an online framework that allows for extension of these methods on large networks. We also evaluate our approaches on real world and synthetic datasets and show that the methods improve their knowledge of unknown parameters while simultaneously trying to optimize spread. The paper is structured as follows: Section 2 discusses the background and related work. In section 3, we define the problem mathematically and propose an algorithm for the solution. We validate our approach by defining experiments and demonstrate its efficacy in section 6. We propose future research directions in section 7 and conclude the paper in section 8.

2 Background and Related Work

2.1 Influence Maximization

The social network is represented as a directed graph $G = (V, E)$, where V represents set of nodes (people) and E represents connections/ followers. For any edge $e \in E$ from node u to node v, we have a diffusion probability $p_{u,v} \in [0, 1]$ that denotes the strength of the connection between u and v. Given a seed set $S \subset V$ and a diffusion model D, the expected number of nodes in G influenced by S is denoted by $\sigma_D(S)$. Given a budget k, the solution to influence maximization problem is finding a set $S^* \subset V$ of size k such that

$$S^* = arg \max_{|S| \leq k} [\sigma_D(S)] \tag{1}$$

Existing work provides a treatment of this problem from various perspectives. [10] consider it as a probabilistic model of interaction and provides heuristics for choosing most influential nodes. [15, 2] treats it as a discrete optimization problem and provides diffusion models in which the spread has monotonic and submodular properties [20]. We used the independent cascade and topic-aware independent cascade in this work. In the following subsections, we define the models and approximation algorithms to solve 1.

2.1.1 Independent cascade (IC)

The independent cascade (IC) model [15] is a stochastic diffusion model. The diffusion in the IC model proceeds in discrete time steps. The central idea is that each recently-activated node u is given one chance to activate any inactive neighbour v. The edge probability $p_{u,v}$ dictates the probability of success. Therefore, if u was active at time t, it can make a one-shot attempt to v at timestep $t+1$ with a probability $p_{u,v}$. However, if u fails in activating v at time $t + 1$, it can make no further attempt to activate v in future time steps. Also, once a node u is active, it cannot become inactive at any time in the future. This process proceeds with an initial set of seed nodes S and continues for nodes which become active in future time steps until no more nodes can be activated. Since this is a stochastic process, we compute the expectation of the spread of a seed set S, which is defined as $\sigma_{IC}(S)$. Under this model, for a budget k, the influence maximization problem aims to find an optimal seed set S^* such that the expected spread is maximized (Equation 1).

2.1.2 Topic-aware independent cascade (TIC)

The TIC model [2] is a topic-aware version of the IC model. It proposes that user interests and item characteristics live a k-dimensional latent topic space. In this setting, for each edge $e = (u, v) \in E$ and each topic $z \in [1, K]$, we have an influence probability $p^z_{u,v}$. For a particular topic z, $p^z_{u,v}$ represents the influence of a user u on a user v. Similarly, the characteristics of an item which propagates in the network can be formulated as a distribution over K topics. Therefore, for each item i and topic z, we have a probability $\gamma^z_i = P(Z = z \mid i)$, with $\sum_{z=1}^{K} \gamma^z_i = 1$.

In this model a propagation happens like in the IC model. When a node v first becomes active on item i, it has one chance of influencing each inactive neighbor v, independently of the history so far. It succeeds with a probability,

$$p^i_{u,v} = \sum_{z=1}^{k} \gamma^z_i p^z_{u,v} \tag{2}$$

2.1.3 Approximation algorithms

Although the solution to 1 is NP-hard, under the monotonic and submodular constraints, polynomial-time approximation algorithms has been proposed with reasonable bounds on accuracy of the solution. Under IC model, [15] showed that the computation of spread ($\sigma_{IC}(S)$) is submodular and monotonic. Under these conditions, [10] provides an greedy algorithm which approximates the optimal solution with a factor of $(1-1/e)$. However the greedy approach is computationally expensive (quadratic) and is empirically known to perform quite slow. [17, 4, 25] proposes a near optimal time approach which gives solution with similar approximation guarantees.

2.2 Learning influence probabilities

The treating of influence maximization problem assumes that we know the pairwise user influence probabilities. However, in real-life settings, they are often not known or are hard to obtain. Various techniques [21, 13, 11] have been proposed in literature to learn influence probabilities. [23] propose an Expectation Maximization (EM) framework for learning probabilities in the IC model. [2] propose an EM framework for learning probabilities in the TIC model. Both these approaches use action-log data of past propagations (cascades) for training. This information is often not available or difficult to obtain in real life settings.

2.3 Multi-Armed Bandits (MAB)

In order to solve the problem mentioned above, [26] proposes the IM problem as an instance of combinatorial multi-armed bandits (CMAB). The CMAB, algorithm A either chooses to *EXPLORE* or *EXPLOIT* and aims to minimize the overall regret due to sub-optimal arm-plays. After each CMAB round, we update our knowledge of influence probabilities (environment) using a feedback mechanism M. The usual feedback mechanism proposed by [26], is the edge-level feedback. However it, assumes that our knowledge of diffusion is to the level of detail where we know the live/ dead status of each edge which is often not available. Instead, we only know the activation sequence of influenced nodes. Under this level of information, [26] suggest a way to distribute the credit among the edges using gradient descent on log-liklihood.

The approaches mentioned in 2.2 solves the problem of learning influence probabilities for both IC and TIC model. However, it assumes that we know the past propagation data, which is often not available. The approach presented in 2.3 solves the problem for IC model by providing an online version using CMAB framework. However, it does not provide an analysis for the TIC model. In this work, we extend the work and implement CMAB framework for the TIC model.

3 Combinatorial Multi-Armed Bandits for Influence Maximization

This section explains the application of the CMAB framework to the IM problem. We first define the requisite notations and the mapping of the CMAB framework to the IM problem in the case of IC and TIC models. Finally, we define the CMAB algorithm for learning probabilities.

3.1 CMAB Mapping to IM

The CMAB framework consists of m base arms. Each arm i is associated to a random variable $X_{i,s}$, which denotes the reward of triggering arm i on trial s. Each arm i is also associated to some unknown distribution μ_i, which is learned over the course of the algorithm. Let A denote a set of base arms (called superarm). In each iteration, a superarm A is played, which triggers all arms in A. In addition, some of the other arms may get probabilistically triggered. Let r_s denote the reward obtained in each round s. Let $T_{i,s}$ denote the total number of times an arm i is triggered until iteration s.

Following the mapping mentioned in [26], each edge $e = (u, v) \in E$ corresponds to a base arm i in the CMAB framework. $X_{i,s}$, that is the reward for each arm i at round s, corresponds to the status of the edge $e = (u, v)$

(live/dead) in each cascade. Superarm A corresponds to the union of all the outgoing edges from the nodes in the seed set S. $T_{i,s}$ corresponds to the number of times edge e is active in s diffusions. The reward r_s is equivalent to the expected spread of the seed set S in the s^{th} cascade. The mean distribution for each arm μ_i corresponds to the influence probability $p_{u,v}$.

The mapping of CMAB to the TIC model is exactly the same as the mapping for the IC model except for the definition of the mean distribution for each arm i (μ_i). In the TIC model, μ_i is composed of K components, each corresponding to one of the k latent topics. Therefore, $\mu_i = \{\mu_i^1, \mu_i^2, \ldots, \mu_i^K\}$. Also, in each iteration of the CMAB algorithm for TIC, for each item i superarm A_i is played. Therefore, if there are I items propagating through the TIC network, each CMAB round would consist of playing a superam A_i for each item i. In the case of TIC, let $A = \{A_1, A_2, \ldots, A_I\}$ be the set of superarms played in each iteration. This implies that there will be a reward for each item i, denoted by r_{s_i}, which is equivalent to the expected spread of the seed set S_i for the i^{th} item.

3.2 CMAB Algorithm

Similar to [26], in each round, the algorithm selects a seed set S (one for each item in the case of TIC) with $|S| = k$ and plays the corresponding superarm A. S can be selected either randomly (EXPLORE) or by solving the IM problem with the current influence probability estimates $\hat{\mu}$ (EXPLOIT). We use ϵ-Greedy [6] as the strategy to select between EXPLORE and EXPLOIT steps. In each round s, this strategy performs exploration with probability ϵ_s and exploitation with probability $1 - \epsilon_s$. The EXPLOIT step requires solving the IM problem with the current estimates $\hat{\mu}$. As both the IC and TIC models are monotone and submodular [15, 2], we can use any of the $(1 - 1/e)$-approximation strategies such as the ones mentioned in [15, 17, 14, 25]. We have used TIM+ [25] in our work.

Whenever a superarm is played, we obtain a diffusion cascade s, the granularity of which is governed by the feedback mechanism M. In case of the TIC model, we obtain one diffusion s_i cascade per item. The feedback mechanism updates our estimates of $\hat{\mu}$. The edge level feedback requires the knowledge of live/dead knowledge of each edge. This means, for a given node, we know the exact neighbour who influenced it. Often, this level of detail is not available in real-life diffusions. Instead, we only know the activation sequence of influenced nodes. Under this level of information, the update step is more complicated since any of the active parents may be responsible for a given activation, leading to credit assignment problem. [26] defines a more realistic node level feedback where we just need the influenced/not-influenced status of each node along with timestep at which it was influenced.

In our work, we have used both types of feedback - Edge Level Feedback and Node Level Feedback. We discuss these in detail in the Section 5. The complete algorithm is mentioned in Algorithm 1.

Algorithm 1 CMAB Framework for IM (Assuming ϵ-Greedy)

Input: Graph $G = (V, E)$, budget k
Randomly Initialize $\hat{\mu}$

for $s = 1$ to T **do**
  Q is a boolean that is `True` with probability ϵ_s and `False` with probability $1 - \epsilon_s$
  if Q **then**
    $A =$ EXPLORE(G, k)
  else
    $A =$ EXPLOIT$(G, \hat{\mu}, TIM+, k)$
  end if
  Play superarm A and observe cascade/cascades s. Let M be the feedback mechanism.
  $\hat{\mu} =$UPDATE(s, M)
end for

4 Expectation Maximization for Learning Distributions

The UPDATE function governs how the mean distributions are updated once we know the diffusion cascade through the feedback mechanism M. This section talks about how Expectation Maximization (EM) algorithm

can be used as a substitute for the UPDATE function. We use these ideas as the basis for our update functions defined in section 5.

4.1 Offline EM

One way to infer parameters given a set of cascades is to use the Maximum Likelihood Estimation (MLE). [23] have proposed an offline EM algorithm to maximize the MLE for the IC model. Similarly, [2] have proposed an offline EM algorithm to maximize the MLE for the TIC model. Both assume that we are given a set of past diffusion cascades S. Though their algorithms assume that the diffusion cascades have information about when each node was active in the propagation, we can use the same equations for cascades where we have edge level information.

It can be seen that the offline EM algorithms can be directly used as the UPDATE function in our CMAB framework. As in each round t of the algorithm we obtain a diffusion cascade s_t, we can store all the generated cascades till round t and use all of them to get a new estimate in every round of the IM campaign in the bandit framework. That is, if we have run the bandit algorithm for t rounds, we will have $S_t = \{s_1, s_2, \ldots, s_t\}$ cascades. Then the EM algorithm could be run for these S_t cascades to update the parameters.

4.2 Online EM

Unfortunately, the time complexity of the above approach will be $O(|E|t^2)$, which is not feasible for large networks. To solve this problem, we adapt results from the online EM algorithm [19] to allow for efficient learning of probabilities. The online update will proceed as follows: Let t be the current round of the IM campaign in the bandit framework and let s_t be the cascade obtained. Let μ_{t-1} be mean estimates learned so far. We will run the expectation and maximization steps only for the new cascade s_t with estimates from the previous iteration μ_{t-1}. Let the values of the parameters obtained by running EM for just this one cascade be x_t. Then, μ_t is updated as follows:

$$\begin{aligned} \mu_t &= (1 - \eta_t)\mu_{t-1} + \eta_t x_t \\ &= \mu_{t-1} + \eta_t(x_t - \mu_{t-1}) \end{aligned} \tag{3}$$

where $\eta_t \in [0, 1]$. Intuitively, we interpolate between the values of the x_t and μ_{t-1}. This is because x_t is obtained just from cascade s_t and therefore is a bad approximation to the true estimates. Also, as this allows us to incorporate the information learned from the previous rounds of the algorithm. By using an online variant of the EM algorithm, the time complexity is reduced to $O(|E|t)$. This approach is similar to the Temporal Difference method [24] generally used in reinforcement learning.

We use the online EM formulation to obtain the UPDATE functions for IC and TIC models. The details of this are mentioned in the next section.

5 UPDATE Functions for IC and TIC

In this section we describe the formulation of the UPDATE Function for the IC and TIC models using edge and node level feedbacks. We use the online EM algorithm (Section 4.2) to obtain the equations to learn the probabilities.

5.1 Notation

We first define the requisite notations that will be used. Given a directed graph $G = (V, E)$, for any node $u \in V$, $F(u)$ is defined as the set of child nodes of u, that is the set of nodes that have an incoming edge from u. Formally,

$$F(u) = \{v \mid v \ : \ (u, v) \in E\} \tag{4}$$

$B(u)$ is defined as the set of parents nodes of u, that is the set of nodes that have an outgoing edge to u. Formally,

$$B(u) = \{v \mid v \ : \ (v, u) \in E\} \tag{5}$$

For any given cascade $s \in S$, T_s denotes the timestep at which the diffusion process stops. The set of nodes that become active at timestep r in cascade s is denoted by $D^{(s)}(r)$. The cumulative set of nodes that become active

Algorithm 2 Edge Level Update Equations for IC

Let s be the cascade obtained at round t of the bandit algorithm.
Let $p_{v,u}^{(t-1)}$ be the estimates of the probabilities learned till the previous iteration.
Let $b_{v,u}^{(t-1)}$ be a parameter for the edge (v, u) until the previous iteration.

for node $u \in V$ **do**
  for $v \in S_{u,s}^{+} \cup S_{u,s}^{-}$ **do**
    if node $v \in S_{u,s}^{+}$ **then**
      $x_{v,u} = 1$
    else
      $x_{v,u} = 0$
    end if
    $b_{v,u}^{(t)} = b_{v,u}^{(t-1)} + 1$
  end for
end for

for edge $e = (v, u) \in E$ **do**
  if $v \in S_{u,s}^{+} \cup S_{u,s}^{-}$ **then**
    $$p_{v,u}^{(t)} = p_{v,u}^{(t-1)} + \frac{1}{b_{v,u}^{(t)}} \left(x_{v,u} - p_{v,u}^{(t-1)} \right)$$
  end if
end for

by time r in s is denoted by $C^{(s)}(r)$. Formally,

$$C^{(s)}(r) = \bigcup_{r=0}^{r=T_s} D^{(s)}(r) \tag{6}$$

Let $P_u^{(s)}(r+1)$ denote the probability that node u is activated at time $r+1$ in cascade s. Let $t_u^{(s)}$ denote the timestep at which node u was activated. $t_u^{(s)} = \infty$ if node u was never influenced in cascade s.

For any node $u \in V$,

1. Let $S_{u,s}^{+}$ denote the set of nodes that could have influenced node u in cascade s.
2. Let $S_{u,s}^{-}$ denote the set of nodes that definitely did not influence node u in cascade s.

The definitions of $S_{u,s}^{+}$ and $S_{u,s}^{-}$ will change depending upon the feedback mechanism. We will use these terms in both the edge level and node level feedbacks for IC and TIC models.

5.2 UPDATE Function for IC Model

In this section, we define the UPDATE function for IC Model in the case of both edge and node level feedbacks. We adapt the update equations of the EM algorithm proposed by [23] and propose an online version using the equations in Section 4.2.

5.2.1 Edge Level Feedback for IC

In the edge level feedback mechanism, we assume that in the diffusion cascade s, we know the status of each triggered edge. That is, we know whether each triggered edge is live or dead. Therefore, we can formally define $S_{u,s}^{+}$ and $S_{u,s}^{-}$ in edge level feedback as follows:

1. $S_{u,s}^{+} = \{v \mid v \in B(u) \cap D_s(r),\ u \in D_s(r+1),\ e = (v, u) \text{ is live}\}$
   This basically refers to the node v that influenced node u. As we know the status of each edge, $|S_u^{+}| = 1$. This is because we cannot have multiple nodes influencing u in the IC model.

Algorithm 3 Node Level Update Equations for IC

Let s be the cascade obtained at round t of the bandit algorithm.
Let $p_{v,u}^{(t-1)}$ be the estimates of the probabilities learned till the previous iteration.
Let $b_{v,u}^{(t-1)}$ be a parameter for the edge (v, u) until the previous iteration.

1: **for** node $u \in V$ **do**
2: **for** $v \in S_{u,s}^{+} \cup S_{u,s}^{-}$ **do**
3: **if** node $v \in S_{u,s}^{+}$ **then**
4: $x_{v,u} = \dfrac{p_{v,u}^{(t-1)}}{P_u^{(s)}}$
5: **else**
6: $x_{v,u} = 0$
7: **end if**
8: $b_{v,u}^{(t)} = b_{v,u}^{(t-1)} + 1$
9: **end for**
10: **end for**

11: **for** edge $e = (v, u) \in E$ **do**
12: **if** $v \in S_{u,s}^{+} \cup S_{u,s}^{-}$ **then**
13: $p_{v,u}^{(t)} = p_{v,u}^{(t-1)} + \dfrac{1}{b_{v,u}^{(t)}} \left(x_{v,u} - p_{v,u}^{(t-1)} \right)$
14: **end if**
15: **end for**

2. $S_{u,s}^{-} = \{v \mid v \in B(u) \cap C_s(r),\ u \in D_s(r+1),\ e = (v, u) \text{ is not live}\}$
This basically refers to all the parent nodes v that failed to influence node u.

The edge level UPDATE function is mentioned in Algorithm 2. In each round of the bandit algorithm, we calculate $S_{u,s}^{+}$ and $S_{u,s}^{-}$ for every node $u \in V$. Then, for only those nodes for which we have some information, that is $S_{u,s}^{+} \cup S_{u,s}^{-} \neq \phi$, we calculate the new estimates (Line 2-8 of Algorithm 2). Finally, for each all the edges that have new estimates, we update the previous estimate using the online EM update mentioned in Equation 3 (Lines 11-17 of Algorithm 2).

5.2.2 Node Level Feedback for IC

As mentioned by [26], edge level feedback if often not realistic as the information about the status of each edge is not readily available. Node level feedback is a more realistic scenario where we observe that status of each node (influenced/not influenced). The disadvantage here is that for an influenced node u, we do not know which of it's active parents influenced it. Therefore, we have a credit assignment problem which was not present in the edge level scenario. In this situation, we can formally define $S_{u,s}^{+}$ and $S_{u,s}^{-}$ as follows:

1. $S_{u,s}^{+} = \{v \mid v \in B(u) \cap D_s(r),\ u \in D_s(r+1)\}$
Given that a node u is activated at timestep $r+1$ in cascade s, it has to be the case that one of its active parents in the previous timestep r influenced it.
2. $S_{u,s}^{-} = \{v \mid v \in B(u) \cap C_s(r-1),\ u \in D_s(r+1)\}$
Given that a node u is activated at timestep $r+1$ in cascade s, we can be certain that any of its parents that were active at or before timestep $r-1$ could not have influenced it.

In each round t of the bandit algorithm, let s be the cascade obtained. We compute the probability of node u being active in s as $P_u^{(s)}$ using the following equation.

$$P_u^{(s)} = 1 - \prod_{v \in S_{u,s}^{+}} (1 - p_{v,u}^{(t-1)}) \tag{7}$$

The intuition behind this is that given that node u was active at time $r+1$, the only situation in which node u isn't activated is if all the parents that were active in the previous timestep r failed to activate it. This failure probability is equal to $\prod_{v \in S^+_{u,s}} (1-\hat{p}_{v,u})$. Therefore, the probability of success is the complementary of the failure probability.

The node level UPDATE function is mentioned in Algorithm 3. In each round of the bandit algorithm, we calculate $S^+_{u,s}$ and $S^-_{u,s}$ for every node $u \in V$. Then, for only those nodes for which we have some information, that is $S^+_{u,s} \cup S^-_{u,s} \neq \phi$, we calculate the new estimates (Line 2-8 of Algorithm 3). In order to address the credit assignment problem mentioned earlier, each parent $v \in S^+_{u,s}$ is assigned a credit equal to $p^{(t-1)}_{v,u}/P^{(s)}_v$ (Line 4 of Algorithm 3). Finally, for each all the edges that have new estimates, we update the previous estimate using the online EM update mentioned in Equation 3 (Lines 11-17 of Algorithm 3).

5.3 UPDATE Function for TIC

In this section, we define the UPDATE function for IC Model in the case of both edge and node level feedbacks. We adapt the update equations of the EM algorithm proposed by [2] and propose an online version using the equations in Section 4.2.

Let I be the total number of items that are being propagated through the network. In the case of TIC, the propagation of each item is independent of all other items. Therefore, we assume that in each round of the bandit algorithm a superarm is played for each item. The consequence is that we obtain one diffusion cascade s_i for each item i in each round. Let $S = \{s_1, s_2, \ldots, s_I\}$ denote the set of cascades obtained in round t.

Let $P(s_i|z)$ denote the probability of the cascade s_i within the z^{th} component of the model ($z \in [1, K]$). Therefore,

$$P(s_i|z) = \prod_{u \in V} P^{s_i,z}_{u,+} P^{s_i,z}_{u,-} \tag{8}$$

where,

1. $P^{s_i,z}_{u,+}$ is the probability that node u was activated at time $t^{(s_i)}_u$, assuming propagation along the z^{th} topic.
2. $P^{s_i,z}_{u,-}$ is the probability that node u wasn't activated till time $t^{(s_i)}_u - 1$, assuming propagation along the z^{th} topic.

Formally,

$$P^{s_i,z}_{u,+} = 1 - \prod_{v \in S^+_{u,s_i}} (1 - p^z_{v,u}) \tag{9}$$

$$P^{s_i,z}_{u,-} = \prod_{v \in S^-_{u,s_i}} (1 - p^z_{v,u}) \tag{10}$$

Again as was the case with IC, S^+_{u,s_i} and S^-_{u,s_i} will be defined depending upon the feedback mechanism.

Let π_z be the prior probability that a generic item is assigned to topic z. This is initialized randomly ensuring that $\sum_z \pi_z = 1$. Also, for each edge $e = (v, u)$

1. Let $H^+_{v,u}$ denote the set of items i where $v \in S^+_{u,s_i}$.
2. Let $H^-_{v,u}$ denote the set of items i where $v \in S^-_{u,s_i}$.

5.3.1 Edge Level Feedback for TIC

In the edge level feedback mechanism, we assume that in the diffusion cascade s, we know the status of each triggered edge. That is, we know whether each triggered edge is live or dead. Therefore, we can formally define $S^+_{u,s}$ and $S^-_{u,s}$ in edge level feedback as follows:

1. $S^+_{u,s} = \{v \mid v \in B(u) \cap D_s(r),\ u \in D_s(r+1),\ e = (v, u) \text{ is live}\}$
   This basically refers to the node v that influenced node u. As we know the status of each edge, $|S^+_u| = 1$. This is because we cannot have multiple nodes influencing u in the TIC model.

Algorithm 4 Edge Level Update Equations for TIC

Let $S = \{s_1, s_2, \ldots, s_I\}$ be the set of cascades obtained at round t of the bandit algorithm, where s_i corresponds to the cascade for item i.
For each topic $z \in [1, K]$, let $p_{v,u}^{z,(t-1)}, \gamma_i^{z,(t-1)}, \pi_z^{(t-1)}$ be the estimates of all the parameters learned till the previous iteration. Let $\overrightarrow{p}^{z,(t-1)}$ be the set of all $p_{v,u}^{z,(t-1)}$
Let $b_{v,u}^{z,(t)}$ be a parameter for the z^{th} dimensions of edge (v, u) until round t.

for $i \in I$ **do** ▷ Expectation Step
  for $z \in [1, K]$ **do**
    $$Q_i(z) = \frac{P(s_i \mid z, \overrightarrow{p}^{z,(t-1)})\pi_z^{(t-1)}}{\sum_{\tilde{z}} P(s_i \mid \tilde{z}, \overrightarrow{p}^{z,(t-1)})\pi_{\tilde{z}}^{(t-1)}}$$
  end for
end for
for $z \in [1, K]$ **do** ▷ Maximization Step
  $$a_z = \frac{\sum_i Q_i(z)}{|S|}$$
  for $(v, u) \in E$ and $H_{v,u}^+ \cup H_{v,u}^- \neq \phi$ **do**
    $$x_{v,u}^z = \frac{\sum_{i \in H_{v,u}^+} Q_i(z)}{\sum_{i \in H_{v,u}^+} Q_i(z) + \sum_{i \in H_{v,u}^-} Q_i(z)}$$
    $$b_{v,u}^{z,(t)} = b_{v,u}^{z,(t-1)} + \sum_{i \in H_{v,u}^+} Q_i(z) + \sum_{i \in H_{v,u}^-} Q_i(z)$$
  end for
end for
▷ Online Update for Parameters
for $z \in [1, K]$ **do**
  $$\pi_z^{(t)} = \pi_z^{(t-1)} + \frac{a_z - \pi_z^{(t-1)}}{t}$$
  for $(v, u) \in E$ and $H_{v,u}^+ \cup H_{v,u}^- \neq \phi$ **do**
    $$p_{v,u}^{z,(t)} = p_{v,u}^{z,(t-1)} + \frac{\sum_{i \in H_{v,u}^+} Q_i(z) + \sum_{i \in H_{v,u}^-} Q_i(z)}{b_{v,u}^{z,(t)}} \left(x_{v,u}^z - p_{v,u}^{z,(t-1)}\right)$$
  end for
end for
for $i \in I$ **do**
  $$\gamma_z^{(t)} = \gamma_z^{(t-1)} + \frac{Q_i(z) - \gamma_z^{(t-1)}}{t}$$
end for

2. $S_{u,s}^- = \{v \mid v \in B(u) \cap C_s(r),\ u \in D_s(r+1),\ e = (v, u) \text{ is not live}\}$
   This basically refers to all the parent nodes v that failed to influence node u.

Unlike the IC model, in the TIC model even if we know the status of each edge e (live/dead), updating the mean estimate of each edge in the TIC model is not trivial. This is because each edge probability is a mixture of the latent factors of the edge topics and item distribution (Equation 2). Therefore, even though we know whether each edge was live or not, we still don't know which latent topic in the edge had a greater contribution in activating the edge. Thus, we have a credit assignment problem in the latent topic space.

The edge level UPDATE function for TIC is mentioned in Algorithm 4. Each round of the bandit algorithm proceeds in the following way:

1. We obtain S cascades, one for each item.
2. We calculate S_{u,s_i}^+ and S_{u,s_i}^- for every node $u \in V$ and each item i.
3. For each item i and each topic z, the probability of the cascade s_i ($P(s_i|z)$) is calculated using Equation 8. When calculating this probability, the estimates from the previous iteration are used.

4. For each item i and topic z we calculate the item estimate $Q_i(z)$ as shown in Line 3 of Algorithm 4. It is normalized to ensure that the values sum to 1 (which is a requirement as it is a distribution).
5. For each topic z, the probability estimate for the average item a_z is calculated as shown in Line 7 of Algorithm 4.
6. For only those edges for which we have some information, that is $H^+_{v,u} \cup H^-_{v,u} \neq \phi$, we calculate the new topic probability estimates (Line 9 of Algorithm 4). This update essentially assigns credit to each topic in the latent space, thus solving the credit assignment problem.
7. Finally, we update all the previous probabilities using the online EM update mentioned in Equation 3. The updates are mentioned in Lines 13-21 of Algorithm 4.

5.3.2 Node Level Feedback for TIC

As mentioned in Section 5.2.2, node level feedback is a more realistic scenario where we observe that status of each node (influenced/not influenced). The disadvantage here is that for an influenced node u, we do not know which of it's active parents influenced it, leading to a credit assignment problem. But, node level feedback for TIC has an additional complexity to it as there is a credit assignment problem in the latent topic space as well. This is because each edge probability is a mixture of the latent factors of the edge topics and item distribution (Equation 2). Therefore, we don't know which latent topic in the edge had a greater contribution in potentially activating the edge.

In this situation, we can formally define $S^+_{u,s}$ and $S^-_{u,s}$ as follows:

1. $S^+_{u,s} = \{v \mid v \in B(u) \cap D_s(r),\ u \in D_s(r+1)\}$
   Given that a node u is activated at timestep $r+1$ in cascade s, it has to be the case that one of its active parents in the previous timestep r influenced it.
2. $S^-_{u,s} = \{v \mid v \in B(u) \cap C_s(r-1),\ u \in D_s(r+1)\}$
   Given that a node u is activated at timestep $r+1$ in cascade s, we can be certain that any of its parents that were active at or before timestep $r-1$ could not have influenced it.

The edge level UPDATE function for TIC is mentioned in Algorithm 5. Each round of the bandit algorithm proceeds in the following way:

1. We obtain S cascades, one for each item.
2. We calculate S^+_{u,s_i} and S^-_{u,s_i} for every node $u \in V$ and each item i.
3. For each item i and each topic z, the probability of the cascade s_i ($P(s_i|z)$) is calculated using Equation 8. When calculating this probability, the estimates from the previous iteration are used.
4. For each item i and topic z we calculate the item estimate $Q_i(z)$ as shown in Line 3 of Algorithm 5. It is normalized to ensure that the values sum to 1 (which is a requirement as it is a distribution).
5. In order to address the first credit assignment problem caused due to not knowing which edge is active, for each edge $e = (v, u)$ for cascades i where $i \in H^+_{v,u}$, each parent v is assigned a credit equal to $p^{z,(t-1)}_{v,u} / P^{i,z}_{u,+}$ in the z^{th} topic (Line 5 of Algorithm 5).
6. For each topic z, the probability estimate for the average item a_z is calculated as shown in Line 10 of Algorithm 5.
7. For only those edges for which we have some information, that is $H^+_{v,u} \cup H^-_{v,u} \neq \phi$, we calculate the new topic probability estimates (Line 12 of Algorithm 5). This update essentially assigns credit to each topic in the latent space using the credit assigned for the first credit assignment problem.
8. Finally, we update all the previous probabilities using the online EM update mentioned in Equation 3. The updates are mentioned in Lines 16-24 of Algorithm 5.

6 Experiments

Our aim is to investigate the performance of our approach on real world and synthetic datasets. We test the performance of our algorithms with respect to the regret achieved and also compare the learned probabilities with the true probabilities in both IC and TIC models.

Algorithm 5 Node Level Update Equations for TIC

Let $S = \{s_1, s_2, \ldots, s_I\}$ be the set of cascades obtained at round t of the bandit algorithm, where s_i corresponds to the cascade for item i.

For each topic $z \in [1, K]$, let $p_{v,u}^{z,(t-1)}, \gamma_i^{z,(t-1)}, \pi_z^{(t-1)}$ be the estimates of all the parameters learned till the previous iteration. Let $\overrightarrow{p}^{z,(t-1)}$ be the set of all $p_{v,u}^{z,(t-1)}$

Let $b_{v,u}^{z,(t)}$ be a parameter for the z^{th} dimensions of edge (v, u) until round t.

for $i \in I$ **do** ▷ Expectation Step
  for $z \in [1, K]$ **do**
    $$Q_i(z) = \frac{P(s_i \mid z, \overrightarrow{p}^{z,(t-1)})\pi_z^{(t-1)}}{\sum_{\tilde{z}} P(s_i \mid \tilde{z}, \overrightarrow{p}^{z,(t-1)})\pi_{\tilde{z}}^{(t-1)}}$$
    for $(v, u) \in E$ and $i \in H_{v,u}^+$ **do**
      $$R_z^i(v, u) = \frac{p_{v,u}^{z,(t-1)}}{P_{u,+}^{i,z}}$$
    end for
  end for
end for
for $z \in [1, K]$ **do** ▷ Maximization Step
  $$a_z = \frac{\sum_i Q_i(z)}{|S|}$$
  for $(v, u) \in E$ and $H_{v,u}^+ \cup H_{v,u}^- \neq \phi$ **do**
    $$x_{v,u}^z = \frac{\sum_{i \in H_{v,u}^+} Q_i(z) R_z^i(v, u)}{\sum_{i \in H_{v,u}^+} Q_i(z) + \sum_{i \in H_{v,u}^-} Q_i(z)}$$
    $$b_{v,u}^{z,(t)} = b_{v,u}^{z,(t-1)} + \sum_{i \in H_{v,u}^+} Q_i(z) + \sum_{i \in H_{v,u}^-} Q_i(z)$$
  end for
end for

for $z \in [1, K]$ **do** ▷ Online Update for Parameters
  $$\pi_z^{(t)} = \pi_z^{(t-1)} + \frac{a_z - \pi_z^{(t-1)}}{t}$$
  for $(v, u) \in E$ and $H_{v,u}^+ \cup H_{v,u}^- \neq \phi$ **do**
    $$p_{v,u}^{z,(t)} = p_{v,u}^{z,(t-1)} + \frac{\sum_{i \in H_{v,u}^+} Q_i(z) + \sum_{i \in H_{v,u}^-} Q_i(z)}{b_{v,u}^{z,(t)}} \left(x_{v,u}^z - p_{v,u}^{z,(t-1)} \right)$$
  end for
end for
for $i \in I$ **do**
  $$\gamma_z^{(t)} = \gamma_z^{(t-1)} + \frac{Q_i(z) - \gamma_z^{(t-1)}}{t}$$
end for

6.1 Datasets

We have used the following datasets in our experiments.

1. **NetHEPT:** We use the NetHEPT dataset provided by [25]. This is a real undirected graph containing 15k nodes and 31k edges. We use the directed version that contains around 62k edges. The graph contains probabilities for each edge. Therefore, it can be used directly for IC experiments. For TIC, if the edge probability is p, we generate topic probabilities randomly from $[0, p]$.
2. **Synthetic:** We generate a synthetic dataset with 5k nodes and 50k edges. The graph had a uniform structure but outgoing edges of 5% of the nodes at random were assigned higher probabilities than others. For TIC, if the edge probability is p, we generate topic probabilities randomly from $[0, p]$.

3. **WikiVote:** This dataset is available in the SNAP repository [18]. The network contains all the Wikipedia voting data from the inception of Wikipedia till January 2008. The network has around 7.1k nodes and 103k edges. As this dataset doesn't provide edge probabilities, we set the probability for each edge (v, u) in proportion to the in-degree, i.e. $1/N_{in}(u)$. We perform experiments only for IC on this dataset.

6.2 Experimental Setup

The probability estimates are initialized randomly. We run the CMAB algorithm for 1000 iterations. For the purposes of this experiment, in order to simulate the diffusion in the real world, we do the following

1. For IC, we sample a deterministic graph from the probabilistic graph G in each round. We assume that the diffusion in the true world happened according to this deterministic graph.
2. For TIC, we sample a deterministic graph from the probabilistic graph G for each item in each round. We assume that the diffusion for that item in the true world happened according to this deterministic graph.

Therefore, for a seed set S, the real spread can be obtained by looking at the number of nodes reachable from S in the deterministic graph. Following [26], regret incurred in one round is defined as follows:

1. For IC, let S^* be the seed sets obtained by using the true probabilities. Let S^B be the seeds obtained using the bandit algorithm. Let $R_C(S)$ denote the spread of a seed in the deterministic graph C. Therefore, the regret can be defined as
$$Regret = R_C(S^*) - R_C(S^B)$$
2. For TIC, let S_i^* be the seed sets obtained by using the true probabilities for item i. Let S_i^B be the seeds obtained using the bandit algorithm for item i. Let $R_C(S)$ denote the spread of a seed in the deterministic graph C. Therefore, the regret for item i can be defined as
$$R_C(S_i^*) - R_C(S_i^B)$$
We then define the average regret per item as
$$Regret = \frac{1}{I}\sum_i R_C(S_i^*) - R_C(S_i^B)$$
where I is the total number of items.

We analyse the average regret, that is $\sum_{i=1}^{t} Regret/t$, where t is the number of bandit rounds.

We use TIM+[25] as our seed selection algorithm. We set the seed budget k to 50. For TIC, we use 10 items (I) and 10 Topics (K). We also analyse how these parameters affect learning. We use ϵ-Greedy as our Regret Minimization Algorithm and try 3 different parameters for ϵ:

1. **Pure Exploration:** $\epsilon = 1$
2. **Pure Exploitation:** $\epsilon = 0$
3. **ϵ-Greedy:** $\epsilon = 0.5$

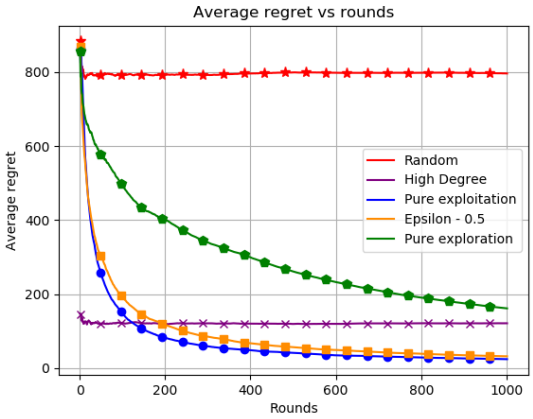
Baselines: Following [26], we use random seed selection and highest degree selection as baseline methods.

6.3 Results and Discussion

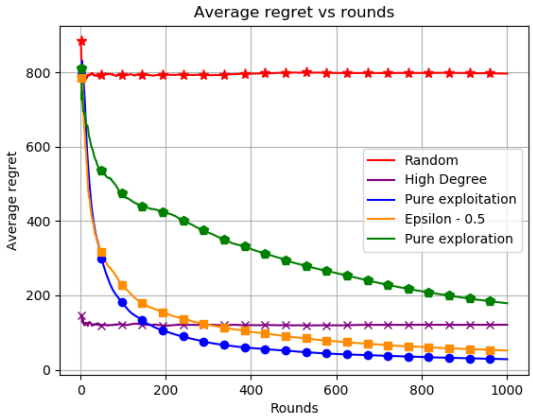
Regret Analysis: We analyse the performance of our methods in reducing the regret incurred per round. We plot the average regret (as mentioned in Section 6.2) for both edge level and node level feedbacks.

Please refer to Figure 1, Figure 5, Figure 3, Figure 2 and Figure 4. There are three observations that are common to all three results:

1. For all the datasets, the average regret for all the methods (Pure Exploration, Pure Exploitation and ϵ-Greedy) reduces quickly in the first few iterations.

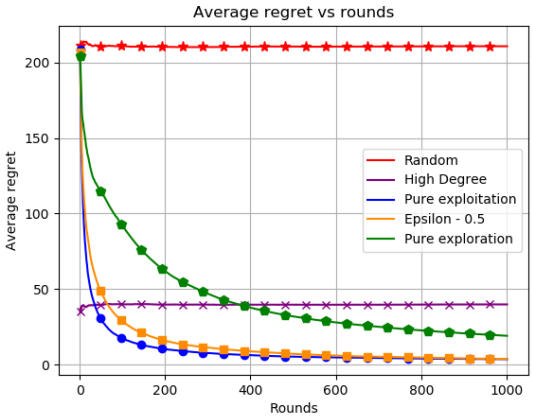


(a) NetHEPT - Regret for Edge Level Feedback

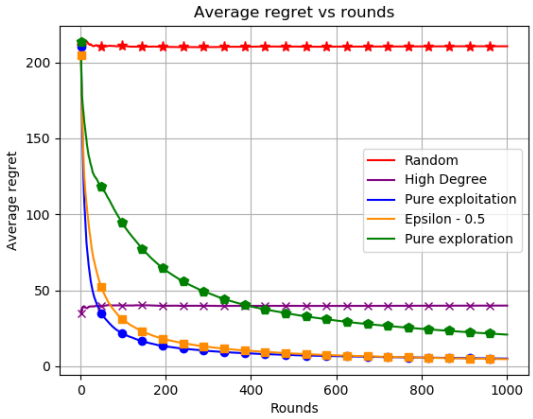


(b) NetHEPT - Regret for Node Level Feedback

Figure 1: $Regret$ vs Number of rounds for NetHEPT in IC Model, $k = 50$



(a) NetHEPT - Regret for Edge Level Feedback



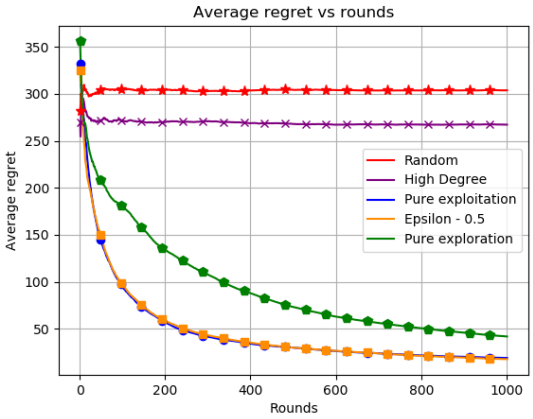
(b) NetHEPT - Regret for Node Level Feedback

Figure 2: Regret vs Number of rounds for NetHEPT in TIC Model, $k = 50$

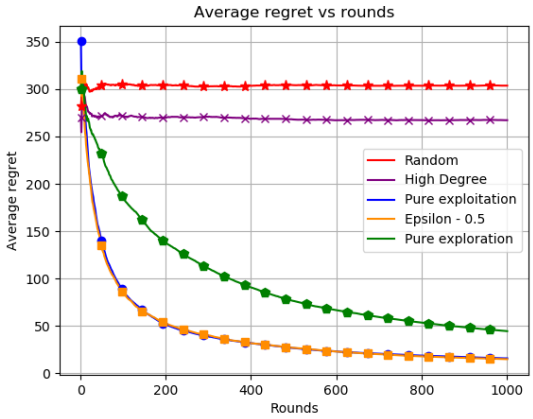
2. For all the datasets, the trend for the node level and edge level feedbacks are similar. But edge level feedback performs slightly better than node level feedback. This is expected as node level suffers from an additional credit assignment problem.
3. Initially, Pure Exploitation has the sharpest drop in regret but then saturates quickly. This is because it quickly finds a locally optimal seed set and then never deviates from it. Pure Exploration, on the other hand, has more focus towards learning probabilities as it is trying out random seeds in each iteration. Therefore, it has a slower rate of change of regret. ϵ-Greedy tries to get the best of Pure Exploration and Pure Exploitation.

The subtle differences in the results are analyzed below.

NetHEPT: Please refer to Figure 1 for IC model and 2 for TIC model. For IC model, Figure 1a and 1b refers to the average regret per round for edge level feedback and node level feedback. For TIC model, Figure 2a and 2b refers to the average regret per item per round for edge level and node level feedback. As we are initializing the probabilities without any priors, High Degree performs better initially as the seeds obtained in the other methods are poor due to lack of probability information. As the iterations proceed, Pure Exploitation and ϵ-Greedy are able to outperform the High Degree baseline.

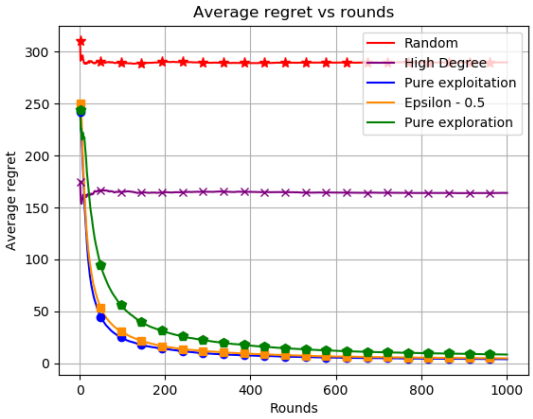


(a) Synthetic - Regret for Edge Level Feedback

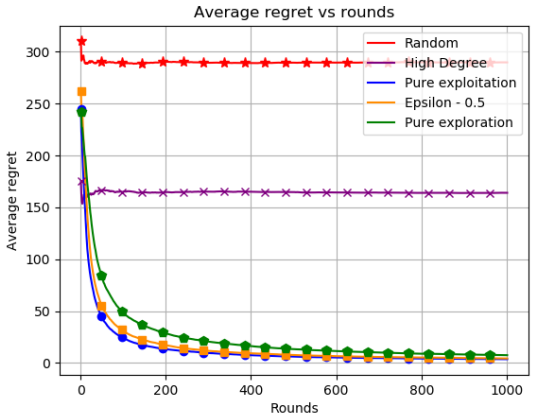


(b) Synthetic - Regret for Node Level Feedback

Figure 3: $Regret$ vs Number of rounds for Synthetic in IC Model, $k = 50$



(a) Synthetic - Regret for Edge Level Feedback



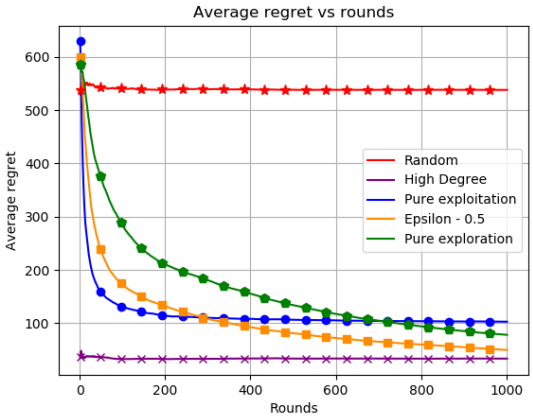
(b) Synthetic - Regret for Node Level Feedback

Figure 4: Regret vs Number of rounds for Synthetic in TIC Model, $k = 50$

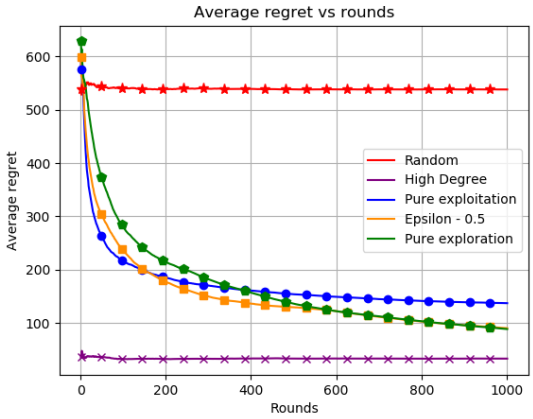
Synthetic: Please refer to Figure 3 for IC model and 4 for TIC model. For IC model, Figure 3a and 3b refers to the average regret per round for edge level feedback and node level feedback. For TIC model, Figure 4a and 4b refers to the average regret per item per round for edge level and node level feedback. As the Synthetic data was generated assuming each edge is equally likely, the graph structure is uniform - implying that the degree of all the nodes are similar. As a result, even selecting the high degree nodes leads to a high regret. But it can be seen that Pure Exploration, Pure Exploitation and ϵ-Greedy perform significantly better.

WikiVote: Please refer to Figure 5. Figure 5a and 5b refers to the average regret per round for edge level feedback and node level feedback. In this case we see that Pure Exploitation has a higher regret than Pure Exploration and ϵ-Greedy. We think this is because Pure Exploitation gets stuck in a local minima. Due to its exploration step, ϵ-Greedy is able to obtain a lower regret. Interestingly, High Degree is a really strong baseline in this dataset. This might be because in this dataset, the average degree of each node is really high (14.5). Therefore, choosing the high degree nodes might result in a lower regret.

Quality of Probabilities Learned in IC: We also test how well our algorithms are learning the true probabilities by plotting the relative L2 Error. Let E_0 be the L2 Error at iteration 0. Let E_t be the L2 Error at iteration t.

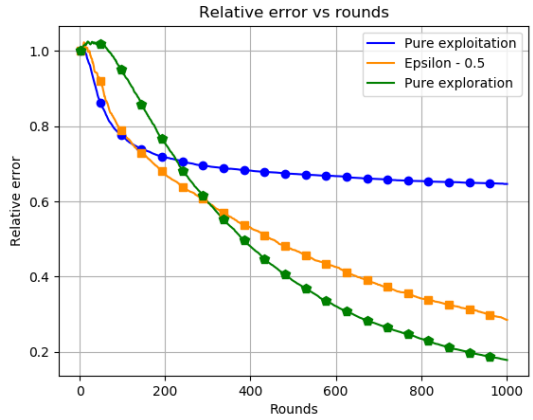


(a) WikiVotes - Regret for Edge Level Feedback

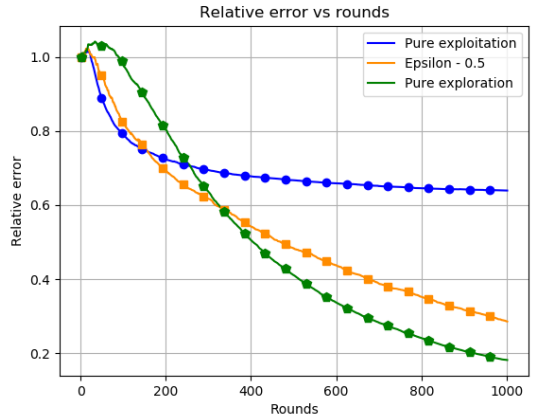


(b) WikiVotes - Regret for Node Level Feedback

Figure 5: $Regret$ vs Number of rounds for WikiVotes in IC Model, $k = 50$



(a) NetHEPT - L2 Error for Edge Level Feedback



(b) NetHEPT - L2 Error for Node Level Feedback

Figure 6: Relative L2 Error vs Number of rounds for NetHEPT in IC Model, $k = 50$

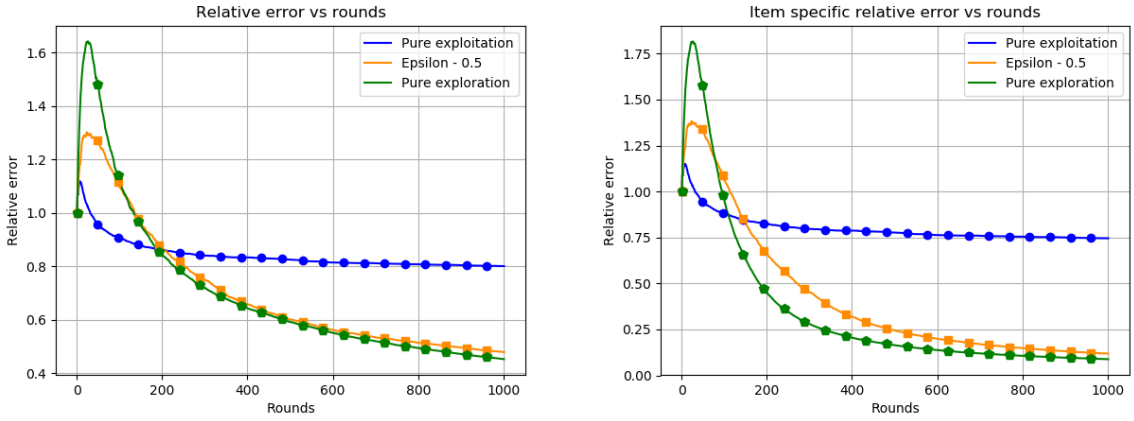
Therefore, the relative L2 Error is defined as

$$\text{L2 Error} = \frac{E_t}{E_0}$$

To reduce the number of figures, we show our results just for the NetHEPT datasaet. Please refer to Figure 6. We can see that Pure Exploration reduces the L2 error significantly. On the other hand, Pure Exploitation stops reducing the error after a point. This is because once it finds a locally optimal seed set, it learns probabilities only for the nodes/edges triggered for that seed set. ϵ-Greedy is somewhere between Pure Exploration and Pure Exploitation as it tries both EXPLORE and EXPLOIT steps.

Quality of Probabilities Learned in TIC: Since we have multiple topics and multiple items in TIC, there are two types of analysis done:

1. Relative L2 Error: This is similar to the L2 Error measure in IC. The L2 Error is calculated for each topic and then aggregated. This measure provides us with an indication of how close the learned topic probabilities are to the true topic probabilities.



(a) Relative L2 error

(b) Item Specific Relative error

Figure 7: Two types of error for NetHEPT in TIC Model, $k = 50$

2. Item Specific Relative Error: In the TIC model, the propagation of each item is dependent on the edge topic probabilities as well as the item distribution. In the real world, the propagation of an item is governed by the dot product mentioned in Equation 2. As a result, we also need to analyze whether the topic probabilities learned replicate the real world diffusions. Therefore, for each item, we convert the TIC model to an IC model by calculating edge probabilities according to Equation 2. Then for each of these item specific graphs we calculate the Relative L2 Error, which is then aggregated.

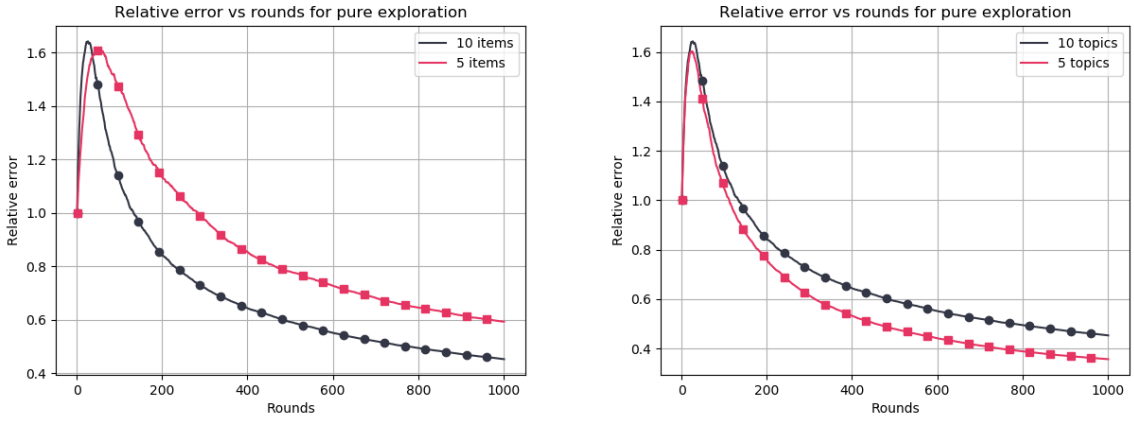
To reduce the number of figures, we show the results for these two metrics on Node Level Feedback for NetHEPT dataset.

Please refer to Figure 7a. This plot shows the relative L2 error. It can be seen that the L2 error falls for Pure Exploration, Pure Exploitation and ϵ-Greedy. Similar to the observations in IC, Pure Exploration attains the lowest relative L2 Error as it tries different seed sets. Pure Exploitation on the other hand saturates quickly as it has found a local minima.

Please refer to Figure 7b. This plot shows the item specific relative error. It can be seen that this error follows a similar trend to the relative L2 error. However, after 1000 iteration, it has a lower value as compared to the relative L2 error. This quick reduction indicates that even when the individual topic probabilities for each edge are not close to the true values, their dot product with each item's probabilistic distribution converges quickly to the true dot product values. This can be justified by using the observation that multiple vectors can have the same dot product. Due to this degree of freedom in the latent topic space, the TIC model can effectively model each item's spread in the real world even though the individual item distribution and edge topic probabilities haven't converged.

Effect of Number of Items on Learning Rate in TIC: We also analyze how the number of items affects the quality of probabilities learned. Refer to Figure 8a where we plot the relative L2 Error for NetHEPT data when the number of topics is fixed to 10 for Pure Exploration. It can be seen that the rate of decrease of L2 error is faster for 10 items as compared to 5 items. This is because more items imply more number of cascades in each iteration of the bandits algorithm, leading to more data to learn from during each iteration.

Effect of Number of Topics on Learning Rate in TIC: We also analyze the impact of the number of topics on the quality of probabilities learned. Refer to Figure 8b where we plot the relative L2 Error for NetHEPT data when the number of items is fixed to 10 for Pure Exploration. It can be seen that the rate of decrease of L2 Error is faster for 5 topics as compared to 10 topics. This is because increasing the number of topics inherently increases the complexity of the model, implying that there are more parameters to be learned. As the number of cascades obtained in each round is the same, model with less parameters (5 topics) learns quicker as compared to model with more parameters (10 topics).



(a) Relative error when items change

(b) Relative error when topics change

Figure 8: Learning rate with change in topics and items for NetHEPT in TIC Model, $k = 50$

6.3.1 Implementation Challenges and Learnings

One of the major challenges faced in the implementation of above experiments was the time consuming step of generation of seed-sets during the iterations of CMAB. Generation of seed set becomes the bottle neck in every EXPLOIT step of bandits. We tried simple greedy approach [15] and CELF++ [14] for the seed set generation before resorting to TIM+ [25] for faster implementation. Even after resorting to state of the art TIM+ method, our implementation in *Python* took hours for a single iteration of bandits on large datasets. To resolve this issue, we had to switch to a *C++* implementation. This helped lower the run time for each iteration to few seconds. Another challenge we faced was the issue of underflow as a result of product of small probabilities in large graphs. We resorted to using summing the probabilities in log scale to solve this issue.

7 Future work

We aim to test this framework on larger datasets (with millions of nodes) found in real life applications to see if it generalizes well. Since, the EXPLOIT step is quite expensive as we increase the number of nodes, there is a need to implement a parallel version of seed selection algorithm. Further, this work can be extended to other diffusion models. In particular, [2] proposed a new model AIR which has fewer parameters as compared to TIC. As [2] also proposed an EM update equation for the AIR model, our bandits framework with online EM update equations can therefore be applied. Also, all of the work till now [7, 26] has been specifically done for the IC model. It would be interesting to see how bandits perform in the case of LT models. Lastly, we are looking to try other regret minimization algorithms like Thompson sampling [12] and Upper Confidence Bound [5], and draw a comparison of regret obtained.

8 Conclusion

In this work, we addressed the problem of influence maximization when the influence probabilities in the graph are not known. In particular, we implement a learning procedure using multi-armed bandits which simultaneously learns influence probabilities and tries to achieve maximum spread. We implement an online version of expectation maximization which reduces the complexity of the MAB update step from quadratic to linear. We implement the MAB for IC and TIC models and show its efficacy by running detailed experiments. We show that both the IC and TIC MAB algorithms are pretty effective in leanring influence probabilities and achieve decent regret in just a few iterations. This shows that the MAB approach is quite effective in practical settings.

9 Acknowledgements

We would like to thank Prof. Laks V.S. Lakshmanan for his guidance and support throughout the term.

References

[1] C. Asavathiratham, S. Roy, B. Lesieutre, and G. Verghese. The influence model. *IEEE Control Systems*, 21(6):52–64, 2001.

[2] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 81–90. IEEE, 2012.

[3] L. E. Blume. The statistical mechanics of strategic interaction. *Games and economic behavior*, 5(3):387–424, 1993.

[4] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 946–957. SIAM, 2014.

[5] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, pages 151–159, 2013.

[6] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 151–159, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[7] W. Chen, Y. Wang, Y. Yuan, and Q. Wang. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *The Journal of Machine Learning Research*, 17(1):1746–1778, 2016.

[8] J. S. Coleman, E. Katz, and H. Menzel. *Medical innovation: A diffusion study*. Bobbs-Merrill Co, 1966.

[9] P. Domingos. Mining social networks for viral marketing. *IEEE Intelligent Systems*, 20(1):80–82, 2005.

[10] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM, 2001.

[11] M. Gomez Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1019–1028. ACM, 2010.

[12] A. Gopalan, S. Mannor, and Y. Mansour. Thompson sampling for complex online problems. In *International Conference on Machine Learning*, pages 100–108, 2014.

[13] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 241–250. ACM, 2010.

[14] A. Goyal, W. Lu, and L. V. Lakshmanan. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web*, pages 47–48. ACM, 2011.

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

[16] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5, 2007.

[17] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007.

[18] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[19] P. Liang and D. Klein. Online em for unsupervised models. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 611–619. Association for Computational Linguistics, 2009.

[20] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[21] P. Netrapalli and S. Sanghavi. Learning the graph of epidemic cascades. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 211–222. ACM, 2012.

[22] M. E. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[23] K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for independent cascade model. In *Knowledge-based intelligent information and engineering systems*, pages 67–75. Springer, 2008.

[24] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[25] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 75–86. ACM, 2014.

[26] S. Vaswani, L. Lakshmanan, M. Schmidt, et al. Influence maximization with bandits. *arXiv preprint arXiv:1503.00024*, 2015.