Influence Maximization with Bandits

Sharan Vaswani, Laks V.S. Lakshmanan, Mark Schmidt
University of British Columbia
{sharanv,laks,schmidtm}@cs.ubc.ca

Abstract

We consider the problem of *influence maximization* in networks, maximizing the number of people that become aware of a product by finding the 'best' set of 'seed' users to expose the product to. Most prior work on this topic assumes that we know the probability of each user influencing each other user, or we have data that lets us estimate these influences. However, this information is typically not available or is difficult to obtain. To avoid this assumption, we adopt a combinatorial multi-armed bandit paradigm that estimates the influence probabilities as we sequentially try different seed sets. We establish bounds on the performance of this procedure under the existing edge-level feedback mechanism as well as a novel and more realistic node-level feedback mechanism. Beyond our theoretical results, we describe a practical implementation and experimentally demonstrate its efficiency and effectiveness on four real datasets.

1 Introduction

Viral marketing uses social networks to spread awareness about a specific product in the market through 'word-of-mouth' influence propagation. Specifically, the marketer aims to select a fixed number of 'influential' users (called *seeds*) to give free products or discounts to. The marketer assumes that these users will influence their neighbours and, by transitivity, other users in the social network. This will result in information propagating through the network as an increasing number of users adopt or become aware of the product. The marketer typically has a budget on the number of free samples or discounts that can be given, so she must strategically choose seeds so that the maximum number of people in the network become aware of the product. The goal is to maximize the *spread* of this influence, and this is referred to as *influence maximization* (IM) [10]. In their seminal paper, Kempe et al. [10] studied the IM problem under well-known probabilistic information diffusion models including the independent cascade (IC) and linear threshold (LT) models. While the problem is NP-hard under these models, there have been numerous papers on efficient approximations and heuristic algorithms (see Section 2). But prior work on IM assumes that in addition to the network structure, we either know the pairwise user influence probabilities or that we have past propagation data from which these probabilities can be learned. However, in practice the influence probabilities are often not available or are hard to obtain. To overcome this, the initial series of papers following [10] simply *assigned* influence probabilities using some heuristic means. However, Goyal et al. [9] showed empirically that learning the influence probabilities from propagation data is critical to achieving high-quality seeds and subsequently to maximizing the spread.

In this work, we consider the common situation where propagation data is not initially available. We adopt a combinatorial multi-armed bandit (CMAB) paradigm and consider an IM campaign consisting of multiple rounds (as in another recent work [5]). Each round amounts to an IM attempt and incurs a *regret* in the influence spread because of the lack of knowledge of the influence probabilities. We seek to minimize the accumulated regret incurred by choosing suboptimal seed sets over multiple rounds. A new marketer may begin with no knowledge (other than the graph structure) and at each round we can choose seeds that improve our knowledge and/or that lead to a large spread, leading to a classic *exploration-exploitation* trade-off. As in prior work, we first consider "edge-level" feedback where we assume that we can observe whether influence propagated via each edge in the network (Section 3). We also propose a novel "node-level" feedback mechanism which is

more realistic (Section 4): it only assumes we can observe whether each node became active (e.g., adopted a product) or not, as opposed to knowing who influenced that user. We establish bounds on the regret achieved by the algorithms under both kinds of feedback mechanisms. We further present regret minimization algorithms (Section 5) and conduct extensive experiments on four real datasets to evaluate the effectiveness of the proposed algorithms (Section 6).

2 Motivation and Related Work

We model a social network as a probabilistic directed graph $G = (V, E, P)$ with nodes V representing users, edges E representing connections, and edge weights $P : E \rightarrow [0, 1]$. The influence probability $P(u, v)$ represents the probability with which user v will perform an action given that it is performed by u. A stochastic diffusion model D governs how information spreads from nodes to their neighbours in the network. Given a seed set S, the expected number of nodes influenced by S under the model D, denoted $\sigma_D(S)$ (just $\sigma(S)$ when D is obvious from context), is called the (expected) *influence spread* of S. Given G and a budget k on the number of seeds to be selected, IM aims to find the seed set S of size k which will maximize $\sigma_D(S)$

$$S^* = \underset{|S| \leq k}{\operatorname{argmax}}\, \sigma(S). \tag{1}$$

Although IM is NP-hard under standard diffusion models, the expected spread function $\sigma_D(S)$ is monotone and submodular. Solving Eq. (1) thus reduces to maximizing a submodular function under a cardinality constraint, a problem that can be solved to within a $(1 - 1/e)$-approximation using a greedy algorithm [13]. Most work on IM assumes knowledge of the influence probabilities, but there is a growing body of work on learning the influence probabilities from data. Typically, the data is a set of diffusions (also called *cascades*) that happened in the past, specified in the form of a log of actions by network users. Learning influence probabilities from available cascades has been used in discrete-time models [15, 8, 14]. However, in many real datasets the cascades are not available and we can't use these existing approaches.

In the traditional stochastic multi-armed framework [11], there are m arms each of which has an unknown reward distribution. The bandit game proceeds in rounds and in every round s, an arm is played and a corresponding reward is generated by sampling the reward distribution for that arm. This game continues for a fixed number of rounds T. The goal is to minimize the regret resulting from playing suboptimal arms across rounds.The *combinatorial* multiarmed bandit paradigm is an extension where we can pull a set of arms (a 'superarm') together [7, 4, 6]. Chen et al. [5] introduce the notion that triggering superarms can also probabilistically trigger other arms. They target both ad placement on web pages and viral marketing applications under semi-bandit feedback. They propose an algorithm based on the upper confidence bound (UCB) called combinatorial UCB (CUCB) for obtaining an optimal regret of $\mathcal{O}(\log(T))$. However, they assume the often- unrealistic "edge-level" feedback and did not experimentally test their algorithm. More recently, Lei et al. [12] studied the related, but very different, problem of maximizing the distinct number of nodes activated across rounds. They assume edge-level feedback and do not theoretically compare performance of their approach with that achievable when influence probabilities are known.

3 CMAB Framework for IM

3.1 Review of CMAB

The CMAB framework consists of m base arms. Each arm i is associated with a random variable $X_{i,s}$ which denotes the outcome (or 'reward') of triggering arm i arm on trial s. The reward $X_{i,s}$ is bounded on the support $[0, 1]$ and is independently and identically distributed according to some unknown distribution with mean μ_i. In each of the T rounds, a superarm A (a set of base arms) is played, which triggers all arms in A. In addition, some of the other arms may get probabilistically triggered. Let p^i_A denote the triggering probability of arm i if the superarm A is played (observe that $p^i_A = 1$ for $i \in A$). The reward obtained in each round s can be a (possibly non-linear) function of the rewards $X_{i,s}$ for each arm i that gets triggered in that round. Let $T_{i,s}$ denote the total number of times an arm i has been triggered at round s. For the special case of $s = T$, we use the notation $T_i := T_{i,T}$. Each time an arm i is triggered, we use the observed reward to update its mean estimate $\hat{\mu}_i$. The superarm that is expected to give the highest reward is selected in each round by an oracle O. This oracle takes as input the current mean estimates $\vec{\hat{\mu}} = (\hat{\mu}_1, ..., \hat{\mu}_m)$, and outputs an appropriate superarm A. In order to accommodate intractable problems, the framework of [4, 5] assumes that the

CMAB	Symbol	Mapping to IM
Base arm	i	Edge (u, v)
Reward for arm i in round s	$X_{i,s}$	Status (live / dead) for edge (u, v)
Mean of distribution for arm i	μ_i	Influence probability $p_(\text{u,v})$
Superarm	A	Union of outgoing edges E_S from nodes in seed set S
No. of times i is triggered in s rounds	$T_{i,s}$	No. of times u becomes active in s diffusions
Reward in round s	r_s	Spread $\bar{\sigma}$ in the s^{th} IM attempt

Table 1: Mapping of the CMAB framework to IM

oracle provides an (α, β)-approximation to the optimal solution; the oracle outputs with probability β a superarm A such that it attains an α approximation to the optimal solution.

3.2 Adaptation to IM

Though our framework is valid for any discrete time diffusion model, we will assume the IC diffusion model [10] in our discussion. At timestep $t = 0$, only the seed nodes are active. Each active node u gets one attempt to influence/activate each of its inactive out-neighbours v in the next time step. This activation attempt succeeds with influence probability $p_{u,v} := P(u, v)$. An edge along which an activation attempt succeeded is said to be *live*, and other edges are said to be *dead*. At a given time t, an inactive node v may have multiple parents which activated at time $(t - 1)$. This set of parents are capable of activating v at time t and we refer to them as the active parents of v at time t. The actual spread is the number of nodes reachable from the selected seed nodes in the real world and we denote it by $\bar{\sigma}$. Table 1 gives our mapping of the various components of the CMAB framework to IM. Note that since each edge can be either live or dead in the true diffusion, $X_{i,s} \in \{0, 1\}$ and we can assume a Bernoulli distribution on these values. We describe the CMAB framework for IM in Algorithm 1. In each round, the regret minimization algorithm $\mathcal{A}$ selects a seed set S with $|S| = k$ and plays the corresponding superarm E_S. S can be selected either randomly (EXPLORE) or by solving the IM problem with the current influence probability estimates $\vec{\hat{\mu}}$ (EXPLOIT). The details for solving Eq. (1) are encapsulated in the oracle O which takes as input the graph G and $\vec{\hat{\mu}}$, and outputs a seed set S under the cardinality constraint k. For the case of IM, O constitutes a $(1 - \frac{1}{e}, 1 - \frac{1}{|E|})$-approximation oracle [5]. Notice that the well-known greedy algorithm used for IM can serve as such an oracle. Once the superarm is played, information diffuses in the network and a subset of network edges become live which leads to a subset of nodes becoming active. The reward $X_{i,s}$ for these edges is 1. Note that the reward $\bar{\sigma}(S)$ is the number of active nodes at the end of the diffusion process and is thus a non-linear function of the rewards of the triggered arms (edges). After observing a diffusion, the mean estimate $\vec{\hat{\mu}}$ vector needs to updated. In this context, the notion of a *feedback mechanism* M plays an important role. It characterizes the information available after a superarm is played. This information is used to update the model to improve the mean estimates (UPDATE in Algorithm 1). Let S^* be the solution to Eq. (1) and let $\sigma^* = \sigma(S^*)$, the optimal expected spread. Since IM is NP-hard, even if the true influence probabilities are known, we can only hope to achieve an expected spread of $\alpha\beta\sigma^*$, where $\alpha = 1 - \frac{1}{e}$ and $\beta = 1 - \frac{1}{|E|}$. We let S_s be the seed set chosen by $\mathcal{A}$ in round s. The regret incurred by $\mathcal{A}$ is then defined by

$$Reg(\mu, \alpha, \beta) = T\alpha\beta\sigma^* - \mathbb{E}_S\big[\sum_{s=1}^{T} \bar{\sigma}(S_s)\big], \tag{2}$$

where the expectation is over the randomness in the seed sets output by the oracle. The usual feedback mechanism is the *edge-level feedback* proposed by [4], where we assume that we know the status ('live' or 'dead') of each triggered edge in the "true" possible world. The mean estimates of the arms distributions can then be updated using $\hat{\mu}_i = \frac{\sum_{s=1}^{t} X_{i,s}}{T_{i,t}}$.

4 Node-Level Feedback

Edge-level feedback is often not realistic because success/failure of activation attempts is not generally observable. Unlike the status of edges, it is quite realistic and intuitive that we can observe the status of each node: did the user buy or adopt the marketed product? While this is a more realistic assumption, the disadvantage of node-level feedback is that updating the mean estimate for each edge is more challenging. This is because we do not know which parent activated the node. This leads to a credit assignment problem, and below we describe a simple and scalable way to address it.

Algorithm 1: CMAB FRAMEWORK FOR IM(Graph $G = (V, E)$, budget k, Feedback mechanism M, Algorithm $\mathcal{A}$)

Initialize $\vec{\hat{\mu}}$;
$\forall i$ initialize $T_i = 0$;
for $s = 1 \rightarrow T$ **do**
  IS-EXPLOIT is a boolean set by algorithm $\mathcal{A}$;
  if *IS-EXPLOIT* **then**
    E_S = EXPLOIT(G,$\vec{\hat{\mu}}$,O,k)
  else
    E_S = EXPLORE(G,k)
  Play the superarm E_S and observe the diffusion cascade c ;
  $\vec{\hat{\mu}}$ = UPDATE(c,M) ;

In typical social networks, the influence probabilities are very small. Low influence probabilities cause the number of active parents to be small. We propose a scheme whereby we choose one of the active neighbours of v, say u_i, uniformly at random, and assign the credit of activating v to u_i. The probability of assigning credit to any one of K active parents is thus $\frac{1}{K}$. That is, edge (u_i, v) is given a reward of 1 whereas edges (u_j, v) corresponding to other active parents $u_j, j \neq i$, are assigned a zero reward. We then follow the same update formula as described for the edge-level feedback model. Owing to the inherent uncertainty in node-level feedback, note that we may make mistakes in credit assignment: we may infer an edge to be live while it is dead in the true world or vice versa. We call the probability of such a faulty inference the *failure probability* ρ under node-level feedback. An important question is whether we can bound this probability. This is important since failures could ultimately affect the achievable regret and the error in the learned probabilities. The result below bounds both the failure probability ρ and the difference between estimates obtained using edge-level and node-level feedback mechanisms.

Theorem 1. *Let p_{min} and p_{max} be the minimum and maximum true influence probabilities in the network. Consider a particular cascade c and any active node v with K_c active parents. The failure probability ρ under our node-level feedback credit assignment scheme for node v satsifies*

$$\rho \leq \frac{1}{K_c}(1 - p_{min})\left(1 - \prod_{k=1, k\neq i}^{K_c}[1 - p_{max}]\right) + \left(1 - \frac{1}{K_c}\right)p_{max}. \tag{3}$$

Suppose $\hat{\mu}_i^E$ and $\hat{\mu}_i^N$ are the inferred influence probabilities for the edge corresponding to arm i using edge-level and node-level feedback respectively. Then the relative error in the learned influence probability is given by:

$$\left|\frac{\hat{\mu}_i^N - \hat{\mu}_i^E}{\hat{\mu}_i^E}\right| = \rho\left|\frac{1}{\hat{\mu}_i^E} - 2\right| \tag{4}$$

We observe from Eq. 4 the error in the mean estimates increases as K_c increases, but that the heuristic will be accurate when K_c is small. In Section 6, we empirically find typical values of p_{max}, p_{min}, and K_c on real datasets and verify that the failure probability is indeed small.

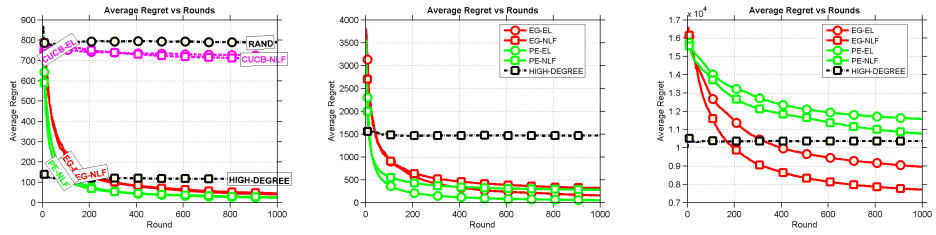
5 Regret Minimization Algorithms

As can be seen from Algorithm 1, the basic components in the framework are the EXPLORE, EXPLOIT and UPDATE subroutines. EXPLORE outputs a random subset S of size k as the seed set, whereas EXPLOIT consults the oracle O and outputs the seed set that (approximately) maximizes the spread according to current mean estimates $\vec{\hat{\mu}}$. UPDATE examines the latest cascade c and updates the parameters using the desired feedback mechanism M (either edge-level or node-level). **Upper Confidence Bound:** The Combinatorial Upper Confidence Bound (CUCB) [4] maintains an overestimate $\overline{\mu_i}$ of the mean estimates $\hat{\mu}_i$ and was theoretically shown to achieve logarithmic regret under edge-level feedback. **ϵ-Greedy:** This strategy performs exploration with probability ϵ_s and exploitation with probability $1-\epsilon_s$ and can achieve logarithmic regret under edge-level feedback [4]. We obtain node-level feedback mean estimates in terms of edge-level estimates (Theorem 1) and can use these to characterize regret. **Pure Exploitation:** This strategy performs exploitation in every round.

6 Experiments

We use 4 real datasets – Flixster, NetHEPT, Epinions and Flickr, with Flickr having the largest network (with 105K nodes and 2.3M edges). Of these, true probabilities are available for the Flixster dataset, as learned by the topic-aware IC (TIC) model [1]. Since true probabilities are not available for the other datasets, we synthetically assign them according to the weighted cascade model [10, 12, 3]: for an edge (u, v), the influence probability is set to $p_{u,v} = \frac{1}{|N^{in}(v)|}$. The probability estimates of our algorithms are initialized to 0. A 'run' of consists of playing the CMAB game for T rounds. We simulate the diffusion in the network by sampling a deterministic graph from the probabilistic graph G on each round: for the purpose of our experiments, we assume that the diffusion in the real world happened according to this sample. For our oracle, as well as for batch-mode seed selection, we use the TIM algorithm [17]. We use a maximum of $T = 1000$ rounds and use $k = 50$ (a standard choice in the IM literature). We verified that we obtain similar results for other reasonable values of k. All our results are obtained by averaging across 3 runs. We compare CUCB, ϵ-greedy (EG) and pure exploitation (PE).[1] For CUCB, if the update results in any μ_i exceeding p_{max}, as in previous works we reset it back to p_{max} [4]. We set $p_{max} = 0.2$ in our experiments, based on the fact that influence probabilities in practice tend to be small [8]. For ϵ-Greedy, we found that $\epsilon_0 = 5$ works well, and we set the exploration parameter on round s to $\epsilon_s = \epsilon_0/s$. Like in [3], we use random selection (RAND) and highest degree selection (HIGH-DEGREE) as baselines. We use 'A-M' to refer to algorithm A with feedback mechanism M. For example, EG-EL means ϵ-greedy with edge-level feedback.

We plot the average regret, $Regret(T)/T$, as the number of rounds increases. As can be seen from Figure 1(a), the average regret for both PE and EG decreases quickly with the number of rounds. This implies that at the end of 1000 rounds, the probabilities are estimated by the bandits approaches well enough that they give a comparable spread to using the seeds selected in batch mode given the true probabilities. Pure exploitation achieves the best average regret at the end of 1000 rounds. This is not uncommon for cases where the rewards are noisy [2]. Initially, with unknown probabilities, rewards are noisy in our case, so exploiting and greedily choosing the best superarm often leads to very good results. Random seed selection has the worst regret, which is constant. For the initial rounds, selecting seeds according to the high degree has a lower regret than other methods. With increasing number of rounds, the influence probabilities become more accurate and the IM oracle O outputs seed sets leading to higher spread than HIGH-DEGREE. We also observe that the regret for CUCB decreases very slowly. CUCB is biased towards exploring edges which have not been triggered often. Since typical networks contain numerous edges, CUCB ends up exploring much more than necessary and results in a slow rate of decrease in regret. We observe this behaviour for other datasets as well. We omit CUCB and RAND from further plots. We also observe the regret decreases quickly under node-level feedback and is close to that obtained with edge-level feedback. For NetHEPT with $k = 50$, the average number of active parents for a node is 1.175. Previous work has shown that the probabilities learned from diffusion cascades are generally small [16, 8, 14]. For example, if $p_{min} = 0$ and p_{max} varies from 0.01 to 0.2, the failure probability ρ (calculated according to the equation 3) varies from 0.0115 to 0.2261. This is true for all our datasets. For other



(a) NetHEPT (b) Flixster (c) Flickr

Figure 1: Regret vs number of rounds for different algorithms

datasets too, the regret using node-level feedback is close to that obtained using edge-level feedback mechanism. For Flixster (similar results are obtained for Epinions), NL is effective for all regret minimization algorithms with PE obtaining the lowest regret. For Flickr, because of the large size

[1] Abbreviations used in plots are in parentheses.

of the graph, it is challenging to find a good seed set with partially learned probabilities. As a result, the average regret after 1000 rounds is higher than for other datasets. We observe that while PE finds a locally optimal seed set. EG is able to find a much better seed set and consequently converges to a much lower because of its exploration phase.

7 Conclusion

We studied the problem of influence maximization when no influence probabilities or diffusion cascades are available. We adopted a combinatorial multi-armed bandit paradigm and used algorithms from the bandits literature to minimize the loss in spread due to lack of knowledge of influence probabilities. We also evaluated their empirical performance on four real datasets.

References

[1] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. *Knowledge and information systems*, 37(3):555–584, 2013.

[2] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[3] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038. ACM, 2010.

[4] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the 30th International Conference on Machine Learning*, pages 151–159, 2013.

[5] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *arXiv preprint arXiv:1407.8339*, 2014.

[6] Y. Gai, B. Krishnamachari, and R. Jain. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*, pages 1–9. IEEE, 2010.

[7] Y. Gai, B. Krishnamachari, and R. Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1466–1478, 2012.

[8] 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.

[9] A. Goyal, F. Bonchi, and L. V. Lakshmanan. A data-based approach to social influence maximization. *Proceedings of the VLDB Endowment*, 5(1):73–84, 2011.

[10] 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.

[11] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[12] S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart. Online influence maximization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 645–654, 2015.

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

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

[15] 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.

[16] K. Saito, K. Ohara, Y. Yamagishi, M. Kimura, and H. Motoda. Learning diffusion probability based on node attributes in social networks. In *Foundations of Intelligent Systems*, pages 153–162. Springer, 2011.

[17] Y. Tang, X. Xiao, and S. Yanchen. Influence maximization: Near-optimal time complexity meets practical efficiency. 2014.