Influence Maximization with Spontaneous User Adoption

背景

传统的影响力最大化即IM问题是在独立级联和线性阈值的传播模型基础上，寻找k个种子节点，使其最终影响节点的期望数量最大化。若仅考虑独立级联即IC模型，它考虑的是种子节点在一次传播中无延迟的概率的影响它的邻居节点，而不考虑节点本身的特点。而真实情境下，节点自身是有可能自发的激活的，这是不受外界干扰的主动行为，而且节点受到影响（无论来自邻居节点还是自身）是有一定延迟时间的，本文将这两点加入到IM问题中，提出了新的问题和优化目标。

解决问题

- 解决了考虑到节点自发激活后的影响力传播最大化问题
- 解决了考虑到影响延迟后的影响力传播最大化问题

创新之处

- 提出了新的传播模型：结合节点自激活后的独立级联模型——SAIC模型。
- 在SAIC上研究了三个影响力最大化问题：BIM、PIM、BPIM并给出了相应的解决方法。

解决方法

SAIC

自激活概率和延迟

在社交网络$G=(V,E)$，对应$\forall u \in V$，u拥有自激活概率$q(u) \in [0,1]$，如果u是自激活的，那么它将在随机延迟$\delta(u) \in [0,+\infty)$后被激活，$\delta(u) \sim \Delta(u)$；如果$u$是被邻居节点$v$激活，那么这个激活将在随机传播延迟$d(u,v) \sim D(u,v)$后发生。

SAIC

1. 对应节点$u \in V$以概率$q(u)$发生自激活，若发生，则延迟$\delta(u)$后被激活，除非在$\delta(u)$之前被其它节点激活。
2. 对应节点$u \in V$在时间t被激活，无论是自激活还是被邻居激活，它都会尝试按概率$p(u,v)$对所有出邻居v进行一次激活。如果成功，那么在传播延迟$d(u,v)$后，v在$t+d(u,v)$时刻被激活。
3. 节点v被激活后，激活状态将一直保持

4. 对与种子节点$u \in S$，它的自激活概率被提升到1，确保被激活，激活时间是$\delta(u)$。

自激活影响最大化

Boosted Influence Maximization

BIM和传统的IM问题接近，它的目标是选择k个种子节点最大化总的影响传播，它将种子节点的自激活概率提升为1。总的提升影响传播计算包括被种子节点激活以及通过自激活节点激活的节点。

假设其目标函数为σ^B，该目标函数满足子模性和单调性。

作者用IMM-BIM算法来解决这个问题。这个算法分两部分：（1）生成θ个RR集合（2）通过贪心算法寻找尽可能覆盖多的RR集合的k个种子节点。在第一步中，应用IMM框架来计算最优解的下界来得到θ，对于一个RR集合R，其中每个节点都进行抛硬币的方式判断是否存在自激活，若存在则直接让变量$cover$+1，R无需再被计算让种子节点覆盖。只有不存在自激活节点的RR集合，才需通过贪心节点选择算法，选取种子节点集合S。

可以将σ^B估计为S或者自激活节点覆盖RR集合的一个分式的n倍：

$$F_{\mathcal{R}}^{S}(S)=\frac{covered+\sum_{R\in\mathcal{R}}\mathbb{I}\{S\cap R\neq\emptyset\}}{covered+|\mathcal{R}|}$$

IMM-BIM满足$1-1/e-\epsilon$的近似，时间复杂度为$O((k+l)(n+m)logn/\epsilon^2)$

Preemptive Influence Spread and Boosted Preemptive Influence Spread

在SAIC模型中，可以将经常容易发生自激活的节点并且它的影响能首先到达许多其它节点（在其它自激活节点影响到达之前）看作为$organic\ influencer$。基于这个说法，将集合A的抢先影响传播$\rho(A)$定义为一些节点u($u \in A\ and\ u\ is\ self-activated$)首先到达的节点的期望数量。

假设其目标函数为$\rho(\cdot)$，它满足可加性。

PIM目标在于选择k个拥有最大的抢先影响传播的节点。

BPIM的目标函数$\rho^B(S)$是将种子集合S的自激活概率提升为1之后所产生的抢先影响传播。目的在于寻找k个种子节点，最大化其抢先影响传播。它的目标函数满足单调性和子模性。

IMM-BPIM算法用来解决BPIM问题，它和IMM-BIM不同之处在于RR集合的定义和生成过程。假如P-RR集合的根节点为v，那么集合中的节点u满足（1）u可以通过活边到达v（2）u到达v的总延迟小于等于其它自激活节点到达v的延迟的最小值。

P-RR集合生成算法根据Dijkstra‘s最短路径算法思想，生成到达根节点v最小延迟的自激活节点的集合u^s和P-RR集合R^P。然后通过IMM算法来求解S。

其最优解满足$1-1/e-\epsilon$的近似，时间复杂度为$O((k+l)(n+m)log^2n/\epsilon^2)$.

IMM-PIM和IMM-BIM一样，都用了IMM框架，但是没有贪心选择节点算法这部分，因为它没有种子节点的说法。该算法返回了$1-1/e$的近似解，时间复杂度为$O((k+l)(n+m)log^2n/\epsilon^2)$.

实验

DataSets

- Flixster
- NetHEPT
- DBLP

Algorithms

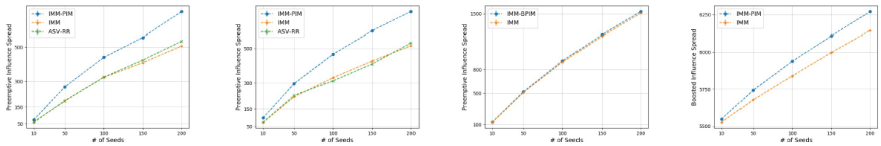
- IMM（PIM,BPIM） 即不考虑自激活和传播延迟为0
- ASV-RR 将所有节点看作为在均匀随机顺序下的自激活
- 文中的算法（IMM-BIM,IMM-PIM,IMM-BPIM）

对于自激活概率$q(u)$考虑五种情况：

1. 均匀的，$q(u)=\alpha_u$
2. 与u的出度$d^+(u)$成正相关
3. 与u的出度$d^+(u)$成负相关
4. 随机1和2设置，对半分
5. 随机1和3设置，对半分

Results

影响传播结果：



(a) PIM-NetHEPT (b) PIM-Flixster (c) BPIM-NetHEPT (d) BPIM-NetHEPT

Figure 1: Influence Spread Results ($\varepsilon = 0.1$ with case 3).

对于图1和附件表格，IMM-BIM综合表现最好，其次IMM-PIM在抢先影响传播上要高于设置的基准IMM和ASV-RR。IMM-BPIM与IMM结果相近，在提升影响传播上IMM-PIM要高于基准IMM。

运行时间：

Table 1: Running time results (in seconds).

Data	IMM-BIM	IMM-PIM	IMM-BPIM	IMM	ASV-RR
NetHEPT	0.7534	195.75	48.123	1.9712	74.175

Data	IMM-BIM	IMM-PIM	IMM-BPIM	IMM	ASV-RR
Flixster	1.3516	955.45	218.51	5.1072	235.34

在两个数据集上，运行时间 IMM-BIM < IMM<IMM-BPIM < ASV-RR < IMM-PIM。IMM-BPIM和IMM-BIM应用Dijkstra-like的逆向传播，因此运行时间较慢。