Profit Maximization for Viral Marketing in Online Social Networks

背景

信息可以通过说词的形式在在线社交网络中迅速传播，病毒式营销就是一个典例。如何选取种子集合，从而使得影响传播最大化是一个研究广泛的问题，这个问题的影响函数在典型的模型中(独立级联、线性阈值...)是满足子模性和单调性的。

然而有时候关心的是利润最大化的问题，选取种子集合需要一定的开销，而通过影响传播来获得收益，需要在这两者之间做一个平衡，通过一定的选取种子集合的策略来使利润最大化，而利润函数被证明为满足子规模性却不单调。

因此在病毒式营销问题上研究利润最大化要比研究影响传播最大化更具有挑战性。

解决的具体问题

- 解决了贪心爬山算法不提供任何界限近似保证的问题
- 解决了双贪心算法在特殊情况下表现任意坏的问题
- 为利润最大化问题的最优解提供了几个在线上界

解决方法

The Profit Maximization Problem

种子集合S的影响传播函数为$\sigma(s)$, 其成本为$c(S)$, 则其利益函数定义为：

$$\phi(S) = \sigma(S) - c(S)$$

目标是找到一个种子集合S，使得利润$\phi(S)$最大化。由于$\sigma(S)$是满足子模性的，很容易证得$\phi(S)$也是子模性的。对于增加一个新的种子节点$v \in V$，将其边界影响增益定义为：

$$\phi(v|S) = \phi(S \cup \{v\}) - \phi(S)$$

虽然在经典模型下$\sigma(\cdot)$和$\phi(\cdot)$都是满足子模性的，但$\sigma(\cdot)$是满足单调性的，而$\phi(\cdot)$被证明是不单调的，因此利润最大化问题是子模非单调问题。

Seed Selection Algorithm

该部分阐述了作者对于上述问题的核心解决方案，基于双贪心算法，改进该算法在$\phi(V) < 0$情况下表现任意坏的情况，并通过迭代剪枝的方式减小最优解的搜索空间。

双贪心算法解决了贪心爬山算法的最优解没有强的近似保证，对于$\phi(V) \geq 0$，确定性的双贪心算法提供了1/3的近似保证。随机的双贪心算法提供了1/2的近似保证。但前提条件都是非负子模函数$\phi(V) \geq 0$，在实际情况下，要满足这样的条件并不现实，因此作者利用了双贪心算法提供的界限近似保证，改进了其只适应非负子模函数的缺点。

- Warm-Start by Iterative Pruning

  为了解决双贪心算法存在的问题，作者使用迭代剪枝技术来减小最大化利润函数的搜索空间：从V的幂集降到一个小的区间。

  首先定义了两个节点集合：$\mathcal{A}_1 = \{v : \phi(v|\mathcal{V}\backslash\{v\}) > 0\}$,
  $\mathcal{B}_1 = \{v : \phi(v|\phi) \geq 0\}$,让区间$\mathcal{L}_1 = [\mathcal{A}_1, \mathcal{B}_1]$.

  对于全局最优解$\mathcal{S}^*$满足$\mathcal{A}_1 \subseteq \mathcal{S}^* \subseteq \mathcal{B}_1 \quad \phi(\mathcal{S}^*) = max_{\mathcal{S}\subseteq\mathcal{V}}\phi(\mathcal{S})$.

  然后通过迭代策略对$\mathcal{L}_1$进行剪枝，由于在$\mathcal{A}_1$中的节点一定被包含在全局最优解中，因此把$\mathcal{B}_1$缩剪到$\mathcal{B}_2 = \{v : \phi(v|\mathcal{A}) \geq 0\}$；相同的，由于在$V\backslash\mathcal{B}_1$中的节点一定不包含在全局最优解中，因此把$\mathcal{A}_1$扩展到$\mathcal{A}_2 = \{v : \phi(v|\mathcal{B}_1\backslash\{v\}) > 0\}$，则返回了一个更小的区间$\mathcal{L}_2 = [\mathcal{A}_2, \mathcal{B}_2]$。循环迭代可以得到区间$\mathcal{L}^* = [\mathcal{A}^*, \mathcal{B}^*]$使得$\mathcal{A}^* \subseteq \mathcal{S}^* \subseteq \mathcal{B}^*$.

  通过迭代剪枝得到的$\mathcal{A}^*$和$\mathcal{B}^*$可以用来做贪心算法的热启动，只有在$\mathcal{B}^*\backslash\mathcal{A}^*$的节点才能被测试是否加入种子集。

- Online Upper Bounds

  对于确定和随机版本的算法，可以得到两个上界：

  $$\mu_1 \triangleq 3 \cdot \phi(\hat{S}_D) - [\phi(\mathcal{A}^*) + \phi(\mathcal{B}^*)] \geq \phi(\mathcal{S}^*)$$
  $$\mu_2 \triangleq 2 \cdot \mathbb{E}[\phi(\hat{\mathcal{S}}_\mathcal{R})] - [\phi(\mathcal{A}^*) + \phi(\mathcal{B}^*)] \geq \phi(\mathcal{S}^*)$$

- Time Complexity

  将u加入S或者将u剔除T的复杂度为$O(M)$，简单贪心算法选择一个节点复杂度为$O((|V| - |S|)M)$，所以总的复杂度为$O((|V| + |V| - 1 + \ldots + 1)M) = O(|V|^2 M)$，双贪心算法检查每个节点选取的复杂度为$O(2M)$，所以其总的复杂度为$O(|V|2M)$（因为双贪心算法没有每次选取最大种子集合的过程，使其复杂度大大降低）。

数据集

Dataset	#Nodes ($\|\mathcal{V}\|$)	#Edges ($\|\mathcal{E}\|$)	Avg. degree
Facebook	4K	176K	43.7
Wiki-Vote	7K	10K	14.6
Google+	108K	14M	127.1
LiveJournal	5M	69M	14.2

实验分析

- 算法
  - Random
  - High Degree
  - IMM
  - Simple Greedy
  - Simple Greedy with Iterative Pruning

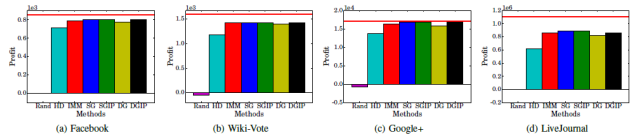
- Double Greedy
- Double Greedy with Iterative Pruning

- 参数设置

  对于独立级联模型，将p_{uv}设置为v入度的倒数，即$p_{uv} = 1/|\mathcal{I}_v|$;种子集合所有节点成本通过一个规模参数$\lambda$来规定，即$\sum_{v \in V} c(v) = \lambda \cdot |V|$.λ越大，种子节点选择的成本越高，默认值为100.

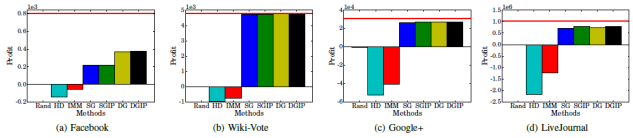
- 结果

  - 在不同成本分布下各个算法的利润



(a) Facebook (b) Wiki-Vote (c) Google+ (d) LiveJournal

Figure 3. Profits produced by different algorithms under uniform cost distribution.



(a) Facebook (b) Wiki-Vote (c) Google+ (d) LiveJournal

Figure 4. Profits produced by different algorithms under degree-proportional cost distribution.

在均匀成本分配下，各个算法在不同数据集上获得的利润，几个贪心算法的效果都非常相近。迭代剪枝的贪心算法要比原贪心算法效果稍好一点。

在比例成本分配下，总体来说，双贪心算法要比简单贪心算法效果好，迭代剪枝技术的贪心算法效果更好。

  - 不同算法的运行时间

Table II
RUNNING TIMES OF DIFFERENT ALGORITHMS (SECONDS).

Dataset	IMM	SG	SGIP	DG	DGIP
Facebook	1.28	0.21	0.24	0.22	0.24
Wiki-Vote	0.91	0.14	0.10	0.19	0.10
Google+	31.02	6.02	7.09	5.72	5.98
LiveJournal	3273.63	550.91	588.66	545.40	587.12

(a) Uniform Cost Distribution

Dataset	IMM	SG	SGIP	DG	DGIP
Facebook	2.12	0.25	0.25	0.22	0.24
Wiki-Vote	1.10	0.08	0.09	0.08	0.09
Google+	29.42	6.87	6.53	6.94	6.55
LiveJournal	2756.32	563.16	581.76	555.90	582.53

(b) Degree-Proportional Cost Distribution

两种成本分配方式下，迭代剪枝技术的贪心算法与原贪心算法有相近的运行时间，但都比IMM算法快很多。

  - 迭代剪枝技术的影响

Table III
IMPACT OF ITERATIVE PRUNING.

Dataset	$\lvert\mathcal{A}^*\rvert$	$\lvert\mathcal{B}^*\rvert$	$\lvert\mathcal{B}^* \setminus \mathcal{A}^*\rvert$	$\phi(\mathcal{A}^*) + \phi(\mathcal{B}^*)$
Facebook	12	158	146	622
Wiki-Vote	54	241	187	2,104
Google+	715	856	141	33,624
LiveJournal	2,719	548,855	546,136	-2,460,900

(a) Uniform Cost Distribution

	$\lvert\mathcal{A}^*\rvert$	$\lvert\mathcal{B}^*\rvert$	$\lvert\mathcal{B}^* \setminus \mathcal{A}^*\rvert$	$\phi(\mathcal{A}^*) + \phi(\mathcal{B}^*)$
Facebook	53	2,589	2,536	-8,678
Wiki-Vote	4,808	4,808	0	9,537
Google+	36,070	43,062	6,992	54,947
LiveJournal	1,136,106	1,738,332	602,226	1,372,225

(b) Degree-Proportional Cost Distribution

从中可以看到迭代剪枝技术降低了在各个数据集上的搜索空间。

- 在线上界

Table IV
NORMALIZED ONLINE UPPER BOUNDS.

Dataset	μ_1^{DG}	μ_3^{DG}	μ_4^{DG}	μ_1^{DGIP}	μ_3^{DGIP}	μ_4^{DGIP}
Facebook	49.79	1.21	8.80	2.26	**1.07**	1.25
Wiki-Vote	49.00	1.63	3.94	1.62	**1.12**	1.28
Google+	64.45	1.37	3.50	1.05	**1.02**	1.03
LiveJournal	56.44	1.51	26.21	5.87	**1.28**	5.64

(a) Uniform Cost Distribution

Dataset	μ_1^{DG}	μ_3^{DG}	μ_4^{DG}	μ_1^{DGIP}	μ_3^{DGIP}	μ_4^{DGIP}
Facebook	101.99	2.31	12.76	27.05	**2.15**	11.22
Wiki-Vote	16.45	1.00	1.02	1.00	**1.00**	1.00
Google+	39.17	1.34	1.59	1.30	**1.14**	1.20
LiveJournal	63.30	2.30	7.89	1.70	**1.31**	1.31

(b) Degree-Proportional Cost Distribution

这些界限被DGIP算法的实际收益标准化以后，对所有测试的数据集，DG的μ_1界限很松，μ_3界限较紧。可以看出，DGIP可以明显的改善界限。

其它解决方法

- Influence Maximization

  用来适应于离散最优问题，目的是找到一个固定大小的种子集合使其有最大影响传播。起先是用贪心算法，后来两个方向改进：启发式和蒙特卡罗模拟。这些工作都是用于影响最大化的，这和利润最大化有很大不同，利润最大化要同时兼顾影响收益和种子集合选取成本。

- Viral Marketing

  对于病毒营销问题，之前工作的研究如下：

  - 计算用户最终购买产品的期望比例
  - 通过寻找最佳营销策略在不同的用户之间控制价格和销售顺序以提高收益
  - 在尽可能开销小的情况下匹配客户与广告服务商
  - 简单贪心算法和双贪心算法，聚焦于设计策略最优化病毒营销的利润。

- Unconstrained Submodular Maximization

  双贪心算法提高了结果的近似保证，前人在子模最小化中给出了两个膜上界，作者借此最先进的算法和思想，改善了双贪心算法搜索空间大和特殊情况下表现任意坏的问题，并得到了几个在线上界作为利益优化算法性能的基准。

创新之处

- 为双贪心算法开发了一种迭代剪枝技术，这种方式提供了好的热启动和强调近似保证。
- 得到了几个可以通过任何算法生成的解的质量的在线界限，可以被用来评估算法在特殊问题实例上的实际表现性能。
- 所有的实验建立在真实的在线社交网络数据集上。