Notes on Social and Behavioral Networks

Cristian Di Pietrantonio

October 13, 2017

Contents

Chapter 1

Meme flow

On the web, information is published by one or more *sources* and often spreads quickly to other parties. The most popular kind of information that spreads quickly is a meme. From Wikipedia:

> A meme is an idea, behavior, or style that spreads from person to person within a culture - often with the aim of conveying a particular phenomenon, theme or meaning represented by the meme.

In this chapter we are going to define a mathematical model that describes the spread of information. We call *node* an entity on the web that holds the information of interest; it can be a blog, a website or a social network profile. Nodes can receive or take information from other nodes, enabling the spread of the meme. Nodes can also publish the same meme independently (without "copying" each other). Whenever a node publishes information, a timestamp of the event is also available. At the end, considering a some particular meme, all we can see is a bunch of nodes having published with an associated timestamp. What we would like to know is the *social graph* that links these nodes, i.e., who retrieves information from who.

1.1 The Trace Spreading Model

Authors in [3] introduce the Trace Spreading Model, where a *trace* is a sequence of nodes ordered by their timestamp.

Define an undirected graph $G = (V, E)$, where:

- V is the set of nodes that holds the information of interest;
- There is only a source of information, chosen uniformly at random;
- $e = \{v, v'\} \in E$ iif information propagated from v to v', $v, v' \in V$ (the actual direction is not important, as you will notice later).

- Each edge $e = \{v, v'\}$ has a *weight* $w(e)$ sampled i.i.d. in $Exp(\lambda)$, which represent the time needed for the information to propagate from v to v'. In fact, for our purpose, any distribution will do (i.i.d. is the important part).

In this model, the trace follows the shortest path tree from the source.

So, having this model and a set of traces (and every trace contains all the nodes), is it possible to reconstruct the unknown social graph G?

1.2 The First-Edge Algorithm

Authors in [1] shows that we can reconstruct (in most cases) the graph $G = (V, E)$ with high probability with a simple algorithm.

```
Input: $T$, the set of traces
$E \leftarrow \{\}$ //the set of edges in the graph ;
for $t \in T$ do
    $u \leftarrow t[0]$ //the first node in t;
    $v \leftarrow t[1]$ //the second node in t;
    $E \leftarrow E \cup \{(u, v)\}$;
end
```

Algorithm 1: The first-edge algorithm.

Consider Algorithm 1. What it does is simply adding an edge between the first and second node in a trace, for every trace. That is because we can only be sure about the existence of that edge. The first node in a trace is the source (for that trace), since it has the lowest timestamp; the second node, which has the second lowest timestamp, could only get the information from the first node.

Theorem 1. *If there are $|T| \geq cn\Delta \log n$ traces, the first-edge algorithm can reconstruct the unknown graph $G = (V, E)$ with probability $p \geq 1 - \frac{1}{n^{2(c-1)}}$, where $n = |V|$, $\Delta = \max_v \deg(\text{v})$ and c is a constant.*

Proof. An edge $\{u, v\}$ will be in the graph if and only if u and v appears at the beginning of at least one trace. We will prove that it happens with high probability.

Consider the following event.

$\xi_1 =$ "A given edge $\{u, v\} \in E$ is inferred using the trace t."

The probability of ξ_1 is given by

$$\begin{aligned} Pr\{\xi_1\} &= Pr\{t[0] = x\} Pr\{t[1] = y, y \in \{u, v\} \setminus \{x\} \mid t[0] = x\},\ x \in \{u, v\} && (1.1) \\ &= \frac{2}{n} \frac{1}{\mathit{deg(x)}}, x \in \{u, v\} && (1.2) \\ &\geq \frac{2}{n\Delta}. && (1.3) \end{aligned}$$

In other words it is the probability of one of the two nodes u, v to appear as first node in the trace multiplied by the probability that the other appears as second node in the trace. Since the source is selected uniformly at random, every node has the same probability $\frac{1}{n}$ to be picked; for us, either u or v will do. Next, assume without loss of generality (wlog) that u is picked as first node. The second node will now be drawn from the neighbors of u (convince yourself that there can't be a node y after the first node in the trace without it being one of its neighbors). The probability that the neighbor picked is u is $\frac{1}{deg(u)}$ that can be overestimated using Δ.

Now, the complementary event of ξ_1 is "the edge $\{u, v\}$ is not inferred using trace t" and its probability is $(1 - \frac{2}{n\Delta})$. For the edge not to be inferred at all, there should be no trace from which that edge can be extracted.

$$Pr\{\textit{An edge in the graph is not inferred}\} = \left(1 - \frac{2}{n\Delta}\right)^{|T|} \leq \left(1 - \frac{2}{n\Delta}\right)^{cn\Delta \log n} \tag{1.4}$$

We will show that this probability is tiny. To do so, we must use the following lemma.

Lemma 1.

$$1 - x \leq e^{-x} \tag{1.5}$$

Proof. Studying the function we see that

$$e^{-x} = (-e^{-x})' = (e^{-x})'' > 0$$

so its tangent in every point is below the function. Taking the tangent in $x_0 = 0$ we have

$$\begin{aligned} e^{-x} &\geq (e^{-x_0})'x + e^{-x_0} \\ &= -e^{-x_0} + 1 \\ &= -x + 1 \\ &= 1 - x. \end{aligned}$$

□

Going back to our proof we have

$$\begin{aligned} Pr\{\textit{Edge } \{u,\ v\} \textit{ in the graph is not inferred}\} &= \left(1 - \frac{2}{n\Delta}\right)^{|T|} && (1.6) \\ &\leq \left(1 - \frac{2}{n\Delta}\right)^{cn\Delta \log n} && (1.7) \\ &\leq \left(e^{-\frac{2}{n\Delta}}\right)^{cn\Delta \log n} && (1.8) \\ &= e^{\log n^{-2c}} && (1.9) \\ &= \frac{1}{n^{2c}}. && (1.10) \end{aligned}$$

Finally, we want to compute the probability that the algorithm fails. Let

$$\xi_{a,b} = \text{“}\textit{Edge } \{a,\ b\} \textit{ is not inferred by the algorithm”}.$$

Then

$$\begin{aligned}
Pr\{\textit{The algorithm fails}\} &= Pr\left\{ \bigcup_{\{a,b\}\in E} \xi_{a,b} \right\} && (1.11)\\
&\leq \sum_{\{a,b\}\in E} Pr\{\xi_{a,b}\} && \text{(by union bound)}\\
&= |E| Pr\{\xi_{a,b}\} && (1.12)\\
&\leq n^2 \frac{1}{n^{2c}} && (1.13)\\
&= \frac{1}{n^{2(c-1)}}. && (1.14)
\end{aligned}$$

Notice that we can make this probability tiny as we want by increasing c, i.e. by increasing the number of traces. □

Chapter 2

Chernoff Bound

In this Chapter we see a way to bound tail distributions, that is, the probability that a random variable assumes values that are far from its expectation.

We state (without proof) the Chernoff Bound, which is a powerful tool to bound tail distribution exponentially decreasing values.

Theorem 2 (Chernoff Bound)**.** *Let $X_1, X_2, \ldots, X_n$, $0 \leq X_i \leq 1, \forall i$, be a set of mutually independent random variables, and $X = \sum_{i=1}^{n} X_i$. Then*

$$\Pr\{X \leq (1-\varepsilon)E[X]\} \leq e^{-\frac{\varepsilon^2 E[X]}{3}} \tag{2.1}$$

and

$$\Pr\{X \geq (1+\varepsilon)E[X]\} \leq e^{-\frac{\varepsilon^2 E[X]}{3}}. \tag{2.2}$$

Consider the following situation. We flip n fair coins. Define

$$X_i = \begin{cases} 1 \textit{ if the i-th flip results in heads} \\ 0 \textit{ otherwise} \end{cases}$$

then

$$X = \sum_{i=1}^{n} = \textit{ \# of heads.}$$

What is the probability of the event $\{X = k\}$. For X to be k there must be exactly k coins which resulted in heads. We can choose among $\binom{n}{k}$ compatible events (set of variables), each having the same probability 2^{-n} to happen. It follows that

$$Pr\{X = k\} = \binom{n}{k} 2^{-n}.$$

What if the coin is not fair? We have the binomial distribution.

$$Pr\{X = k\} = \binom{n}{k} p^k (1-p)^{n-k}. \tag{2.3}$$

We would like to know what is the probability that X assumes a value far from the expected one. We can use the Chernoff Bound.

$$Pr\{X \leq (1-\varepsilon)np\} \leq e^{-\frac{\varepsilon pn}{3}}, \tag{2.4}$$

Suppose we choose $p = \frac{1}{2}$ and

$$\varepsilon = \sqrt{\frac{\log n}{n}} \tag{2.5}$$

Then we have

$$e^{-\frac{\frac{\log n}{n}\frac{1}{2}n}{3}} = e^{-\frac{1}{6}\log n} = \frac{1}{\sqrt[6]{n}}.$$

That is, the probability of error can be made as tiny as we want by increasing n.

2.1 Chain letters

Chain letters are (were) online petitions that spread virally through emails. They work in the following way. The petition's creator writes an email explaining a certain problem he or she wants to address. Then he/she signs the petition, sends it to a set of recipients and ask them to sign and forward the email. Some recipients may ignore the email, others may sign and forward it instead. One can model the spread using a tree T, where the root is the petition's creator and every node has a child for each person he forwarded the email to. Of course, email are privates. For an external observer to know the petition, at least one person must post the email on a public accessible place on the Web (e.g. public mailing lists). When a node v (person) *exposes* itself (i.e. publish the email), all its ancestors in the tree are revealed too (the path from the root to v).

We would like to know some information about the spread process, such as the reach of the petition. The full tree T is in general unknown, and what can be observed of it depends on how many and which nodes get exposed. We can assume that there exists a small probability $\delta > 0$ such that every node $v \in T$ gets exposed independently with that probability. Define T_δ to be the visible part of T reconstructed though the exposed nodes. Let V be the set of nodes in T_δ. Define $E \subseteq V$ to be the set of nodes that exposed themselves by disclosing their email to the public. Finally, let $L \subseteq E$ be the set of leaves in T_δ.

As we said, we would like to know $|V| = n$, but we only have E. Can we estimate n? If we assume that each node in E exposed itself independently from the others, we can make a reasonable guess about n by looking at T_δ [2]. In fact

$$n\delta = |E| \tag{2.6}$$

so

$$n = \frac{|E|}{\delta}. \tag{2.7}$$

At this point, we need to find δ' such that $\delta' \approx \delta$. One can wrongly think that

$$\delta' = \frac{|E|}{|T_\delta|}. \tag{2.8}$$

The problem here are the leaves of T_δ. If they hadn't exposed themselves, we wouldn't know about them (or other nodes). From the point of view of T_δ, they are *not* exposed independently with probability δ'; rather, they are exposed with probability 1. To compute δ' then, we excludes T_δ's leaves. What follows is an algorithm that outputs δ'.

```
**Input:** $E, L, V$.
if $|V| = 0$ then
    return 0
end
if $|V| = 1$ then
    return 1
end
return $\frac{|E|-|L|}{|V|-|L|}$
```

Algorithm 2: Estimation of δ

Finally, if $|V| \geq 2$,

$$n \approx \frac{|E|}{\delta'}. \tag{2.9}$$

Lemma 2. *If* $|V| \geq 2$ *then*

$$\Pr\left\{|\delta' - \delta| \geq \epsilon\delta\right\} \leq 2e^{\frac{1}{3}\epsilon^2\delta|V-L|}. \tag{2.10}$$

Proof. For each node i, let X_i be a random variable such that

$$X_i = \begin{cases} 1 & \textit{if node } i \textit{ exposes itself} \\ 0 & \textit{otherwise.} \end{cases} \tag{2.11}$$

Consider the three sets A, B and C built in the following way. We can scan the original tree in a bottom up fashion. At a certain time, node i is considered. If $i \notin A$, check if the node was exposed. If $X_i = 1$, add i to B and every its ancestor to A; otherwise, add i to C. Observe that the tripartition process does not disclose the value of X_i for all $i \in A$, and that $A = V - L$ and thus the set $E - L$ coincides with the set of exposed nodes in A. Then, we can apply the Chernoff bound, having $X = |E - L|$.

$$
\begin{aligned}
Pr\{X \geq (1+\epsilon)\delta|A|\} &= Pr\{|E-L| \geq |A|\delta + \epsilon\delta|A|\} && (2.12)\\
&= Pr\left\{\frac{|E-L|}{|A|} \geq \delta + \epsilon\delta\right\} && (2.13)\\
&= Pr\left\{\delta' - \delta \geq \epsilon\delta\right\} && \text{(We want the difference small enough)}\\
&= Pr\{|X - |A|\delta| \geq \epsilon\delta|A|\} && (2.14)\\
&\leq 2e^{-\frac{1}{3}\epsilon^2 E[X]} && (2.15)\\
&= 2e^{-\frac{1}{3}\epsilon^2\delta|V-L|}. && (2.16)
\end{aligned}
$$

□

Chapter 3

Submodular functions

In this chapter we first look at the k-max cover problem. Then, we will se that this problem is part of a sequence of problems having the same structure and that can be solved essentially with the same algorithm. This class of problems is the one of submodular function optimization problems.

3.1 k-max cover

Given $k \geq 1$ and a set of sets S, find a $S_k \subseteq S$, $|S_k| = k$, such that $|\bigcup_{s \in S_k}|$ is maximum. There is a similar problem, the set-cover problem, where we want to cover all the elements with the minimum number of sets. Here we have a budget of k sets, and we want to cover as many elements as possible. The following algorithm gives a greedy solution (in fact, in approximation algorithms there are not so many techniques you can use).

```
Input: $S, k$
$X_0 \leftarrow \emptyset$;
for $i = 1, \ldots, k$ do
    $s \leftarrow argmin_{s \in S} |X_{i-1} \cup s|$;
    $X_i \leftarrow X_{i-1} \cup s$
end
return $X_k$
```

Algorithm 3: A greedy algorithm for k-max cover.

Theorem 3. *Algorithm 3 returns a* $1 - \frac{1}{e}$ *approximation of the optimal solution.*

Proof. Define OPT to be the value of the optimal solution. Let $t_i = |X_i|$. The value of the greedy solution is going to be t_k. Define

$$n_i = |s_i - X_{i-1}|, \tag{3.1}$$

the number of new elements introduced at iteration i. Finally, define

$$g_i = OPT - t_i, \tag{3.2}$$

that indicates how far the greedy solution is far from the optimal one at iteration i. We want to prove

$$g_k \leq \frac{1}{e} \iff t_k \geq \left(1 - \frac{1}{e}\right) OPT \tag{3.3}$$

Lemma 3.

$$n_{i+1} \geq \frac{g_i}{k} \tag{3.4}$$

In words, it looks at how far it is from the optimal solution at iteration i and when it picks the $(i+1)$-th set it gets at least $\frac{1}{k}$ of what it was missing in the previous iteration.

Proof. Let $\{s_1^\star, s_2^\star, \ldots, s_k^\star\}$ be the sets representing the optimal solution and $O^\star$ their union. Let $T \subseteq O^\star$, maybe $T = O^\star - X_i$, for any i.

$$T \subseteq \bigcup_{i=1}^{k} s_i^\star \implies \exists i \ |s_i^\star \cap T| \geq \frac{|T|}{k} \tag{3.5}$$

The fact that $O^\star$ is realized by the choice of k sets that represents the optimal solution implies the existence of at least one set in $O^\star$ that covers at least a $\frac{1}{k}$ fraction of T. This is because if all sets in $O^\star$ covered less than that fraction, then overall they could not cover T and therefore they couldn't cover $O^\star$.

There exists one set in the optimal solution that covers at least $\frac{1}{k}$ fraction of T. Even for the specific T suggested above. But if this is true, then it means that there exists one set in S that covers at least a $\frac{1}{k}$ fraction of $O^\star - X_i = T$, and Algorithm 3 is going to pick at iteration $i+1$ the one set that maximizes the number of new elements it picks. So the new set Algorithm 3 will choose, will have at least as many elements as this one set in the optimal solution, that is $\frac{g_i}{k}$. □

Lemma 4.

$$g_i \leq \left(1 - \frac{1}{k}\right)^i OPT, \ \forall i. \tag{3.6}$$

Proof. We prove this by induction. For $i = 0$, $g_0 = OPT$ and so $g_0 \leq (1) OPT$ it is true. Let's assume Lemma 4 is true until i and we want to prove it for $i+1$.

$$
\begin{aligned}
g_{i+1} &= g_i - n_{i+1} && (3.7)\\
&\leq g_i - \frac{g_i}{k} && \text{(By Lemma 3)}\\
&= g_i\left(1 - \frac{1}{k}\right) && (3.8)\\
&\leq OPT\left(1 - \frac{1}{k}\right) && (g_i \leq OPT)\\
&< OPT\left(1 - \frac{1}{k}\right)\left(1 - \frac{1}{k}\right)^i && (3.9)\\
&= OPT\left(1 - \frac{1}{k}\right)^{i+1}. && (3.10)
\end{aligned}
$$

□

Instantiating Lemma 4 with $i = k$ we have

$$g_k \leq \left(1 - \frac{1}{k}\right)^k OPT = \frac{1}{e} OPT \tag{3.11}$$

□

In the next section we define what a submodular function is and then why k-max cover is a submodular optimization problem. We will discover that we can get a $(1 - \frac{1}{e})$ approximation for any problem in this class.

3.2 Submodular functions

Given a *ground set* V, a function $f : 2^V \to \mathbb{R}$ is a set function. We would like to maximize the value of this kind of functions, but if we have no information about them, the only thing we can do is to compute f for every set.

Suppose now we know about some of f's structure. The function f is *modular* (or *linear*) if, given some set $A \in 2^V$,

$$f(A) = \sum_{i \in A} w(i), \tag{3.12}$$

where $w(i)$ is the value of element i. The function 3.12 can be maximized easily by simply looking at every one-element set ($n = |V|$ queries or computations) and return the set which contains all the elements with positive weight.

Suppose now that we want to maximize the value of the function under the constraint that the set picked must have cardinality k. We choose the k biggest values. These problems are easy with modular functions, and hard with general set functions.

Definition 1 (Submodular function)**.** *Submodular functions are more general than linear functions and are less general than set functions. A function f is submodular if*

$$\forall S, T \subseteq V \ f(S) + f(T) \geq f(S \cup T) + f(S \cap T). \tag{3.13}$$

They are a generalization of modular functions. In fact, one can prove that if f is modular then Equation 3.13 holds only with equality.

Let $S = A$, $B \subseteq A$, $x \notin A$, $T = B \cup \{x\}$. Let's apply the submodularity definition

$$f(A) + f(B \cup \{x\}) = f(A \cup \{X\}) + f(B) \tag{3.14}$$

that implies

$$f(B \cup \{x\}) - f(B) \geq f(A \cup \{x\}) - f(A). \tag{3.15}$$

Definition 2. *We define the* marginal return *of adding an extra element x to a set A to be*

$$\Delta(x|A) = f(A \cup \{x\}) - f(A). \tag{3.16}$$

Putting together equations 3.15 and 3.16 we get the economics Law of diminishing returns.

Definition 3 (Law of diminishing returns)**.** *Given A, B, with $B \subseteq A$, and an item $x \notin B$, adding x to B increase the value of that set more than adding that element to a bigger set A which includes B. Formally*

$$\forall A, B \subseteq A, x \notin B \ \Delta(x|B) \geq \Delta(x|A). \tag{3.17}$$

We proved that Definition 1 implies Definition 3. We now prove the opposite direction, to show that the two definitions are equivalent.

Bibliography

[1] Abrahao et al. “Trace Complexity of Network Inference”. In: *Knowledge Discovery and Data Mining* (2013).

[2] Chierichetti, Kleinberg, and Liben-Nowell. “Reconstructing Patterns of Information Diffusion from Incomplete Observations”. In: *Advances in Neural Information Processing Systems* (2011).

[3] Gomez-Rodriguez, Leskovec, and Krause. “Inferring Networks of Diffusion and Influence”. In: *Knowledge Discovery and Data Mining* (2010).